LESSON BOOK



Time-Series Forecasting for Energy

What is forecastable, how to know a forecast is good, and the decisions it is for

Sixteen lessons · 363 pages

Ricardo Marquez, Ph.D.

ricardo-marquez.com

Time-Series Forecasting for Energy

From the forecasting problem to risk-aware fleet dispatch

Track A lessons, written up as a book

Ricardo Marquez

October 26, 2026 (one chapter per lesson, sixteen lessons)

How to read this book

Each chapter of this book expands one lesson of "Track A", a sequence on time-series forecasting built around energy problems rather than generic examples. The progression is

what is forecastable → baselines → features → models → validation → probabilistic forecasts → fleet forecasting → operational decisions.

The running example throughout is an aggregator's fleet of residential thermostats, electric vehicles and batteries, and the quantity most often forecast is not load but *flexibility*: how much the fleet's consumption could be changed on request. The emphasis is on the physical and operational logic of each method, not on statistics for its own sake. Where the field uses a shorthand term, a *Jargon* box expands it once and the text then uses the expanded meaning. Every symbol is defined in the notation chapter that follows and is used with the same meaning in every chapter. Each chapter closes with the conceptual questions the lesson raised and a set of exercises.

Contents

How to read this book **ii**

Notation and vocabulary **vii**

1 The forecasting problem **1**
1.1 What a forecast is . . . 1
1.2 The target variable in an energy system . . . 3
1.3 Issue time and horizon . . . 5
1.4 The information set . . . 8
1.5 The forecast as a conditional statement . . . 10
1.6 The forecasting loop and a worked example . . . 12
1.7 What to retain . . . 15
1.8 Exercises . . . 16

2 Structure of energy time series **17**
2.1 The decomposition of a load series . . . 17
2.2 Calendar structure . . . 20
2.3 Weather dependence . . . 24
2.4 Persistence and the autocorrelation function . . . 27
2.5 Fleet aggregation and common-mode structure . . . 31
2.6 Nonstationarity, control and the consequences for modeling . . . 33
2.7 What to retain . . . 37
2.8 Exercises . . . 37

3 Baseline forecasts **39**
3.1 The role of a baseline . . . 39
3.2 Naive and seasonal baselines . . . 41
3.3 Baselines that use domain knowledge . . . 44
3.4 Error sources and failure modes . . . 46
3.5 Measuring improvement over a baseline . . . 48
3.6 Baselines for fleet flexibility . . . 51
3.7 What to retain . . . 54
3.8 Exercises . . . 54

4 Feature engineering **56**
4.1 The feature vector and the supervised table . . . 56
4.2 Lag and rolling features . . . 59
4.3 Calendar and weather features . . . 63
4.4 State and control features for a controllable fleet . . . 67
4.5 Target transformations and feature selection . . . 69
4.6 Leakage, latency and feature hygiene . . . 71

4.7 What to retain . . . 74
4.8 Exercises . . . 74

5 Regression and tree models 76
5.1 Supervised learning of the forecast function . . . 76
5.2 Linear models and regularization . . . 79
5.3 Decision trees and ensembles . . . 82
5.4 Gradient boosting . . . 85
5.5 Choice of model class . . . 88
5.6 Diagnostics after fitting . . . 92
5.7 What to retain . . . 95
5.8 Exercises . . . 95

6 Autoregressive models and residual dynamics 97
6.1 The residual as the starting point . . . 97
6.2 Autoregressive models . . . 99
6.3 Autoregressive residual correction . . . 102
6.4 Stationarity, differencing and the ARIMA family . . . 105
6.5 The autocorrelation and partial autocorrelation functions . . . 108
6.6 Model comparison, multi-step forecasting and horizon . . . 110
6.7 What to retain . . . 115
6.8 Exercises . . . 116

7 Recurrent networks 117
7.1 From explicit lags to a learned state . . . 117
7.2 The recurrent network . . . 119
7.3 Vanishing gradients and the long short-term memory cell . . . 122
7.4 The gated recurrent unit . . . 126
7.5 Architectures for forecasting . . . 129
7.6 When to use a recurrent model . . . 132
7.7 What to retain . . . 136
7.8 Exercises . . . 136

8 Attention and transformers 138
8.1 The bottleneck of a single recurrent state . . . 138
8.2 The attention mechanism . . . 140
8.3 Transformers . . . 144
8.4 The temporal fusion transformer . . . 146
8.5 Interpretation and model choice . . . 149
8.6 Data requirements and the fleet case . . . 152
8.7 What to retain . . . 154
8.8 Exercises . . . 154

9 Forecast validation 157
9.1 Time order and the validation split . . . 157
9.2 Error metrics . . . 161
9.3 Slicing by regime and horizon . . . 163
9.4 Event-level and decision-aware validation . . . 166
9.5 Leakage, revisions and drift . . . 168
9.6 Validation protocol and a worked comparison . . . 172
9.7 What to retain . . . 178
9.8 Exercises . . . 178

10 Probabilistic forecasting **181**
10.1 From a point forecast to a distribution . . . 181
10.2 Calibration and sharpness . . . 183
10.3 Learning a predictive distribution . . . 185
10.4 Sources of uncertainty and their propagation . . . 188
10.5 Evaluating probabilistic forecasts . . . 190
10.6 From distribution to decision . . . 194
10.7 What to retain . . . 200
10.8 Exercises . . . 201

11 Fleet forecasting **202**
11.1 The fleet problem . . . 202
11.2 Local, cohort and global models . . . 203
11.3 Bottom-up, top-down and the statistics of aggregation . . . 206
11.4 Hierarchical forecasting and reconciliation . . . 210
11.5 Composition and availability . . . 213
11.6 Heterogeneity and fleet design . . . 216
11.7 What to retain . . . 221
11.8 Exercises . . . 221

12 Flexibility and counterfactual forecasting **223**
12.1 The missing world . . . 223
12.2 Two architectures and the baseline problem . . . 226
12.3 Response as a function of action and state . . . 230
12.4 Causal estimation at fleet scale . . . 233
12.5 Uncertainty and the interventional distribution . . . 238
12.6 Architecture, validation and the worked event . . . 240
12.7 What to retain . . . 246
12.8 Exercises . . . 246

13 Forecasting under distribution shift **249**
13.1 The changed data-generating process . . . 249
13.2 Sources of shift in a fleet . . . 252
13.3 Recognizing unfamiliar inputs . . . 255
13.4 Monitoring and diagnosis . . . 259
13.5 Regimes, actions and the optimizer . . . 264
13.6 Operating under shift . . . 268
13.7 What to retain . . . 272
13.8 Exercises . . . 274

14 Forecast to optimization **275**
14.1 Information versus action . . . 275
14.2 Reliability as a constraint . . . 277
14.3 Cost, quantiles and portfolios . . . 281
14.4 Multi-period dispatch and model predictive control . . . 284
14.5 Stochastic and robust formulations . . . 287
14.6 Designing the interface and closing the loop . . . 291
14.7 What to retain . . . 297
14.8 Exercises . . . 297

15 Production forecasting systems **300**
15.1 From model to system 300
15.2 The data pipeline 303
15.3 Features and serving 307
15.4 Monitoring 311
15.5 Failure handling and retraining 313
15.6 Learning from operation and the control loop 318
15.7 What to retain 324
15.8 Exercises 324

16 Capstone: 75 MW on the hottest day **327**
16.1 The case and the decision 327
16.2 The baseline and response models 330
16.3 Fleet representation and uncertainty 334
16.4 The hottest day and the optimizer 337
16.5 Closing the loop and validating the design 341
16.6 The pushback round and the end-to-end design 346
16.7 What to retain 352
16.8 Exercises 353

Notation and vocabulary

The same symbols are used throughout. Time t is the instant at which a forecast is *issued*; h is the *horizon*, the time between issue and target; power is in kilowatts (kW) or megawatts (MW); temperature is in degrees Celsius.

Symbols

Symbol	Unit	Meaning
t	time	Forecast issue time: the moment the prediction is made.
h	time	Forecast horizon; the target time is $t + h$.
y_{t+h}	–	The target: the quantity being predicted.
$\hat{y}_{t+h}$	–	A point forecast of the target.
$\mathcal{I}_t$	–	Information set: everything genuinely known at issue time t.
$p(y_{t+h} \mid \mathcal{I}_t)$	–	The probabilistic forecast: a distribution over the target given the information.
u	–	A planned control action (setpoint change, start, duration, cohort); the target may depend on it.
P_t	kW, MW	Load (power) at time t.
$P^{\text{base}}(t)$	MW	Baseline load: what the fleet would consume without intervention.
$P^{\text{ctrl}}(t \mid u)$	MW	Controlled load: what it would consume under action u.
$F(t)$	MW	Flexibility, $P^{\text{base}} - P^{\text{ctrl}}$; a counterfactual quantity.
T_t, T_{out}	°C	Outdoor temperature at t.
$\hat{T}_{t+h\mid t}$	°C	Weather forecast for $t + h$ as issued at t.
CD_t, HD_t	°C	Cooling and heating degrees, $\max(T_t - T^*, 0)$ and $\max(T^* - T_t, 0)$ about a base T^*.
RH_t	%	Relative humidity.
N_{avail}	–	Number of devices available (connected, enrolled, not opted out).
D_t	–	Duty cycle: fraction of the fleet's compressors running.
X_t	–	Feature vector built from $\mathcal{I}_t$: lags, weather, calendar, state.
$\boldsymbol{\beta}$	–	Coefficients of a linear model, $\hat{y} = X_t\boldsymbol{\beta}$.
λ	–	Regularization weight (ridge, lasso).
η	–	Learning rate of a boosted ensemble.

Symbol	Unit	Meaning
$T_m(X)$	–	The m-th tree of an ensemble.
ε_t	–	Noise or residual at t.
e_t	–	Forecast error, $y_t - \hat{y}_t$.
ϕ_k, θ_k	–	Autoregressive and moving-average coefficients.
p, d, q	–	ARIMA orders: AR lags, differences, MA lags; $(P, D, Q)_s$ their seasonal counterparts at period s.
ACF(k), PACF(k)	–	Autocorrelation and partial autocorrelation at lag k.
$\mathbf{h}_t$, $\mathbf{c}_t$	–	Hidden state and cell state of a recurrent network at step t (bold, to distinguish from the horizon h).
$\mathbf{Q}, \mathbf{K}, \mathbf{V}$	–	Query, key and value matrices of an attention layer; d_k the key dimension.
MAE, RMSE, MAPE, Bias	–	Mean absolute error, root-mean-square error, mean absolute percentage error, mean error.
Skill	–	$1 - \mathrm{RMSE}_{\mathrm{model}}/\mathrm{RMSE}_{\mathrm{baseline}}$.
τ	–	Quantile level; $\hat{y}_\tau$ the forecast τ-quantile.
$L_\tau(y, \hat{y})$	–	Pinball (quantile) loss at level τ.
CRPS	–	Continuous ranked probability score of a distributional forecast.
$y_{i,t}$	–	The series of device (or cohort) i at time t.
x_t, w_t	–	Latent physical state and disturbance in the state-space view, $x_{t+1} = g(x_t, u_t, w_t)$, $P_t = h(x_t)$.

Vocabulary

Issue time, target time, horizon
: When the forecast is made, when the predicted quantity occurs, and the gap between them.

Information set
: Everything that actually exists at issue time; a forecast may use nothing outside it.

Data leakage
: Training or evaluating with information that would not exist at issue time, for example an observed future temperature in place of the forecast that was available.

Point forecast, probabilistic forecast
: A single number versus a distribution (or a set of quantiles) for the target.

Baseline (model)
: A simple forecast a sophisticated model must beat: persistence, seasonal persistence, similar-day average, weather regression.

Baseline (load)
: In demand response, the counterfactual load without intervention. The two meanings coexist in this book; the context makes clear which is meant.

Persistence
: The forecast that the next value equals the last one; seasonal persistence uses the value one cycle earlier.

Skill score
: Error reduction relative to a baseline.

Trend, seasonality, autocorrelation
: Slow change; repeating daily, weekly or annual cycles; dependence between a series and its own past.

Exogenous variable
: A predictor outside the series itself: weather, calendar, device state.

Stationarity
: Constancy of a series' statistical properties in time; differencing is the usual way to obtain it.

Feature engineering
: Turning raw timestamps, loads and weather into the predictor vector X_t: lags, rolling statistics, degree variables, cyclic encodings.

Regularization
: A penalty on model complexity added to the training error (ridge, lasso) to improve generalization.

Gradient boosting
: An ensemble of small trees fitted sequentially, each to the residual of the ones before.

Recurrent network
: A model whose hidden state summarizes the history it has seen; LSTM and GRU are gated variants that control what is remembered.

Attention
: A mechanism that weights past observations by learned relevance to the current prediction; the basis of transformers.

Rolling-origin validation
: Evaluating a forecaster by repeatedly training up to a date and testing on the period after it, never on a random row split.

Calibration, sharpness
: Whether stated probabilities match observed frequencies; how narrow the forecast distribution is.

Pinball loss
: The loss whose minimizer is a quantile.

Global model, cohort model, local model
: One model for all series, one per group of similar series, one per series.

Hierarchical forecasting, reconciliation
: Forecasting at several aggregation levels and adjusting the forecasts so they add up.

Common-mode uncertainty
: Error shared by every device in a fleet (for example a weather forecast error); aggregation does not average it away.

Counterfactual
: What would have happened under a different intervention; never observed alongside what did happen.

Chapter 1

The forecasting problem: target, horizon and information set

1.1 What a forecast is

Suppose it is 2:00 PM on a hot afternoon. An aggregator operates a fleet of residential thermostats and needs to know how much of that fleet's load it can curtail between 5:00 and 6:00 PM. The natural first move is to start building a forecasting model. That move is premature, because the question as stated does not yet name a quantity. At least three different numbers are hiding inside it, and they have different values, different inputs and different error consequences. The chapter's question is therefore

what exactly is being predicted, for what moment, and from what information?

An *aggregator* is a company that enrolls many small customer-owned devices — thermostats, water heaters, batteries, electric-vehicle chargers — and operates them together as one resource it can sell to a utility or a market. To *curtail* the fleet is to reduce its electrical consumption on request, which for a thermostat fleet means raising cooling setpoints so that compressors stop. The *setpoint* is the indoor temperature a thermostat is trying to hold.

Jargon: issue time, target, horizon, information set, data leakage, counterfactual

The *issue time* t is the moment at which the forecast is made and after which no new information may enter it. The *target* y_{t+h} is the quantity being predicted, at the *target time* $t + h$; the gap h is the *horizon*. The *information set* $\mathcal{I}_t$ is everything that genuinely exists and is accessible at the issue time. *Data leakage* is the use, in training or in evaluation, of anything outside $\mathcal{I}_t$ — most commonly an observed future weather value in place of the forecast that was actually available. A *counterfactual* quantity is one defined by what would have happened under an action that was not taken; it is never observed alongside what did happen.

Key idea

The fundamental object of forecasting is not a number $\hat{y}$ but the conditional distribution $p(y_{t+h} \mid \mathcal{I}_t)$: a statement about a named target, at a named time, given only what was genuinely known when the statement was made.

This section fixes the three things a forecasting problem must name before any model is chosen, writes the general formulation they imply, and distinguishes the deterministic from the probabilistic version of it.

1.1.1 Three targets hiding in one question

"How much load can we curtail from 5:00 to 6:00 PM" resolves into three distinct quantities. The first is the *baseline load* $P^{\text{base}}(t)$: what the fleet would consume at that hour if the aggregator did nothing. The second is the *controlled load* $P^{\text{ctrl}}(t \mid u)$: what it would consume under a specific intervention u. The third is the *available flexibility* $F(t) = P^{\text{base}}(t) - P^{\text{ctrl}}(t \mid u)$, the reduction the intervention achieves. With the illustrative numbers used throughout this chapter, $P^{\text{base}}(17{:}00) = 120\,\text{MW}$, $P^{\text{ctrl}}(17{:}00 \mid u) = 75\,\text{MW}$, and the flexibility is 45 MW.

Only the third number answers the operational question, and it is the only one of the three that is never directly measured. Section 1.2 treats each in turn. The immediate point is structural: a forecasting problem is not defined by the data available or by the model class, but by a choice of target, and that choice is made by the person asking the question, not by the modeler.

1.1.2 The triple that defines a forecasting problem

Every forecasting problem is specified by three things, and changing any one of them produces a different problem:

(i) the **target** y_{t+h} — which physical quantity, for which population of devices, at which temporal resolution;

(ii) the **horizon** h and the issue time t — how far ahead, measured from when;

(iii) the **information set** $\mathcal{I}_t$ — what is genuinely available at t.

Two problems that share a target but differ in horizon need different predictors and often different model classes (Section 1.3). Two problems that share a target and a horizon but differ in what is known at issue time — for example a day-ahead forecast made with and without a numerical weather prediction — have different attainable accuracy (Section 1.4).

A fourth question sits beside the triple and determines how the forecast should be summarized: which decision will consume it. That question does not change the distribution $p(y_{t+h} \mid \mathcal{I}_t)$, but it changes which feature of the distribution is reported and which errors are expensive. Section 1.5 takes it up.

1.1.3 The general formulation

At the issue time t the forecaster holds an information set, which for the aggregator contains the recent fleet load, the recent and current outdoor temperature, a weather forecast for the hours ahead, the state of the devices, the calendar, and the fleet's historical behavior,

$$\mathcal{I}_t = \{\, P_t,\ P_{t-1},\ \ldots,\ T_t,\ \hat{T}_{t+1:t+H|t},\ \text{device state},\ \text{calendar},\ \text{history} \,\}. \tag{1.1}$$

The target is y_{t+h}. The deterministic version of the problem is to produce a single number,

$$\hat{y}_{t+h} = f(\mathcal{I}_t), \tag{1.2}$$

and the probabilistic version is to produce a distribution,

$$p(y_{t+h} \mid \mathcal{I}_t). \tag{1.3}$$

Equation (1.3) is the more complete problem and equation (1.2) is a summary of it. The question it answers is not "what will the load be" but "given everything known now, which values of the future load are plausible, and with what probabilities". The remainder of the book is organized around this object: Chapters 2 to 8 build models that estimate its center, Chapter 9 tests them, Chapter 10 estimates its spread and shape, and Chapter 11 studies what happens to it when thousands of series are forecast at once.

Principle 1.1. *A forecasting problem is defined before any model is chosen, by naming the target, the issue time and horizon, and the information set. Model selection is downstream of that definition, and an undefined target cannot be forecast well by any method.*

1.2 The target variable in an energy system

The three candidate targets of Section 1.1.1 are different functions of the same fleet. This section defines each, identifies which is a conventional forecasting problem and which is not, and states what makes a target well defined.

1.2.1 Baseline load

The baseline load $P^{\text{base}}(t)$ is the fleet's consumption in the absence of intervention. It is a conventional load forecast: a function of the recent load, the weather and the calendar,

$$P^{\text{base}}(t+h) = f(P_t,\ P_{t-1},\ T_{\text{out}},\ \text{hour},\ \text{day type},\ \ldots), \tag{1.4}$$

and it answers the question "what will happen naturally". Every technique in this book applies to it directly, because the quantity is observed whenever no intervention occurs, so training data and verification data both exist in quantity.

The term *baseline* carries two meanings in this field and both appear in this book. Here it means the counterfactual load without intervention, which is the demand-response usage. In Chapter 3 it will mean a simple reference forecast that a sophisticated model must beat. The context separates them; where confusion is possible, this book writes "baseline load" for the first and "baseline model" for the second.

1.2.2 Controlled load

Now suppose the aggregator raises thermostat setpoints. The controlled load $P^{\text{ctrl}}(t \mid u)$ is what the fleet consumes under that specific action. The conditioning argument u is not a scalar; it is a description of the intervention,

$$u = \{\,\text{setpoint change},\ \text{start time},\ \text{duration},\ \text{device cohort}\,\}, \tag{1.5}$$

where a *cohort* is a subset of the fleet selected to receive the command, for instance all thermostats in one climate zone. The forecast is accordingly

$$P^{\text{ctrl}}(t+h \mid u) = f(\mathcal{I}_t,\ u). \tag{1.6}$$

This is a different problem from equation (1.4), because the answer depends on what the aggregator decides to do. Two features follow. The target is not a property of the world alone, so a model fitted to historical load without recording which commands were sent is fitting a mixture of the two quantities. And the data available for fitting is much thinner: the baseline load is observed on every non-event day, while $P^{\text{ctrl}}(\cdot \mid u)$ is observed only on the days on which that particular kind of u was actually dispatched.

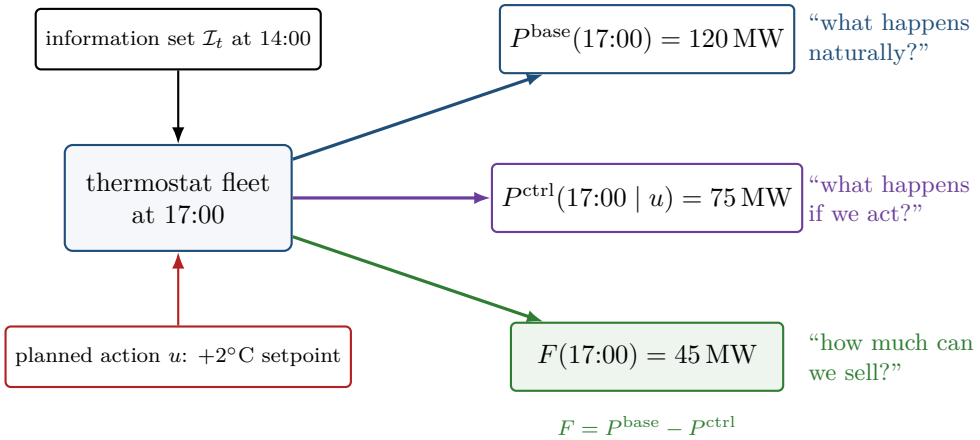
1.2.3 Available flexibility as a counterfactual

The quantity the aggregator sells is the difference,

$$F(t) = P^{\text{base}}(t) - P^{\text{ctrl}}(t \mid u) = 120 - 75 = 45\,\text{MW}, \tag{1.7}$$

a load reduction of 45 MW. Figure 1.1 draws the three targets together: a single box on the left for the fleet, with the information set entering it from above and the planned action u entering from below, and three target boxes on the right for the baseline load, the controlled load and their difference, each labeled with the question it answers and its illustrative value. The figure makes visible that only the two upper targets are powers that could in principle be metered; the third is constructed from them.

That construction is the difficulty. Flexibility is a *counterfactual* quantity: if the aggregator dispatches the fleet it observes P^{ctrl}, and the P^{base} for the same devices under the same weather at the same instant then does not exist to be measured. Figure 1.2 shows the two worlds against time. The horizontal axis is the clock from 16:00 to 19:00 and the vertical axis is the fleet's power; the two curves coincide before 17:00, because nothing has been done yet, and separate afterward, with the controlled trajectory drawn solid because it is the one observed and the baseline trajectory dashed because it is not. The shaded gap during the event hour is F. Since the dashed curve is never recorded, every flexibility measurement is really a flexibility *estimate*, obtained by substituting a modeled baseline for the missing world, and the error of that model enters the quantity being sold. Chapter 3 builds the estimators, and the rebound visible after 18:00 in the figure — the controlled curve overshooting the baseline as warmed buildings cool back down — is a second reason the two worlds cannot simply be spliced.



The upper two targets are powers that could be metered;
the third is their difference and is never measured directly.

Figure 1.1: The three candidate targets inside one operational question. The fleet is shown with the information set entering from above and the planned action u from below; each target box states the quantity, its illustrative value from Section 1.1.1, and the question it answers. Only P^{ctrl} is observed when the fleet is dispatched, and only P^{base} when it is not.

1.2.4 What makes a target well defined

A target is well defined when it is reproducible: two analysts given the same data and the same specification compute the same series. For the quantities above that requires four things to be stated. The *population*: which devices, enrolled on what date, including or excluding those that have opted out. The *resolution*: a 5-minute average power is a different series from an hourly average, and a peak is different from either. The *units and sign convention*: this book takes $F > 0$ to mean a load reduction. And, for the control-dependent targets, the *action* u in the sense of equation (1.5).

Omitting any of the four produces a target that looks like a single series but is a mixture. A common instance is a flexibility series assembled from events of varying duration: a 15-minute

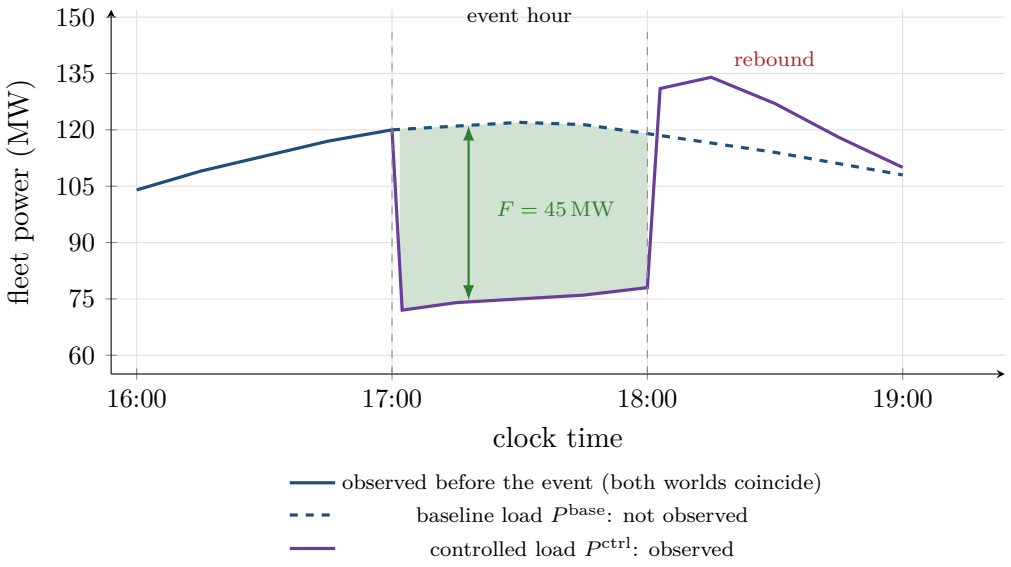


Figure 1.2: Flexibility as a counterfactual. Fleet power against clock time for a one-hour curtailment beginning at 17:00. The two worlds coincide before the event; afterward the controlled trajectory is observed and the baseline trajectory is not. The shaded area is F of equation (1.7), averaging 45 MW over the event hour. Trajectories are illustrative, drawn through the stated values of Section 1.1.1.

curtailment and a three-hour curtailment of the same fleet yield very different reductions, and averaging them produces a number that describes no event the fleet ever performed. A second instance is a fleet whose enrollment grew over the sample, so that the target's trend is partly an accounting artifact rather than a physical change.

1.3 Issue time and horizon

The same target at two different horizons is two different statistical problems. This section separates the issue time from the target time, works through three horizons for one target, and draws the consequence for model selection.

1.3.1 Issue time and target time

Fix $t = 14{:}00$ and a target at 17:00, so $h = 3$ hours. The rule is that the model may use only information available at 14:00. Figure 1.3 draws the arrangement: the horizontal line is the clock, the left region shaded blue is time that has already happened at the issue instant and the right region shaded red is time that has not, the arrow spanning them is the horizon, and the two labeled items below the axis are a temperature at 16:30 in two versions. The observed version $T_{16:30}$ is marked unavailable: at 2:00 PM it does not exist. The forecast version $\hat{T}_{16:30|14:00}$ is available, because it is a weather forecast issued at 2:00 PM for 4:30 PM.

The distinction is purely about when a value came into existence, and it is enforced not only at prediction time but also during training and evaluation. A model trained on $T_{16:30}$ learns the relationship between load and the temperature that actually occurred; operationally it will be fed $\hat{T}_{16:30|14:00}$ instead, which is a different and noisier variable. Section 1.4.3 quantifies the gap.

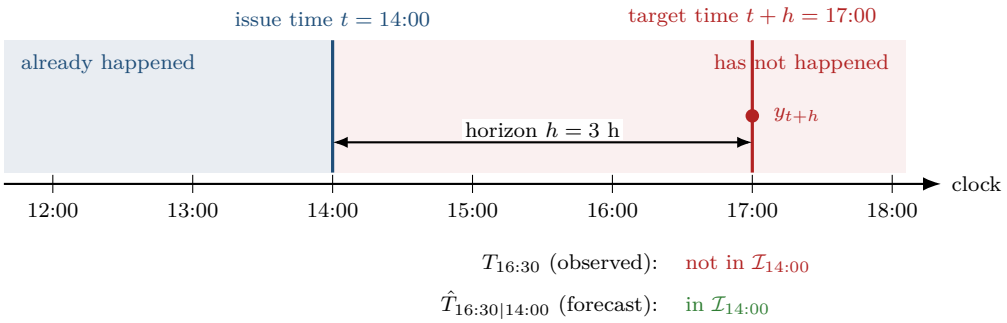


Figure 1.3: Issue time, horizon and target time. The clock runs left to right; the blue region is time that has already elapsed at the issue instant and the red region is time that has not. The two entries below the axis are the same physical temperature in two versions: the observed value, which does not yet exist at 14:00, and the forecast value issued at 14:00, which does.

1.3.2 The horizon changes which information matters

Take the fleet load as the target and vary only the horizon. Five minutes ahead, the load is dominated by the load now, because almost nothing in the physical system has had time to change: the same compressors are running, the buildings have the same indoor temperatures, and

$$P_{t+5\,\mathrm{min}} \approx f(P_t,\ P_{t-5\,\mathrm{min}},\ \ldots) \tag{1.8}$$

is an accurate model. The simplest such model, $\hat{y}_{t+h} = y_t$, is called *persistence*, and at this horizon it is a strong competitor.

Three hours ahead, persistence is weak and the drivers are the weather forecast, the time of day and the fleet's thermal state,

$$P_{t+3\,\mathrm{h}} = f(\text{recent load},\ \hat{T}_{t+3\mathrm{h}|t},\ \text{hour},\ \text{thermal state}). \tag{1.9}$$

A day ahead, the instantaneous state has almost no value — the fleet will have passed through an entire evening and night — and what remains informative is the weather forecast, the day of week, the season and the enrolled population's behavior. Figure 1.4 shows the shift as three stacked bars, one per horizon; the vertical axis is the share of predictive information attributable to each of three groups of inputs, and the three segments of each bar are the recent load, the weather forecast, and the calendar and behavior terms. The shares are illustrative, but the ordering is general: the recent-load segment shrinks monotonically with the horizon while the weather and calendar segments grow.

1.3.3 There is no universal best forecasting model

The practical consequence is that a claim of the form "gradient boosting is the best load forecasting method" is not well posed until the horizon is fixed. Table 1.1 gives the three horizons with the information that dominates each, the decision each supports in an aggregator's operation, and the model class that typically wins. The rows are not three tunings of one model; they are three problems with different predictors, different achievable accuracy and different reference baselines.

The same reasoning applies to the target. A method that forecasts P^{base} well need not forecast F well, because the second is a difference of two forecasts and inherits the errors of both. For the rest of the book, every statement about accuracy is a statement about a specific target at a specific horizon, and comparisons are made within a row of Table 1.1, never across rows.

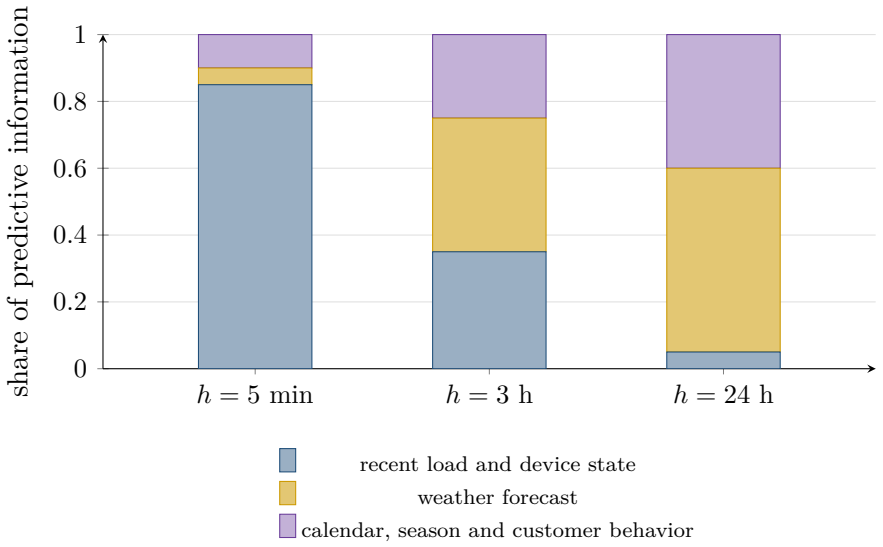


Figure 1.4: How the dominant information changes with the horizon, for one fixed target. Each bar divides the predictive information into three groups of inputs; shares are illustrative and are the ones used in Section 1.3.2. The recent-load share falls from 0.85 at five minutes to 0.05 at a day, and the weather share rises from 0.05 to 0.55.

Table 1.1: The same target at three horizons: three different problems.

Horizon	Dominant information	Decision supported	Typical winning model
5 min	current load, compressor state, indoor temperatures	real-time correction of a dispatch already running	persistence and short-lag autoregression
3 h	recent load plus the weather forecast and the fleet's thermal state	whether to call an event this afternoon, and how large	regression or gradient boosting on lag and weather features
24 h	weather forecast, day type, season, enrolled population, behavior	day-ahead capacity offer to the utility	weather-driven model with calendar features, reported probabilistically

1.4 The information set

The information set is the formal statement of what a forecast is allowed to use. This section enumerates its contents for the aggregator, separates forecast weather from observed weather, shows what happens when the separation is violated, and notes that in practice the set is smaller than the clock suggests.

1.4.1 Contents of the information set at issue time

Equation (1.1) lists the groups abstractly; Table 1.2 instantiates them for the aggregator at 14:00, with the nature of each entry in the last column. The grouping matters because the four groups behave differently. Measured history is exact but stale the instant it is recorded. Device state is partly measured and partly inferred, since a thermostat reports its indoor temperature but not the thermal mass of the building around it. Exogenous forecasts are themselves uncertain and carry their own error into the chain. The calendar is known exactly and arbitrarily far ahead, which makes it the only input whose quality does not degrade with the horizon.

The information set also grows. Figure 1.5 draws $\mathcal{I}_{13:00}$, $\mathcal{I}_{14:00}$ and $\mathcal{I}_{15:00}$ as nested rectangles over a common time axis: each one contains all of its predecessors and extends further to the right, and the target at 17:00 is marked outside all three. Two properties are visible. A forecast for 17:00 issued at 15:00 can be no worse than one issued at 14:00, because it conditions on a superset of the information, which is why operational forecasts are reissued as time passes. And the target itself never enters any of the sets drawn, which is what makes the problem a forecasting problem rather than an interpolation.

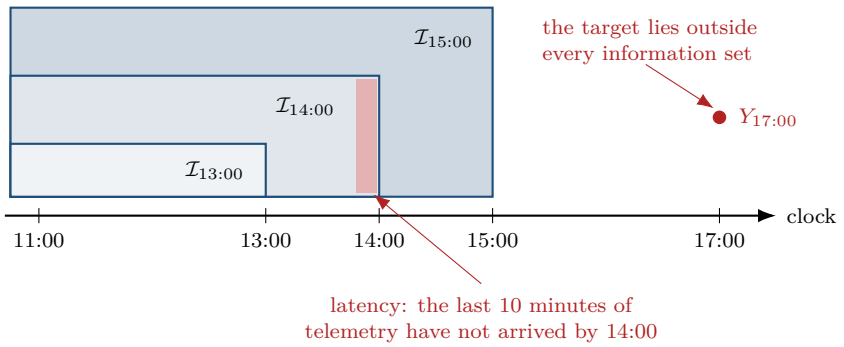
Table 1.2: The aggregator's information set $\mathcal{I}_{14:00}$, by group.

Group	Contents at 14:00	Nature
Measured history	fleet load $P_t, P_{t-1}, \ldots$; outdoor temperature T_t and its recent path; past events and their verified response	exact but subject to latency
Device state	indoor temperatures T_{in}, setpoints T_{set}, compressor on/off, duty cycle D_t, devices available N_{avail}	partly measured, partly inferred
Exogenous forecasts	$\hat{T}_{t+1:t+H\|t}$: temperature, humidity RH, cloud cover, wind, for every hour ahead	uncertain; error grows with h
Calendar and known plans	hour, day of week, holiday, season; scheduled maintenance; the aggregator's own planned action u	known exactly, arbitrarily far ahead

1.4.2 Forecast weather and observed weather are different variables

Weather enters an energy forecast in two incompatible forms. The observed temperature T_s at a past time $s \le t$ is a measurement and belongs to $\mathcal{I}_t$. The temperature at a future time $t + h$ does not belong to $\mathcal{I}_t$ in any form; what belongs is the forecast $\hat{T}_{t+h|t}$, a prediction issued by a weather service at or before t. These are not two names for one quantity. The forecast has its own error, which grows with h, varies by variable — temperature is forecast far better than cloud cover or wind — and is correlated across the hours of a day, because a numerical weather prediction run that is too warm at noon is usually too warm at 3:00 PM as well.

The consequence is that an energy forecast at horizon h is a composition of two forecasts, and its error has a component inherited from the weather forecast. If the load responds to



$\mathcal{I}_{13:00} \subset \mathcal{I}_{14:00} \subset \mathcal{I}_{15:00}$: each issue time conditions on a superset of the one before.

Figure 1.5: The information set as a growing family. Each rectangle is the information available at one issue time, drawn over a common clock axis; later sets contain earlier ones. The red sliver inside $\mathcal{I}_{14:00}$ is the telemetry that exists physically but has not been delivered by 14:00 (Section 1.4.4). The target at 17:00 belongs to none of the sets.

temperature with sensitivity $\partial P/\partial T$ and the weather forecast has temperature error standard deviation σ_T, then under independence the inherited term adds in quadrature,

$$\sigma_P^2 \;=\; \sigma_{\text{model}}^2 \;+\; \left(\frac{\partial P}{\partial T}\right)^2 \sigma_T^2. \tag{1.10}$$

For the fleet of this chapter, a plausible sensitivity is 4 MW per degree Celsius, and a three-hour-ahead temperature forecast might have $\sigma_T = 1.5°\text{C}$; the inherited term is then 6 MW, which is larger than the error of many well-built load models. Weather forecast error is not a detail of the input pipeline; at useful horizons it is often the dominant term.

1.4.3 Data leakage

Suppose a modeler fits

$$P_{17:00} = f(T^{\text{obs}}_{17:00}) \tag{1.11}$$

using the temperature that actually occurred, and reports excellent accuracy on held-out data. The model is not wrong as a description of the relationship between load and temperature. It is unusable, because at 14:00 the quantity $T^{\text{obs}}_{17:00}$ does not exist. The operationally realizable model is

$$P_{17:00} = f(\hat{T}_{17:00|14:00}), \tag{1.12}$$

and equation (1.10) says by how much its error exceeds the reported figure. A model with a 3 MW in-sample error and the sensitivity of the previous subsection has an operational error of $\sqrt{3^2 + 6^2} = 6.7$ MW, more than double the number that was reported.

Leakage of this kind is easy to introduce accidentally, because historical archives usually store the observed weather and not the forecast that was current at each issue time. Three habits prevent it. Reconstruct the information set explicitly for each training row, by asking what existed at that row's issue time, rather than selecting columns from a table. Store weather forecasts with both their valid time and their issue time, so that $\hat{T}_{t+h|t}$ can be retrieved for the correct t. And evaluate with the same inputs that will be used operationally, a discipline taken up in Chapter 9.

Leakage through the target. The same error occurs on the target side when the series being predicted has been revised after the fact. A verified flexibility number computed weeks

later with a settled baseline model is not the number an operator could have had, and a model trained to predict the settled figure while evaluated against it will overstate what the operator can expect.

1.4.4 Latency makes the information set smaller than the clock

In a deployed system the information set at 14:00 does not extend to 14:00. Meter reads are aggregated and transmitted on a cycle; a reading timestamped 13:55 may not be available to the model until 14:07. The effective information set therefore ends at some $t - \delta$, with δ a delivery latency of minutes to an hour, and this is the red sliver drawn inside $\mathcal{I}_{14:00}$ in Figure 1.5.

Latency changes the problem rather than merely degrading it. A forecast nominally at $h = 5$ minutes built on data that is 12 minutes old is in truth a 17-minute-ahead forecast, and the persistence term of equation (1.8) is correspondingly weaker. The specification of a forecasting problem should therefore state the latency alongside the horizon, and a model should be trained on data lagged by the same δ it will face in production.

1.5 The forecast as a conditional statement

A forecast is a statement about a distribution, and a single number is one summary of it. This section contrasts two forecasts that share a mean, identifies the summary the decision actually needs, and adds the control action to the conditioning set.

1.5.1 A point forecast is a summary of a distribution

The statement $\hat{F} = 45\,\text{MW}$ is a *point forecast*: one number for the target. The statement $p(F \mid \mathcal{I}_t)$ is a *probabilistic forecast*: a distribution over the target. The first is recoverable from the second by choosing a summary — the mean, the median, or some other quantile — and the choice of summary is not innocuous, as Chapter 10 shows. The second is not recoverable from the first at all.

Consider two probabilistic forecasts of the same target,

$$\text{A: } F \sim \mathcal{N}(45,\, 2^2), \qquad \text{B: } F \sim \mathcal{N}(45,\, 15^2), \tag{1.13}$$

in MW. Both have $\mathbb{E}[F] = 45\,\text{MW}$, so both collapse to the same point forecast, and they describe very different situations. Forecast A says the fleet will deliver between about 41 and 49 MW; forecast B admits anything from 15 to 75 MW. An aggregator holding forecast A can make a firm commitment near 45 MW; one holding forecast B cannot.

1.5.2 The decision context selects the summary

Suppose the utility requires at least 40 MW of reduction and pays nothing, or imposes a penalty, if the fleet delivers less. The operationally relevant quantity is then not the expectation but the exceedance probability

$$\Pr(F \geq 40\,\text{MW} \mid \mathcal{I}_t), \tag{1.14}$$

which the two forecasts answer very differently. Under A the shortfall threshold is 2.5 standard deviations below the mean and $\Pr(F \geq 40) = 0.994$; under B it is one third of a standard deviation below and $\Pr(F \geq 40) = 0.631$. The same point forecast corresponds to a near-certainty and to a situation that fails on more than one occasion in three.

Figure 1.6 plots the two densities in separate panels, because their vertical scales differ by a factor of seven. In each panel the horizontal axis is delivered flexibility, the curve is the forecast density, the vertical line is the 40 MW requirement and the shaded area to its left is the shortfall probability. Those two areas are the complements of the two numbers above, 0.006

and 0.369; the first is too small to see at this scale, and the second occupies most of the left half of the lower panel. The figure is the first appearance of the theme that connects forecasting to operations: a decision needs a statement about a region of the distribution, usually a tail, and the mean is rarely that statement.

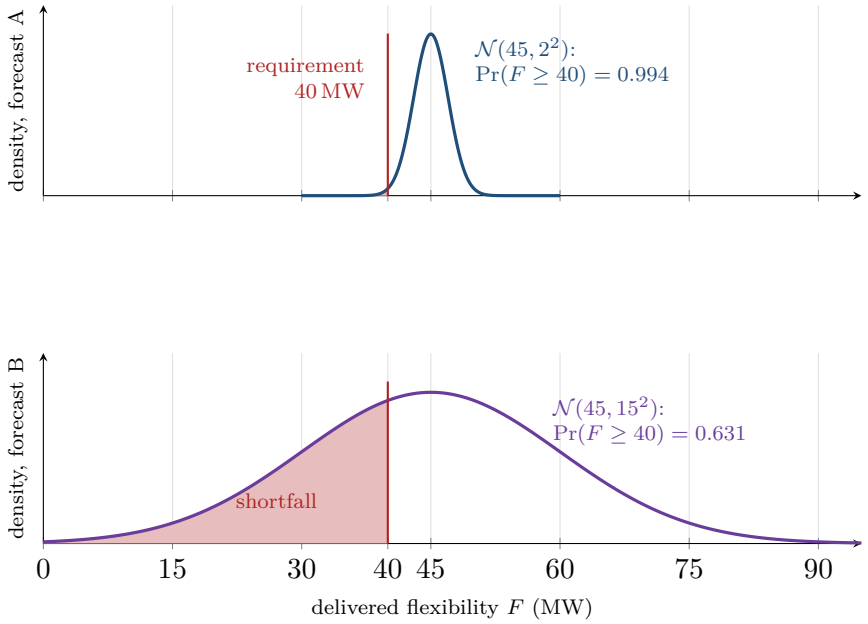


Figure 1.6: Two probabilistic forecasts with the same point forecast. Both densities are normal with mean 45 MW, as in equation (1.13); forecast A has $\sigma = 2$ MW and forecast B has $\sigma = 15$ MW. The vertical line is the 40 MW requirement and the shaded area is the shortfall probability, the complement of equation (1.14). The panels have separate vertical scales.

1.5.3 Conditioning on the planned control action

For the aggregator the conditioning set has one more element. Because two of the three targets depend on the intervention, the object to produce is

$$p(y_{t+h} \mid \mathcal{I}_t,\ u), \tag{1.15}$$

the distribution of the target given both the information and the planned action of equation (1.5). Reporting a single unconditional distribution for P^{ctrl} or F at a future hour is incomplete in the same way as reporting a point forecast for a wide distribution: it suppresses a dependence the user needs.

The addition of u is small in notation and large in consequence. A weather forecaster does not change tomorrow's weather, so the forecast is a statement about an exogenous process. An aggregator's flexibility forecast is a statement about a system its own organization is driving, and the forecast and the dispatch decision are parts of one problem. This is the point at which ordinary forecasting becomes forecasting for control, and Chapter 11 and the chapters beyond it return to it.

Principle 1.2. *For a controllable fleet the forecast is conditional on the planned action as well as on the information: the object is $p(y_{t+h} \mid \mathcal{I}_t, u)$. A forecast reported without the assumed action is not reproducible.*

1.6 The forecasting loop and a worked example

The pieces assembled above form an operating loop rather than a single calculation. This section draws the loop, works the chapter's numbers through it, and extracts the four questions that should be answered before any model is built.

1.6.1 From data to decision

A common mental model of forecasting is "historical load produces a forecast". Figure 1.7 replaces it with the structure this chapter has built. The boxes along the top row are the stages: the information set assembled at the issue time; the forecast model; the two target distributions $p(P^{\text{base}})$ and $p(P^{\text{ctrl}} \mid u)$; the flexibility distribution formed from them; and the optimization that consumes it. The inputs feeding the first box are listed beneath it, the planned action u enters the model in red because it is the aggregator's own choice, and the arrow returning from the decision to the fleet closes the loop: the dispatch changes the devices, which changes the next information set.

Two features of the diagram are worth naming. The forecast model produces distributions, not numbers, so the flexibility box is formed from a joint distribution rather than by subtracting two point values; Section 1.6.2 shows why that distinction has consequences. And the loop has a direction: information flows forward to a decision, and the decision flows back into the physical state. A forecasting system with no return arrow is a forecasting system nobody is acting on.

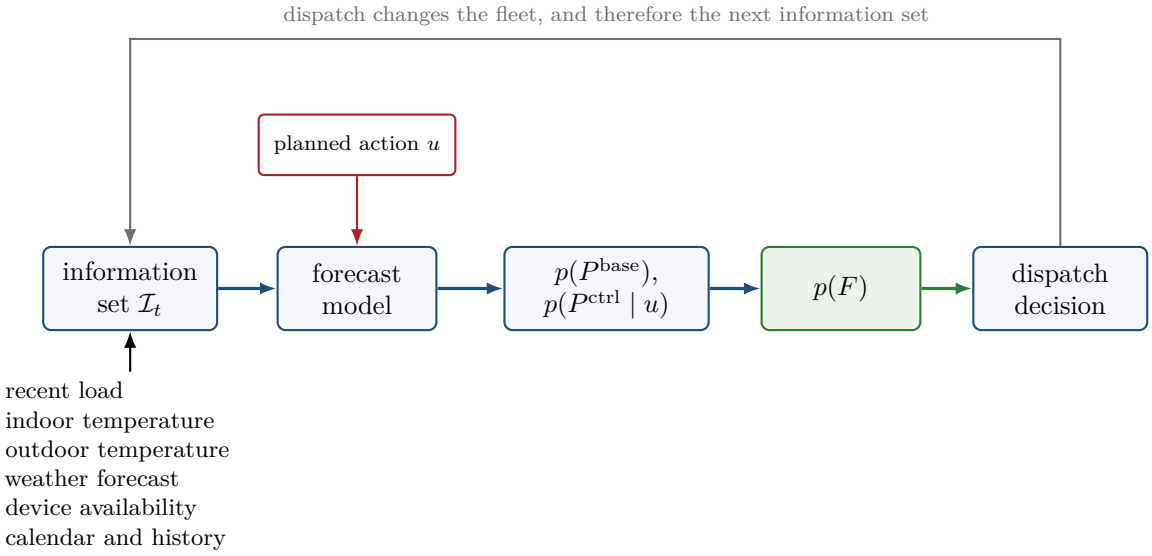


Figure 1.7: The forecasting loop for a controllable fleet. The stages run left to right from the information set to the dispatch decision; the inputs to the information set are listed below it, the aggregator's planned action enters the model from above, and the gray return path closes the loop through the physical fleet.

1.6.2 The chapter's numbers, worked through

Example 1.1 (The 2:00 PM question)**.** At 14:00 the aggregator holds the following information. The fleet is currently drawing $P_{\text{fleet}}(14{:}00) = 90\,\text{MW}$. The forecast outdoor temperature for 17:00 is $\hat{T}_{17:00|14:00} = 36°\text{C}$. Under comparable conditions the fleet's 5:00 PM load has historically been about 120 MW. The intended action is a $+2°\text{C}$ setpoint change, and the historical response of similar devices to such a change has ranged from 35 to 50 MW. A model built on this

information returns

$$P^{\text{base}}(17{:}00) \sim \mathcal{N}(120,\, 5^2), \qquad P^{\text{ctrl}}(17{:}00 \mid u) \sim \mathcal{N}(77,\, 7^2), \tag{1.16}$$

in MW, so that the expected flexibility is $\mathbb{E}[F] = 120 - 77 = 43\,\text{MW}$.

The mean of the difference is unambiguous. Its variance is not, and the naive calculation is wrong. Adding the variances, $5^2 + 7^2 = 74\ \text{MW}^2$, assumes that the two forecast errors are independent, and they are not: both P^{base} and P^{ctrl} are driven by the same forecast temperature, so a weather forecast that is two degrees too warm inflates both. In general

$$\text{Var}(F) = \sigma^2_{\text{base}} + \sigma^2_{\text{ctrl}} - 2\rho\,\sigma_{\text{base}}\sigma_{\text{ctrl}} = 74 - 70\rho \quad [\text{MW}^2], \tag{1.17}$$

with ρ the correlation between the two forecast errors. Three cases are instructive. Independence, $\rho = 0$, gives $\sigma_F = 8.6\,\text{MW}$. Strong common weather error, $\rho = 0.8$, gives $\sigma_F = 4.2\,\text{MW}$, less than half as wide. Perfect correlation, $\rho = 1$, gives $\sigma_F = |7 - 5| = 2.0\,\text{MW}$: the shared error cancels exactly in the difference and only the discrepancy between the two sensitivities survives.

Figure 1.8 shows the computation in two panels. The upper panel plots the two power densities of equation (1.16) against fleet power, with their means marked; the lower panel plots the resulting flexibility density under $\rho = 0$ and under $\rho = 0.8$ against flexibility, with the 40 MW requirement marked and the shortfall region of the wider case shaded. The two lower curves have the same mean, 43 MW, and the exceedance probability $\Pr(F \geq 40)$ is 0.64 in the independent case and 0.76 in the correlated case. The correlation assumption is therefore not a refinement; it moves the number the decision depends on. It reappears as a central issue in Chapter 11, where the errors being correlated belong to thousands of devices sharing one weather forecast.

1.6.3 Four questions before building a model

Everything in this chapter reduces to four questions, to be answered before a model class is considered.

1. What exactly is the target, including population, resolution, units and sign?

2. At what horizon, and from what issue time?

3. What exactly is known at that issue time, including latency and including which weather variables are forecasts rather than observations?

4. Which decision will consume the forecast, and therefore which feature of the distribution must be reported?

The questions are cheap and they prevent a specific class of failure: a system that is accurate against its own test set and useless in operation. The two most common instances have both appeared above. Question 3 catches the leakage of Section 1.4.3, in which a model was fitted to a variable that will not exist at issue time. Question 1 catches the mixture of Section 1.2.4, in which a target was assembled from events that are not comparable. Question 4 is the one most often skipped, and it is the reason a well-calibrated distribution is worth more than a slightly better mean.

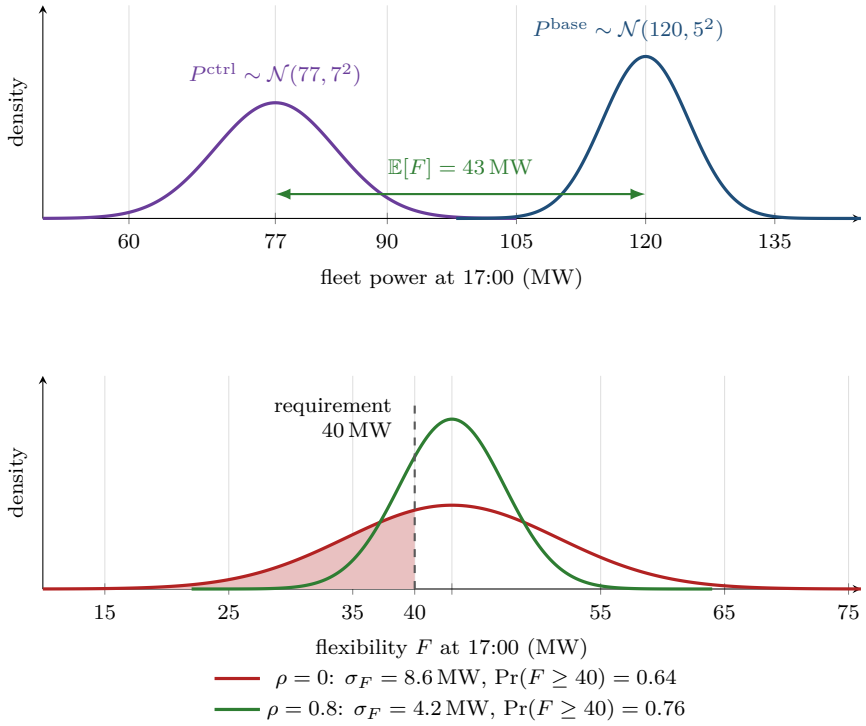


Figure 1.8: The worked example of Section 1.6.2. Upper panel: the two power densities of equation (1.16) against fleet power at 17:00, with the gap between their means labeled. Lower panel: the density of their difference under two correlation assumptions, from equation (1.17), with the 40 MW requirement dashed and the shortfall region of the independent case shaded. All densities are normal with the stated parameters.

Conceptual questions

"We want to forecast thermostat flexibility three hours ahead. How would you start?"

A strong answer does not begin with a model. It begins by defining the target: whether the quantity wanted is the baseline load, the controlled load under a specified dispatch action, or the difference between them, and it notes that the third is a counterfactual that will have to be estimated rather than measured. It then fixes the issue time and verifies that every intended input genuinely exists at that moment, separating observed weather from the forecast that will actually be available and accounting for telemetry latency. Next it establishes simple reference forecasts — persistence, a similar-day average, a temperature regression — so that any later model has something to beat. Only then does it compare more sophisticated models, and it reports the comparison at the horizon asked for rather than at a convenient one. A good answer also asks which decision the number serves, because a capacity commitment needs a lower quantile rather than a mean.

1.7 What to retain

Definition 1.1 (The forecasting problem)**.** The fundamental object of forecasting is not a value y_t but the conditional distribution $p(y_{t+h} \mid \mathcal{I}_t)$, in which t is the issue time, h the horizon, y_{t+h} the named target and $\mathcal{I}_t$ the information genuinely available at t. For a controllable fleet the target may depend on the operator's own action, so the object is $p(y_{t+h} \mid \mathcal{I}_t, u)$; that addition of u is what turns ordinary forecasting into forecasting for control.

Table 1.3 collects the chapter's objects with the subsection that defines each.

Table 1.3: The objects of a forecasting problem specification.

Object	Definition	Subsection
t, h	issue time and horizon; target time $t + h$	1.3.1
y_{t+h}	the target: a named quantity, population, resolution and sign	1.2.4
$\mathcal{I}_t$	information set: measured history, device state, exogenous forecasts, calendar	1.4.1
$\hat{y}_{t+h} = f(\mathcal{I}_t)$	point forecast: one summary of the distribution	1.5.1
$p(y_{t+h} \mid \mathcal{I}_t)$	probabilistic forecast: the complete object	1.1.3
$P^{\text{base}}(t)$	baseline load: consumption without intervention	1.2.1
$P^{\text{ctrl}}(t \mid u)$	controlled load under the action u of equation (1.5)	1.2.2
$F(t)$	flexibility $P^{\text{base}} - P^{\text{ctrl}}$; counterfactual, never measured	1.2.3
$\hat{T}_{t+h\mid t}$	weather forecast for $t + h$ issued at t; not the observed value	1.4.2
leakage	use of anything outside $\mathcal{I}_t$ in training or evaluation	1.4.3
δ	telemetry latency; the information set ends at $t - \delta$	1.4.4
$\Pr(F \geq F_0)$	the decision-relevant functional when a threshold must be met	1.5.2
$p(y_{t+h} \mid \mathcal{I}_t, u)$	the forecast for a controllable fleet	1.5.3

1.8 Exercises

Exercise 1.1. An operator asks three questions about the same thermostat fleet at 18:00 tomorrow: (a) "how much will these customers consume?"; (b) "if we raise setpoints by three degrees for ninety minutes, what will they consume?"; (c) "how much can we offer the utility?". Name which of P^{base}, $P^{\text{ctrl}}(\cdot \mid u)$ and F each question asks for, and state for each whether the quantity can be verified against a meter after the fact. For the one that cannot, say what must be supplied in its place.

Exercise 1.2. A load model fitted on observed temperatures reports a root-mean-square error of 2.5 MW. The fleet's sensitivity is $\partial P/\partial T = 4\,\text{MW}$ per degree Celsius and the three-hour-ahead temperature forecast has $\sigma_T = 1.5°\text{C}$. Using equation (1.10), compute the operational error at $h = 3$ hours. Repeat for a day-ahead forecast with $\sigma_T = 2.5°\text{C}$. In which of the two cases is the load model itself the limiting factor?

Exercise 1.3. For the two forecasts of equation (1.13), compute the largest reduction the aggregator could commit to while keeping $\Pr(F \geq \text{commitment}) \geq 0.95$. State both answers, compare them with the shared point forecast of 45 MW, and say what the difference is worth if committed capacity is paid \$40 per MW.

Exercise 1.4. Using equation (1.17), find the correlation ρ for which $\sigma_F = 3\,\text{MW}$, and compute $\Pr(F \geq 40\,\text{MW})$ at that value. Then explain, in terms of the shared weather forecast, why a *positive* correlation between the two forecast errors makes the flexibility forecast narrower rather than wider, and what would have to be true for ρ to be negative.

Chapter 2

Structure of energy time series: level, seasonality, weather dependence and persistence

2.1 The decomposition of a load series

Chapter 1 fixed the forecasting problem: at issue time t, using only the information set $\mathcal{I}_t$, produce $p(y_{t+h} \mid \mathcal{I}_t)$ for a target h hours ahead. Nothing in that statement promises the target is predictable. Whether it is, and by how much, depends on the structure of the series, and this chapter's question is

What is an hourly energy series made of, and which of its parts can be predicted?

A fleet load series is not an arbitrary sequence of numbers. It is the measured output of a large physical system whose inputs are weather, household schedules, device states and, for an aggregator, its own control commands. Each of those inputs leaves a recognizable signature in the series: a level, repeating daily and weekly cycles, a kinked dependence on outdoor temperature, and a strong dependence of each hour on the hours before it. Those signatures are what every forecasting method in this book exploits, and naming them precisely is a prerequisite for choosing a method at all.

Jargon: level, trend, seasonality, autocorrelation, stationarity, balance temperature

The *level* of a series is its local average, the number the series fluctuates about. A *trend* is a slow change in that level over months or years. *Seasonality* is any cycle of fixed period: electricity load has a 24-hour cycle, a 168-hour (weekly) cycle and an annual cycle, so "seasonal" in this book means "periodic", not "to do with summer". *Autocorrelation* is the correlation of a series with its own past, $\mathrm{Corr}(P_t, P_{t-k})$ at lag k. A series is *stationary* when its statistical properties do not depend on the time of observation; load is not stationary, and much of forecasting practice consists of removing the parts that make it nonstationary. An *exogenous variable* is a predictor that comes from outside the series, principally weather and calendar. The *balance temperature* (also called the balance point or base temperature) is the outdoor temperature at which a building needs neither heating nor cooling; below and above it the load responds to weather, between them it does not. *Common-mode uncertainty* is error shared by every device in a fleet, such as a single wrong temperature forecast for the whole territory; averaging over devices does not reduce it.

Key idea

An energy series is the output of a physical system driven by weather, by calendar-locked human behavior and by control. Its level, cycles, weather response and persistence are signatures of those mechanisms rather than statistical curiosities, and that physical origin is why the series is forecastable at all.

This section writes the series as a sum of parts, restates the same object in state-space form so that the persistence of Section 2.4 has somewhere to live, and decomposes one synthetic week of fleet load into its four components.

2.1.1 From causes to components

Write the fleet load at hour t as a function of its causes,

$$P_t = f(x_t,\ w_t,\ b_t,\ c_t,\ u_t) + \varepsilon_t, \tag{2.1}$$

with x_t the physical state of the fleet (indoor temperatures, stored heat in the building fabric, which compressors are running, how many devices are connected), w_t the weather, b_t occupant behavior, c_t the calendar, u_t the aggregator's control action, and ε_t everything left over. Equation (2.1) is not a model that can be fitted; it is a statement about where the numbers come from, and it already contains the chapter's agenda, because each argument produces a different kind of structure.

For working purposes the function is usually replaced by a sum of components,

$$P_t = \underbrace{L_t}_{\text{level and trend}} + \underbrace{S_t}_{\text{calendar}} + \underbrace{W_t}_{\text{weather response}} + \underbrace{C_t}_{\text{control}} + \underbrace{\varepsilon_t}_{\text{residual}}, \tag{2.2}$$

which is the classical additive decomposition with one term added for the aggregator's own intervention. Three properties of equation (2.2) should be stated at once, because all three matter later.

The decomposition is a modeling choice, not a property of the data. Load could equally be written multiplicatively, $P_t = L_t S_t W_t \varepsilon_t$, which is often a better description of long-horizon growth, since a fleet twice the size has roughly twice the daily swing as well as twice the level. The additive form is used here because the components can then be plotted on the same axis and read in megawatts.

The components are not orthogonal. Nothing forces S_t and W_t to be uncorrelated, and in practice they are strongly correlated: afternoons are both the hot part of the day and the part of the day when the hour-of-day profile is high. The consequence is that the variances of the components do not add up to the variance of the series, and that "how much of the variance is weather" has no unique answer. A number for it is only meaningful once the decomposition that produced it is stated.

The residual is defined by what the other terms contain. ε_t is not a property of the physical world. It is whatever the other four terms failed to represent, so it shrinks as the model improves and, importantly, it is usually not white noise. The residual of a calendar-and-weather model retains autocorrelation, which is exactly the structure Chapter 6 models.

2.1.2 The state-space viewpoint

Equation (2.2) has no memory in it: every term is a function of the current hour. That cannot be right for a fleet of buildings, because a building that was hot an hour ago is still warm now. The compact way to carry memory is the state-space pair

$$x_{t+1} = g(x_t,\ u_t,\ w_t), \qquad P_t = h(x_t) + \varepsilon_t, \tag{2.3}$$

in which g is the dynamics, h is the measurement, and the state x_t is the only channel through which the past reaches the present. For a thermostat fleet the state includes quantities a forecaster never measures directly: the distribution of indoor temperatures across the fleet, the heat stored in walls and floors, the on/off phase of each compressor, and the number of devices currently reachable. Figure 2.1 draws this structure.

The figure is read left to right. The four boxes on the left are the exogenous drivers of equation (2.1): weather, calendar, occupant behavior, and the control action, drawn in red because it is the only input the aggregator chooses. They feed the large central box, the latent state x_t, whose contents are listed inside it. The curved arrow above that box is the dynamics g: the state at the next hour depends on the state at this one, which is where the series' memory physically resides. The arrow on the right is the measurement h, which maps state to metered power, with the noise ε_t entering at the meter. The dashed path along the bottom is the part of household load that reaches the meter without passing through the thermal state at all: lighting, cooking, consumer electronics. That path has calendar structure but almost no thermal memory, which is why a fleet series shows both fast and slow dependence on its own past.

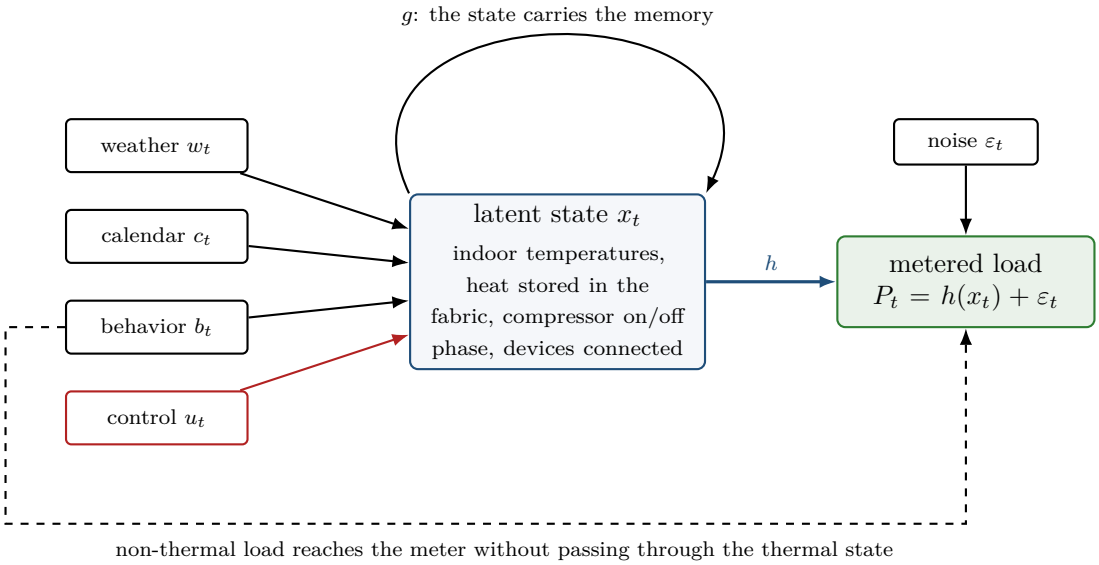


Figure 2.1: The generative structure of a fleet load series. Exogenous drivers (left) act on a latent physical state (center) whose own dynamics g carry memory from hour to hour; the measurement h turns state into metered power. The red input is the only one the aggregator chooses. The dashed path is non-thermal household load, which has calendar structure but no thermal memory.

The state-space form also explains a fact that would otherwise look like a coincidence: recent observations of load are useful out of proportion to their apparent information content. A single number, P_t, is useful because it is a measurement of $h(x_t)$, and therefore evidence about the hidden state that will generate the next several hours. Persistence, in other words, is a crude state estimator. That reading is developed in Section 2.4 and is the reason Chapter 7 can replace hand-built lag features with a learned hidden state.

2.1.3 A week of fleet load, decomposed

Figure 2.2 decomposes one week of synthetic hourly fleet load into exactly the terms of equation (2.2), with the control term set to zero because no event is dispatched. The series is generated by the script named in the caption and is illustrative, not measured; it is used here so that every component can be shown exactly rather than estimated.

The figure has four stacked panels sharing one horizontal axis, the hour of the week, with hour 0 at Monday 00:00 and the ticks at the start of each day. The top panel shows the realized load in gray and the sum of the level and the calendar component, $L_t + S_t$, as a blue dashed curve; the shaded band on the right is the weekend. The second panel is the outdoor temperature, with a horizontal line at the balance temperature of 22°C used in the construction. The third panel is the weather component W_t, and the bottom panel is the residual ε_t.

Four features of the picture carry the chapter. First, the load oscillates between 69.5 and 127.5 MW about a mean of 96.5 MW, and almost all of that oscillation repeats daily: the dashed curve in the top panel, which contains no weather information at all, already follows the shape of the series. Second, the daily mean rises through the working week, 94.5, 97.3, 98.5, 100.3 and 102.3 MW from Monday to Friday, not because of any calendar effect but because the synthetic temperature rises; it then falls to 92.5 and 90.1 MW at the weekend, where the calendar term is reduced. Third, the weather component peaks at 20.4 MW and, in the third panel, peaks about two hours later than the temperature in the second panel, a lag that Section 2.4.4 attributes to thermal mass. Fourth, the residual in the bottom panel has a standard deviation of only 2.3 MW but is visibly smooth rather than jagged: its lag-one correlation is 0.65, so even this deliberately simple residual is autocorrelated.

The non-orthogonality warned about in Section 2.1.1 is visible in this example as arithmetic. The component variances are 169.8, 32.9 and 5.1 MW^2 for calendar, weather and residual, summing to 207.8, whereas the variance of the load itself is 296.5 MW^2. The missing 88.7 MW^2 is twice the covariance between the calendar and weather components, whose correlation is 0.59 because both peak in the afternoon. A statement such as “weather explains 11 % of the variance”, obtained by dividing 32.9 by 296.5, is therefore an artifact of the split and not a physical fact.

2.1.4 The mechanisms and the structures they create

Table 2.1 lists the mechanisms of equation (2.1) against the structure each leaves in the series and the section of this chapter that treats it. The table is the chapter’s plan. Reading it downward also gives a ranking by magnitude for a typical residential fleet: the calendar and weather terms are the large, highly predictable parts; thermal memory and device state are smaller but are what make short-horizon forecasting work; and the last two rows, common-mode uncertainty and the aggregator’s own control, are the parts that an ordinary load forecaster can ignore and an aggregator cannot.

2.2 Calendar structure

The calendar term S_t of equation (2.2) is the largest and most reliably predictable part of a load series, because it is produced by habits rather than by physics. This section separates it into the daily, weekly and annual cycles, treats the irregular calendar (holidays and the days around them), and explains why a thermostat fleet carries two calendar signals rather than one.

2.2.1 The daily cycle and the hour-of-day profile

The dominant period in an hourly load series is 24 hours. People wake, leave, return and sleep on a schedule locked to the clock, and the devices they operate follow. The standard summary

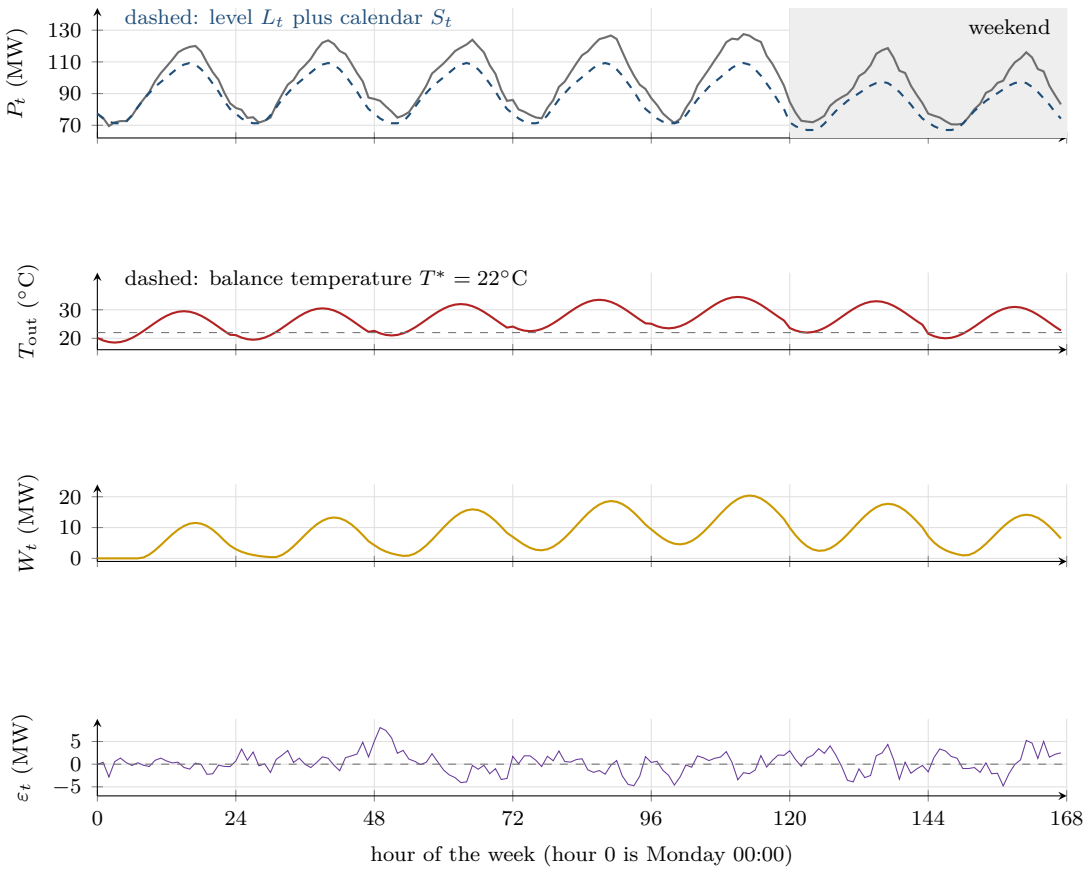


Figure 2.2: One synthetic week of hourly fleet load decomposed into the terms of equation (2.2). Panels from the top: load with the level-plus-calendar sum; outdoor temperature with the balance temperature; the weather component; the residual. The series is built by `scripts/ch02_data.py` as $P_t = 90\,\mathrm{MW} + S_t + 1.8\,\mathrm{CD}_t^{\mathrm{eff}} + \varepsilon_t$, where S_t is a fixed mean-zero hour-of-day profile scaled by 0.80 and shifted by $-8\,\mathrm{MW}$ at the weekend, $\mathrm{CD}_t^{\mathrm{eff}}$ is a three-hour exponential moving average of $\max(T_{\mathrm{out}} - 22°\mathrm{C}, 0)$, and ε_t is an AR(1) process with $\phi = 0.6$ and innovation standard deviation 1.6 MW. The numbers are illustrative.

Table 2.1: The mechanisms behind a fleet load series and the structure each one creates.

Mechanism	Structure it creates in P_t	Section
Household schedules and occupancy	daily and weekly cycles, holiday effects, a weekday/weekend level shift	2.2
Space-conditioning physics	dependence on outdoor temperature, with a flat band and a kink; saturation in extreme heat	2.3
Building thermal mass	thermal memory: this hour's load depends on the last several hours of weather, not only on the present hour	2.4.4
Device cycling and device state	short-lag autocorrelation; dependence of the response on how the fleet was driven recently	2.4.1, 2.5.1
Fleet composition and growth	the level, a trend, and nonstationarity of the whole series	2.6.1
One weather field over many devices	common-mode fleet uncertainty that aggregation does not reduce	2.5.3
The aggregator's own control	structural breaks, synchronization and rebound; a recorded series that is not natural load	2.6.2
Everything not represented	the residual, which is itself autocorrelated	2.1.1

is the *hour-of-day profile*, the conditional mean of load given the hour and the day type,

$$\mu(k, j) = \mathbb{E}\big[P_t \mid \text{hour}(t) = k,\ \text{daytype}(t) = j\big], \qquad k = 0, \ldots, 23, \tag{2.4}$$

estimated by averaging the load of every hour k of every day of type j in the training record. Equation (2.4) is both a description and a forecast: the similar-day baselines of Chapter 3 are essentially this table, and the calendar features of Chapter 4 are a way of giving a regression enough information to reconstruct it.

Figure 2.3 plots the profile used in the synthetic week. The horizontal axis is the hour of the day from 0 to 23 and the vertical axis is load in megawatts; the blue curve is the weekday profile and the purple curve is the weekend profile. Three features are worth naming. The minimum is in the early morning, hours 3 and 4, when occupants are asleep and only refrigeration, standby and some space conditioning remain. The maximum is in the late afternoon, hour 16, by which time the accumulated heat of the day and the return of occupants coincide. And the curve is smooth rather than square: the fleet contains many households whose schedules differ by an hour or two, so the aggregate transition is gradual even though each household's is abrupt. That smoothing is the same effect as the device-count smoothing of Section 2.5.2, applied to schedules instead of compressors.

The practical consequence of the daily cycle is that the lag-24 value of a load series is a strong predictor, and the seasonal-persistence baseline of Chapter 3, "the same hour yesterday", is hard to beat precisely because it carries the whole profile without estimating anything.

2.2.2 The weekly cycle and day types

Superimposed on the daily cycle is a 168-hour cycle. In the synthetic week it is imposed as a -8 MW level shift and a reduction of the profile amplitude to 0.80 on Saturday and Sunday, which is the qualitative pattern of a residential fleet: weekend load is lower in total and flatter in shape, because the sharp morning departure and evening return of the working week are absent and occupancy is spread through the day. The two curves of Figure 2.3 show the effect: the weekend curve lies below the weekday curve at every hour and its peak-to-trough range is smaller.

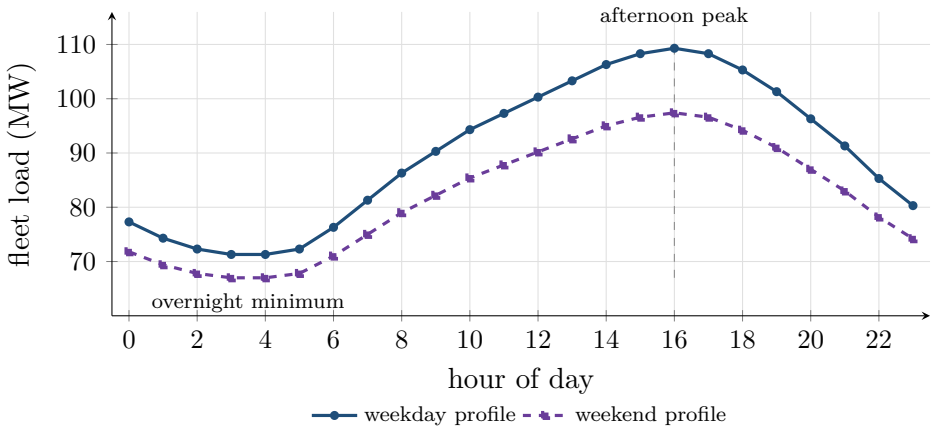


Figure 2.3: Hour-of-day profiles for the synthetic fleet of Figure 2.2. The weekday curve is 90 MW plus the mean-zero profile used in the construction; the weekend curve is 82 MW plus 0.80 times the same profile, which is the day-type effect of Section 2.2.2. Values are illustrative.

The weekly structure is usually best represented not as a sinusoid of period 168 hours but as a small set of *day types*. For a residential fleet, three or four types suffice: Monday to Thursday, Friday, Saturday, and Sunday, with Monday sometimes separated because it follows two low days. For a commercial or industrial fleet the split is different, since a single large customer's shift pattern can dominate. The choice of day types is a modeling decision with a direct cost: too few types, and the model averages genuinely different days together; too many, and each type is estimated from fewer observations.

2.2.3 The annual cycle and the two-regime year

The third period is the year. Its mechanism is mostly weather, and therefore the annual cycle is largely a consequence of Section 2.3 rather than an independent effect: the seasonal swing in load follows the seasonal swing in temperature. Two parts of the annual cycle, however, are genuinely calendar rather than weather. Daylight hours set lighting load and shift the evening peak earlier or later in a way that no temperature variable captures. And the calendar itself moves relative to the week, so a fixed-date holiday falls on different weekdays in different years.

For a cooling-dominated fleet the annual structure is better described as two regimes than as one cycle. In the cooling season the fleet has large weather-driven load and large flexibility; in the shoulder season it has neither; and the transition between them takes a few weeks. A single model fitted across the whole year must either contain the regime as a feature or be fitted separately per season. This matters for validation: a model trained on cooling-season data and tested on shoulder-season data will appear to fail at something it was never asked to do, a trap Chapter 9 returns to.

2.2.4 Holidays and the irregular calendar

Holidays are the part of the calendar that regression models handle worst, for three reasons. They are rare, so each holiday type has few historical examples. They are not interchangeable: a mid-summer public holiday resembles a Sunday in shape but occurs in hot weather, while a winter holiday may resemble nothing else in the record. And their effect spills over, so the day before and the day after a long weekend are also atypical.

The standard treatments are a holiday indicator feature, a mapping of each holiday onto the day type it most resembles, and in the extreme a separate model. There is also a class of calendar-like irregularities that are not holidays at all: school terms, large local events, and

anything that moves many people at once. For an aggregator one of these is special, because the aggregator causes it. A demand-response event makes a day atypical in exactly the way a holiday does, and it is recorded in the same series; Section 2.6.2 treats it separately for that reason.

2.2.5 Two calendar signals in one fleet

A thermostat fleet carries calendar structure through two distinct channels, and distinguishing them explains the shape of Figure 2.3.

The first channel is behavioral. Occupants change setpoints, arrive and depart, open doors, cook and run appliances. This channel acts on the clock, is sharp at the household level, and produces the morning and evening features of the profile. It is also the channel that differs most between weekdays and weekends.

The second channel is physical, and it is calendar-locked only because the sun is. Solar gain through windows and roofs follows the solar day, the building fabric stores that heat and releases it over hours, and the resulting cooling demand peaks in the late afternoon regardless of anyone's schedule. This channel is smooth, lagged, and nearly identical on a Saturday and a Tuesday with the same weather.

The two channels respond differently to everything an aggregator might do. A setpoint change acts on the physical channel and leaves the behavioral one untouched, so the non-thermal load, which is the dashed path of Figure 2.1, sets a floor under what any curtailment can achieve. Separating them is therefore not an academic exercise: the quantity the aggregator can sell is the controllable fraction, not the metered load.

> **Principle 2.1.** *Calendar structure in a thermostat fleet arrives through two channels, a sharp behavioral one locked to the clock and a smooth physical one locked to the sun. Only the second is controllable, so only the second contributes to flexibility.*

2.3 Weather dependence

Weather is the largest exogenous driver of a residential fleet's load and the one whose functional form matters most, because the relationship is neither linear nor monotonic. This section establishes the balance temperature and the piecewise response, defines the degree variables that linearize it, treats saturation at extreme temperature, and lists the weather channels beyond temperature.

2.3.1 The balance temperature and the piecewise response

A building exchanges heat with the outdoors at a rate roughly proportional to the temperature difference across its envelope, and its occupants run heating or cooling only when the resulting indoor temperature would leave the comfort band. The consequence for load is a three-part relationship. Below a lower balance temperature, heating runs and load rises as the outdoor temperature falls. Between the two balance temperatures, no space conditioning runs and load is flat in temperature. Above an upper balance temperature, cooling runs and load rises as the outdoor temperature rises.

Figure 2.4 draws this curve. The horizontal axis is outdoor temperature from 0 to 40°C and the vertical axis is fleet load in megawatts. The shaded region between 15°C and 22°C is the balance band, where the curve is flat at 86 MW: that value is the non-weather-dependent load, lighting, appliances, water heating and standby. To the left the heating branch rises with slope 1.2 MW per degree; to the right the cooling branch rises with initial slope 1.8 MW per degree

and then bends over. The dashed straight line is the tangent to the cooling branch at 22°C, drawn to make the bend visible.

The shape is why the relationship is often called a hockey stick, or a V, or a U depending on the fleet's heating technology. For a fleet with gas heating the left branch is nearly absent and the curve is a hockey stick; for an all-electric fleet it is a V with two steep arms. The practical point is the same in both cases: a model given raw temperature and asked to fit a straight line will produce a line that is wrong in all three regions at once, overpredicting in the balance band and underpredicting at both extremes.

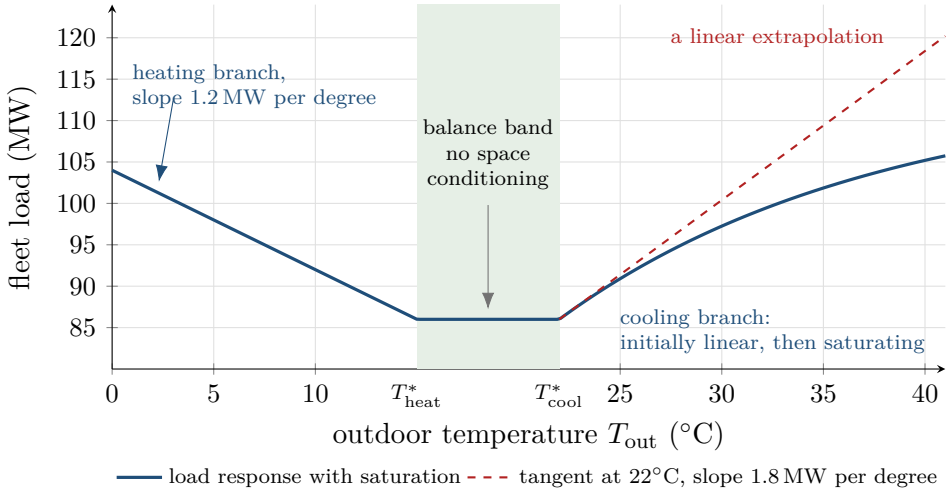


Figure 2.4: The piecewise load-temperature relationship. The curve is $86 + 1.2\max(15 - T_{\text{out}}, 0) + 28\big[1 - \exp(-\max(T_{\text{out}} - 22, 0)/15.556)\big]$ MW, so that the cooling branch has initial slope 1.8 MW per degree and the saturating asymptote of equation (2.7) is 28 MW above the balance load. The shaded band is the balance band. Values are illustrative.

2.3.2 Cooling and heating degrees

The remedy is to supply the kink as part of the predictor rather than asking the model to find it. Define the cooling and heating degrees about their balance temperatures,

$$\text{CD}_t = \max(T_{\text{out}} - T^*_{\text{cool}},\ 0), \qquad \text{HD}_t = \max(T^*_{\text{heat}} - T_{\text{out}},\ 0), \tag{2.5}$$

both nonnegative and both zero inside the balance band. The weather component of equation (2.2) then becomes linear in these variables,

$$W_t = \beta_c\,\text{CD}_t + \beta_h\,\text{HD}_t, \tag{2.6}$$

with β_c and β_h the cooling and heating sensitivities in megawatts per degree. Equation (2.6) is a linear model in $(\text{CD}_t, \text{HD}_t)$ and a nonlinear model in T_{out}, which is the whole point: the nonlinearity has been moved into the definition of the feature, where it can be stated from physics instead of estimated. The synthetic week of Figure 2.2 uses exactly equation (2.6) with $\beta_c = 1.8$ MW per degree and no heating branch.

Two details decide whether the construction works. The balance temperatures are properties of the building stock and the local comfort convention, not universal constants; 18°C is the traditional value for heating degree days and is often wrong for a modern air-conditioned fleet, where a cooling balance point between 20 and 24°C is more usual. And because CD_t and HD_t are products of a threshold, they are only as good as the threshold: fitting the balance temperature as an extra parameter, or giving the model several degree variables at different

thresholds, both work better than guessing once. Chapter 4 builds these variables as features, and Chapter 5 shows that a tree ensemble can find the kink without being told, which is one of the main reasons to prefer one.

Principle 2.2. *The load-temperature relationship has a flat band and a kink. A linear model must be given the kink, through degree variables about a balance temperature; a model given raw temperature instead is wrong in every region of the curve.*

2.3.3 Saturation at high temperature

Equation (2.6) cannot hold indefinitely, because the fleet has finite installed cooling capacity. As the outdoor temperature rises, compressor duty cycles rise toward one; once essentially every compressor runs continuously, further heat cannot increase the fleet's electrical load, and the load-temperature curve flattens. A saturating form,

$$W_t = A\left[1 - \exp\!\left(-\frac{\mathrm{CD}_t}{\kappa}\right)\right], \qquad \left.\frac{\partial W_t}{\partial \mathrm{CD}_t}\right|_{\mathrm{CD}_t=0} = \frac{A}{\kappa}, \tag{2.7}$$

captures both ends: the slope near the balance point is A/κ, which is the β_c of equation (2.6), and the asymptote is A, the fleet's controllable cooling capacity. The curve of Figure 2.4 uses $A = 28\,\mathrm{MW}$ and $\kappa = 15.556°\mathrm{C}$, so $A/\kappa = 1.8\,\mathrm{MW}$ per degree. At 40°C the linear extrapolation predicts 118.4 MW and the saturating form predicts 105.2 MW, a difference of 13 MW at the hour when the error matters most.

Saturation also has a consequence that belongs to this book rather than to load forecasting generally. The duty cycle D_t is the fraction of the fleet's compressors running, and the flexibility available from curtailment is roughly proportional to it. Saturation means $D_t \to 1$, so load is at a maximum and the fleet's *reserve* is at a minimum: there is nothing left to turn on, and turning units off produces a response that decays quickly because the buildings heat up fast. The hottest hour therefore has the largest load, the largest instantaneous flexibility, and the shortest sustainable flexibility, and a model that extrapolates linearly gets all three wrong in the same direction.

2.3.4 Humidity, irradiance and the other channels

Temperature is the dominant weather variable but not the only one. Four others carry useful information for a cooling-dominated fleet.

Humidity. An air conditioner removes both sensible and latent heat, and removing moisture costs energy without lowering the thermostat's reading. At 34°C and 30 % relative humidity a unit is doing much less work than at 34°C and 70 %. The usual treatments are to include RH_t directly, to include an interaction term $\mathrm{CD}_t \times \mathrm{RH}_t$, or to replace the dry-bulb temperature with a combined index such as a wet-bulb or apparent temperature. The interaction form is often the most informative, because humidity matters only when cooling is running.

Solar irradiance. Solar gain through glazing and roofs is a heat input that the outdoor temperature does not record. It is also the mechanism behind the physical calendar channel of Section 2.2.5, and it is partly collinear with the hour of the day, which makes its coefficient hard to identify from load data alone.

Wind. Wind increases the envelope's effective heat-transfer coefficient, raising both heating and cooling demand slightly. Its effect is small for a well-built residential stock and not usually worth a feature.

Yesterday's weather. Two days with the same temperature at 17:00 do not have the same load at 17:00 if one of them followed a cool night and the other followed a hot one, because the building fabric carries heat across hours. This is the thermal memory of Section 2.4.4, and it enters the weather block as lagged and rolling weather variables rather than as a new measurement channel.

2.3.5 Weather enters the model as a forecast

One property of the weather block has nothing to do with physics and everything to do with the information set. At issue time t, a forecast for $t + h$ cannot use T_{out} at $t + h$, because that temperature has not happened; it can use only $\hat{T}_{t+h|t}$, the weather forecast available at t. Chapter 1 established the rule and named the failure mode, which is leakage.

The consequence for this chapter is that the strength of the load-temperature relationship seen in historical data is an upper bound on what a forecast can exploit. In the synthetic week the correlation between load and outdoor temperature is 0.92 and the correlation between load and the lagged cooling-degree component is 0.87; both are computed from the temperature that actually occurred. Replacing that temperature with a day-ahead forecast carrying a root-mean-square error of a degree or two moves a sizeable part of the weather component into the forecast error, and the size of that movement is β_c times the weather forecast error. With $\beta_c = 1.8\,\text{MW}$ per degree, a 1.5°C weather error is a 2.7 MW load error that no amount of model improvement can remove. The same arithmetic reappears in Chapter 11 as the dominant common-mode term.

2.4 Persistence, thermal memory and the autocorrelation function

The remaining structure in a load series is its dependence on its own past. This section measures that dependence with the autocorrelation function, identifies the lags that carry information, and then explains where the dependence comes from physically, which is the chapter's main conceptual point.

Jargon: lag, autocorrelation function, partial autocorrelation function

The *lag* k is a displacement in time measured in sampling intervals, so for hourly data lag 24 is one day earlier. The *autocorrelation function* ACF(k) is the correlation between the series and itself k steps earlier. The *partial autocorrelation function* PACF(k) is the correlation remaining at lag k once the effect of all shorter lags has been removed, that is, the coefficient on P_{t-k} in a regression of P_t on $P_{t-1}, \ldots, P_{t-k}$. This chapter only reads these two functions as diagnostics; Chapter 6 uses them to specify models.

2.4.1 Lag-one persistence

For hourly load, ACF(1) is typically between 0.95 and 0.99. The mechanism is direct: an hour is short compared with every time constant in the system. The same houses are occupied, the same appliances are running, the outdoor temperature has moved by a degree at most, and the buildings' stored heat has barely changed. The physical state x_t of equation (2.3) is almost the same state one hour later, and since $P_t = h(x_t) + \varepsilon_t$, the loads are almost the same too.

This is why the persistence forecast of Chapter 3, $\hat{y}_{t+h} = P_t$, is a serious competitor at short horizons and why Chapter 1 insisted that the horizon changes the problem. At $h = 5$ minutes, persistence is close to optimal because the state has not moved. At $h = 3$ hours it is poor, because the state has moved along the daily cycle; by then the right comparison is the same hour yesterday, which is a statement about ACF(24) rather than ACF(1). The horizon

at which the crossover happens is a property of the series, readable from the autocorrelation function, and it is different for a single household than for a fleet.

2.4.2 The autocorrelation function of hourly load

Define the autocorrelation at lag k in the usual way,

$$\mathrm{ACF}(k) = \mathrm{Corr}(P_t,\ P_{t-k}) = \frac{\mathrm{Cov}(P_t, P_{t-k})}{\sqrt{\mathrm{Var}(P_t)\ \mathrm{Var}(P_{t-k})}}, \tag{2.8}$$

estimated from a long record by the sample analog. Plotting equation (2.8) against k is the single most informative first look at a new energy series.

The upper panel of Figure 2.5 shows what that plot looks like for hourly load. The horizontal axis is the lag in hours from 0 to 168, the vertical axis is the correlation, and each lag is drawn as a vertical stem from zero; the dashed line marks zero. Four features should be read off it in order. The function starts at 1 by construction and falls only slowly over the first few hours, which is the lag-one persistence of Section 2.4.1. It then turns negative around lag 12, because half a day earlier the daily cycle was in its opposite phase: the hot afternoon is compared with the cool night. It returns to a local maximum at lag 24 and at every multiple of 24 thereafter, the signature of the daily cycle. And the heights of those daily peaks do not decay monotonically: they fall to a minimum around lag 96 and rise again toward lag 168, because one week earlier is not merely four days further away but the same day of the week, with the same day type.

Two cautions apply to reading such a plot. The first is that the autocorrelation of a seasonal series says nothing about causation: ACF(24) is large because both hours share a cause, the hour of the day, not because yesterday's load influences today's. The second is that a large ACF(k) does not imply that lag k is a useful *feature* once other features are present; that question is the one the partial autocorrelation answers.

2.4.3 Partial autocorrelation and where it cuts off

The autocorrelation at lag 3 is large mostly because lag 3 is correlated with lag 2, which is correlated with lag 1. The partial autocorrelation strips that chain out: PACF(k) is the extra correlation at lag k that the shorter lags do not already account for.

The lower panel of Figure 2.5 plots an illustrative PACF(k) for lags 1 to 30, on the same stem convention as the upper panel but over a shorter range of lags. It is large at lag 1, noticeably smaller at lag 2, essentially zero from lag 3 to lag 22, and then spikes again at lag 24. The reading is that the direct dependence of load on its own past is short, a few hours, plus a separate direct dependence on the same hour yesterday. A measured series would show a further spike at lag 168, outside the plotted range, for the same reason.

This pattern is what makes a small feature set work. A model given P_{t-1}, P_{t-2}, P_{t-24} and P_{t-168} has most of the exploitable dependence, which is exactly the lag set Chapter 4 builds and is not a coincidence: lag selection follows the partial autocorrelation. The qualifier is that equation (2.8) and its partial counterpart are linear measures of dependence. They cannot see the nonlinear dependence that the saturation of Section 2.3.3 produces, nor dependence that appears only in certain regimes, so an empty partial autocorrelation is evidence of no *linear* structure left, not of no structure left.

2.4.4 Thermal memory and the RC time constant

The physical origin of the short-lag dependence is thermal mass. Treat a building as a single thermal capacitance C behind a single envelope resistance R, driven by outdoor temperature

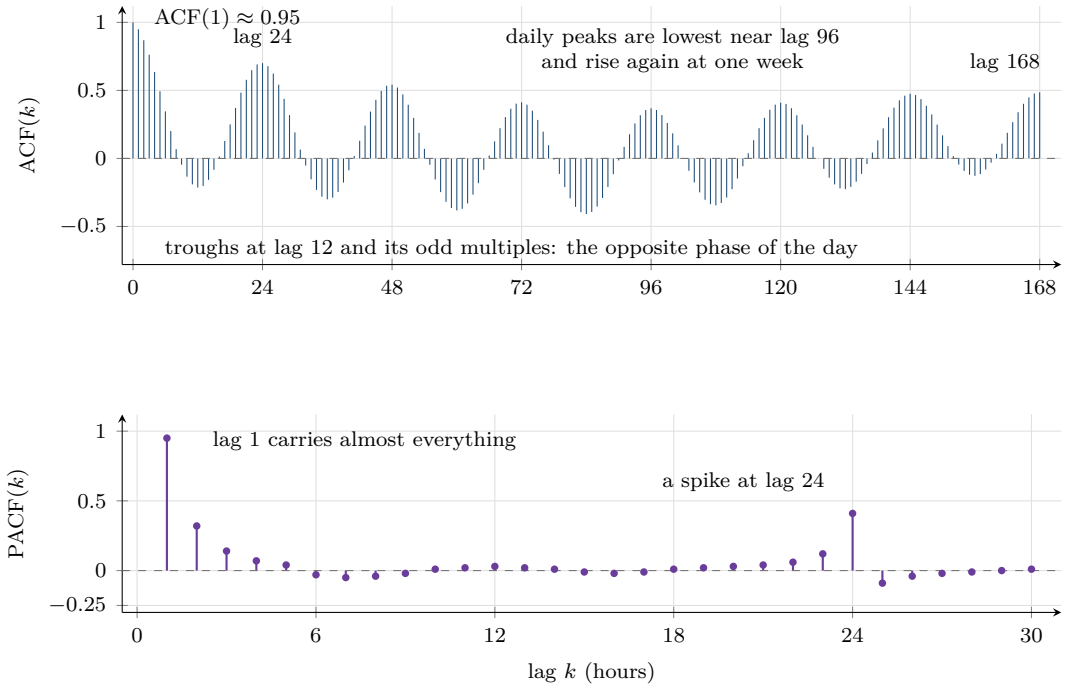


Figure 2.5: Autocorrelation and partial autocorrelation of an hourly load series. Upper panel: ACF(k) drawn as the illustrative analytic form $0.20e^{-k/6}+0.16e^{-k/200}+0.52e^{-k/300}\cos(2\pi k/24)+0.12\cos(2\pi k/168)$, which reproduces the qualitative features of a measured load autocorrelation: slow initial decay, a negative trough near lag 12, peaks at every multiple of 24 and a weekly revival at lag 168. Lower panel: an illustrative PACF(k) for lags 1 to 30, large at lag 1, small from lags 3 to 23 and spiking again at lag 24.

and cooled by an air conditioner delivering heat removal $\dot{Q}_t$. The indoor temperature then obeys

$$\frac{\mathrm{d}T_{\mathrm{in}}}{\mathrm{d}t} = \frac{T_{\mathrm{out}} - T_{\mathrm{in}}}{RC} - \frac{\dot{Q}_t}{C}, \qquad \tau = RC, \tag{2.9}$$

a first-order linear system whose time constant $\tau = RC$ is a few hours for a typical detached house and can be ten hours or more for a heavy masonry building. Equation (2.9) is the simplest model that has the right qualitative behavior; the same equation underlies the thermostat models used later in this book.

Its consequence for forecasting is the step response. If the outdoor temperature steps up by ΔT and the thermostat holds the indoor temperature at its setpoint, the cooling power required rises not instantaneously but as

$$W(t) = \beta_c \, \Delta T \left[1 - \exp\!\left(-\frac{t - t_0}{\tau} \right) \right], \qquad t \geq t_0, \tag{2.10}$$

reaching 63 % of its final value after one time constant and 95 % after three. Figure 2.6 draws this. The upper panel is outdoor temperature against time in hours, stepping from 24 to 32°C at hour 4; the lower panel is fleet load, which stays at 60 MW until the step and then approaches 74.4 MW along equation (2.10) with $\tau = 3$ hours. The vertical dashed line at hour 7 marks one time constant after the step, where the response has covered 63 % of its range, and the horizontal dashed line marks the final value.

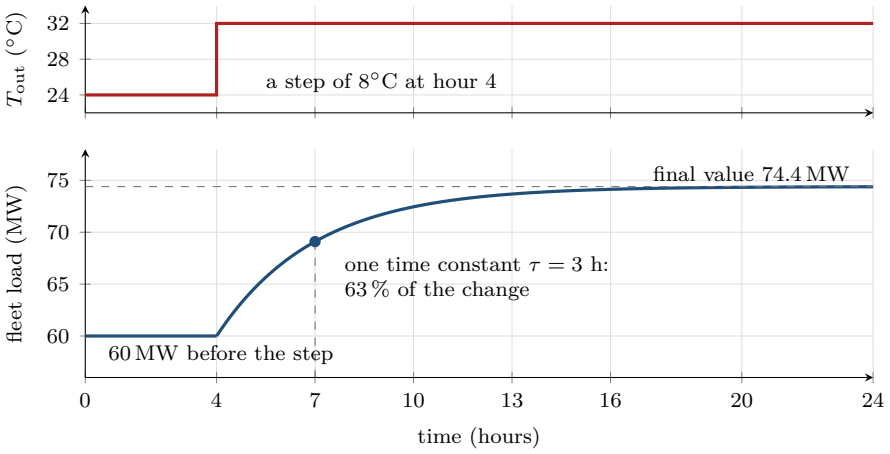


Figure 2.6: Thermal memory as a step response. A step in outdoor temperature (upper panel) produces a lagged rise in fleet load (lower panel) following equation (2.10) with $\beta_c \Delta T = 1.8 \times 8 = 14.4$ MW and $\tau = 3$ hours. Values are illustrative; a real fleet mixes many time constants and its response is a sum of such curves.

Two forecasting consequences follow directly. First, the right weather predictor for the next hour's load is not the next hour's temperature but a weighted average of the last several hours of temperature, the weights decaying with τ. The convenient discrete form is an exponential moving average of the cooling degrees,

$$\mathrm{CD}^{\mathrm{eff}}_t = (1 - \alpha)\, \mathrm{CD}^{\mathrm{eff}}_{t-1} + \alpha \, \mathrm{CD}_t, \qquad \alpha = 1 - e^{-\Delta t / \tau}, \tag{2.11}$$

which for hourly data and $\tau = 3$ hours gives $\alpha = 0.28$. The synthetic week of Figure 2.2 is built with exactly this variable, and its two-hour lag behind the temperature peak, noted in Section 2.1.3, is the visible consequence. Second, the lagged response is itself a source of autocorrelation in load that has nothing to do with the daily cycle: even under constant weather and constant occupancy, a fleet recovering from a disturbance produces a smoothly decaying load trajectory.

2.4.5 Autocorrelation as physical persistence

The conceptual claim of this chapter is now available in full. The autocorrelation of a load series is not primarily a statistical regularity to be modeled away; it is the visible trace of state that persists. Four distinct kinds of memory contribute, with different time scales.

Thermal state. Heat stored in the building fabric decays with the time constant τ of equation (2.9), a few hours to a day. This is the memory that makes $\mathrm{CD}_t^{\mathrm{eff}}$ a better predictor than CD_t.

Weather regime. The weather itself is autocorrelated: a hot day is usually followed by a hot day, because the synoptic pattern driving it persists for several days. This contributes correlation at lags of a day and more, and it is the reason the daily peaks of Figure 2.5 decay slowly rather than dropping off after one day.

Device operating state. A compressor that is running will probably still be running in a few minutes; a device that was offline an hour ago is probably still offline. This contributes strongly at short lags and is also why the duty cycle D_t and the availability count N_{avail} are useful features.

Household schedules. Occupants repeat their behavior, which creates the lag-24 and lag-168 structure. This is memory in a looser sense, a shared cause rather than a persisting physical quantity, and the distinction matters: it is why lag 24 remains informative even after a disturbance that wiped out the thermal state.

> **Principle 2.3.** *Autocorrelation in an energy series is mostly physical and behavioral persistence, not a statistical artifact. The useful statement about* $\mathrm{ACF}(1)$ *is that the state has barely changed; the useful statement about* $\mathrm{ACF}(24)$ *is that the causes recur.*

The practical payoff of this reading is a rule for choosing lags. Include lags at the time scales of the mechanisms: one and two hours for device and thermal state, twenty-four hours for the schedule, one hundred sixty-eight hours for the day type. Lags chosen this way survive changes in the data; lags chosen by searching for whatever correlates best in a particular sample often do not.

2.5 Fleet aggregation and common-mode structure

Everything so far describes a series that is already a sum over many devices. Aggregation is not a neutral operation: it changes which parts of the series are predictable. This section works through what aggregation removes, what it does not remove, and the one situation in which it stops working.

2.5.1 Individual cycling is binary and nearly unpredictable

A single residential air conditioner does not consume a smoothly varying power. Its compressor is either on, drawing a few kilowatts, or off, drawing almost nothing, and the thermostat switches it between the two to keep the indoor temperature inside a deadband. The resulting load is a square wave whose duty cycle encodes the thermal load but whose *phase* is set by where in its deadband that particular house happened to be when the hour began.

For forecasting, the practical statement is that the phase is not predictable and does not need to be. Predicting whether one specific compressor will be energized at 17:23 would require

the indoor temperature, the deadband and the exact switching history of that house. Predicting what fraction of a hundred thousand compressors will be energized at 17:23 requires only the thermal conditions, because the phases average out. The first problem is nearly hopeless and the second is routine, and they differ only by aggregation.

2.5.2 Aggregation smooths idiosyncratic variation

Write the fleet load as a sum over devices,

$$P_{\text{fleet}}(t) = \sum_{i=1}^{N} P_i(t), \tag{2.12}$$

and consider the per-device average $P_{\text{fleet}}(t)/N$. Figure 2.7 shows what happens to that average as N grows. The horizontal axis is time in minutes over three hours and the vertical axis is power per device in kilowatts. The gray square wave is one thermostat cycling between 0 and 3.5 kW with a 20-minute on time and a 15-minute off time, so a duty cycle of 0.57. The purple curve is the average over twenty such devices with independent phases, and the green curve is the average over a thousand. The dashed horizontal line is the expected value, $3.5 \times 0.571 = 2.0$ kW.

The reading is that the single device never takes the value 2.0 kW at all, the twenty-device average fluctuates around it by a few tenths of a kilowatt, and the thousand-device average is visually indistinguishable from it. The arithmetic behind the picture is the standard one: if each device's power has standard deviation $\sigma_{\text{idio}} = P_{\text{AC}}\sqrt{D(1-D)}$, which is $3.5\sqrt{0.571 \times 0.429} = 1.73$ kW here, then the average over N independent devices has standard deviation $\sigma_{\text{idio}}/\sqrt{N}$: 0.39 kW at $N = 20$ and 0.055 kW at $N = 1000$.

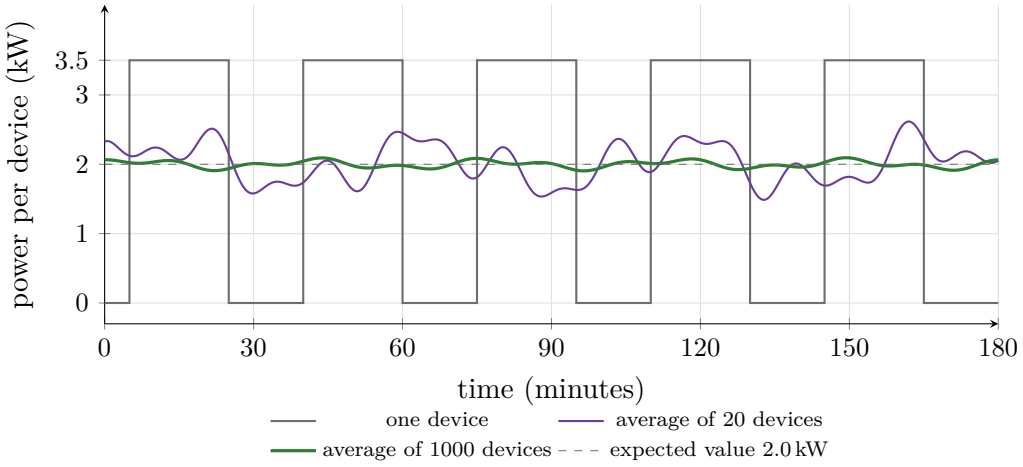


Figure 2.7: Aggregation smooths idiosyncratic cycling. One thermostat cycles between 0 and 3.5 kW with a 20/15 minute duty cycle; the two smooth curves are per-device averages over 20 and 1000 devices with independent phases, drawn as 2.0 kW plus sums of sinusoids of amplitude 0.39 kW and 0.055 kW respectively, which are the $\sigma_{\text{idio}}/\sqrt{N}$ values of Section 2.5.2. Values are illustrative.

Aggregation therefore buys predictability, and a great deal of it. It is the reason an aggregator can quote a fleet forecast with a few percent error while being unable to say anything useful about any individual customer, and the reason Chapter 11 treats fleet size as a forecasting resource.

2.5.3 Common-mode error survives aggregation

The $1/\sqrt{N}$ argument assumes the deviations are independent across devices, and the most important source of forecast error in a fleet is not. Every device in the territory experiences one

weather field, and the forecaster uses one weather forecast for it. If that forecast is 2°C too cool, every device's load is underpredicted at once. Separating the two kinds of deviation,

$$\mathrm{Var}\left(\frac{P_{\mathrm{fleet}}(t)}{N}\right) = \frac{\sigma_{\mathrm{idio}}^2}{N} + \sigma_{\mathrm{com}}^2, \tag{2.13}$$

where σ_{com} is the standard deviation of the shared component, shows what aggregation can and cannot do. The first term vanishes as N grows; the second does not move at all.

The numbers decide which term matters. With $\sigma_{\mathrm{idio}} = 1.73\,\mathrm{kW}$ per device and a shared weather error of 1.5°C at 0.08 kW per degree per device, so $\sigma_{\mathrm{com}} = 0.12\,\mathrm{kW}$, the two terms are equal when $N = (1.73/0.12)^2 \approx 208$ devices. Above a few hundred devices the fleet's forecast uncertainty is almost entirely common-mode. For a fleet of a hundred thousand the idiosyncratic term contributes 0.006 kW per device and the shared term 0.12 kW: a factor of more than twenty in standard deviation, and more than four hundred in variance.

Principle 2.4. *Aggregation averages away idiosyncratic device variation as $1/\sqrt{N}$ and does nothing to error shared across devices. Beyond a few hundred devices, improving a fleet forecast means improving the shared inputs, above all the weather forecast, not modeling individual devices better.*

This is also the reason Chapter 1 warned that the variance of a difference of two forecasts is not the sum of their variances. Both P^{base} and P^{ctrl} depend on the same weather forecast, so their errors are positively correlated and the flexibility $F = P^{\mathrm{base}} - P^{\mathrm{ctrl}}$ is more certain than either of its terms. The joint treatment that makes this precise is the subject of Chapters 10 and 11.

2.5.4 Synchronization destroys the smoothing

The independence of device phases is an assumption about normal operation, and there is one reliable way to break it: send every device the same command at the same time. After a curtailment is released, every compressor in the fleet restarts at once, and the fleet load briefly exceeds what it would have been with no event at all. The devices have been phase-locked by the control action, the $1/\sqrt{N}$ averaging no longer applies, and the fleet behaves for a while like one very large device.

A natural experiment with the same structure occurs after a distribution outage, where the simultaneous restart of every thermostatically controlled load is known as cold-load pickup. The forecasting consequence is that the smoothness of a fleet series is a property of its operating regime rather than of its size, and the regimes in which it is lost are exactly the ones an aggregator creates. That observation leads directly to the next section.

2.6 Nonstationarity, control and the consequences for modeling

A load series' structure is not fixed. The fleet grows, equipment is replaced, tariffs change, and the aggregator intervenes. This section treats the slow changes, the abrupt ones caused by control, and what the whole chapter implies for the choice of model.

2.6.1 Fleet growth and trend

For an aggregator the level L_t of equation (2.2) moves for a reason that has no analog in utility load forecasting: enrollment. A fleet that grows from eighty thousand to a hundred thousand devices over a season sees its load rise by a quarter with no change whatsoever in weather, behavior or device physics. Customers also leave, equipment is replaced with more efficient units, and rooftop solar changes the metered quantity without changing consumption.

The treatment is to normalize rather than to detrend. Dividing by the number of enrolled or available devices gives a per-device series whose level is stable, and multiplying the per-device forecast by the known future device count reconstructs the fleet forecast. This works because the device count at the target time is often known with much greater confidence than the load is: enrollment is administrative data, not a physical process. The same reasoning makes N_{avail} a feature rather than something to be forecast, and the chain of reasoning is the hierarchical structure Chapter 11 develops.

Genuine trend remains after normalization, from changes in the building stock, in equipment efficiency and in customer expectations about comfort, but it is slow: percentage points per year against a daily swing of tens of percent. For horizons up to a few days it can be ignored; for a capacity planning study measured in years it cannot.

2.6.2 Control events as structural breaks

When the aggregator dispatches the fleet, the recorded series changes discontinuously. Figure 2.8 shows the shape. The horizontal axis is the hour of the day from 12:00 to 22:00 and the vertical axis is fleet load in megawatts. The gray dashed curve is the counterfactual baseline P^{base}, what the fleet would have drawn with no intervention; the red solid curve is the recorded load P^{ctrl}; and the shaded band from 17:00 to 19:00 is the event window. Within the window the recorded load falls below the baseline by up to 35 MW: this difference is the flexibility F of Chapter 1. At 19:00 the setpoints are restored, every compressor restarts together as described in Section 2.5.4, and the recorded load jumps *above* the baseline before decaying back to it over about two hours. That excursion is the rebound, or payback.

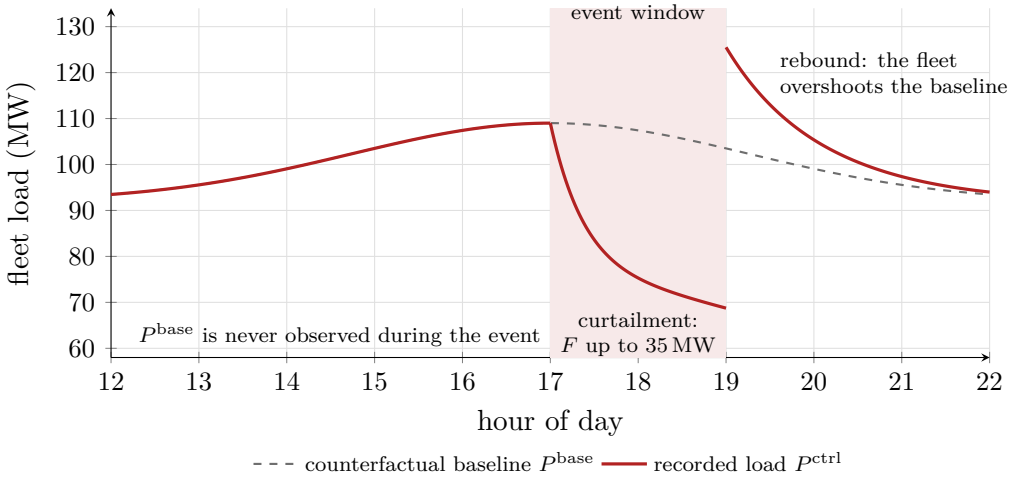


Figure 2.8: A control event as a structural break in the recorded series. The baseline is $92 + 17\exp[-((t-17)/3.2)^2]$ MW; during the event the recorded load is the baseline less $35[1 - e^{-(t-17)/0.4}]$ MW, and after release it is the baseline plus $22e^{-(t-19)/0.8}$ MW. Values are illustrative; the shapes are the curtailment and rebound of a thermostat fleet.

Figure 2.8 contains the counterfactual difficulty of Chapter 1 in visual form. Only the red curve is measured. The gray curve inside the event window is an estimate of something that did not happen, and the flexibility, the vertical gap between them, is therefore never directly observed. Chapter 3 is largely about producing that gray curve defensibly.

Principle 2.5. *A dispatched event is a structural break in the series the aggregator records. Treating the recorded load as natural load teaches a model that afternoons are sometimes mysteriously cheap; the control action belongs in the model as an input, not in the residual.*

2.6.3 The series an aggregator sees is not natural load

The practical consequences of the preceding subsection run through the whole modeling pipeline.

Event hours must be labeled. Every historical hour needs a flag saying whether an event was active, which cohort was addressed, and what the command was. Without the flag the training data mixes two different physical regimes under one label.

The target must be chosen deliberately. A model fitted to the recorded series predicts P^{ctrl}; a model fitted to event-free hours only, or to the recorded series with the control as an input, predicts P^{base}. Both are legitimate, and they answer different questions, which is the distinction Chapter 1 drew.

Rebound hours are not ordinary hours either. The two hours after an event look, to a model without a control feature, like an inexplicable surge. They are a deterministic consequence of the event, and a model given the event history can predict them.

The control term is endogenous. Because the aggregator chooses when to dispatch, and chooses to dispatch on hot afternoons, the event flag is correlated with weather. A model that includes the flag but not the weather will attribute the weather effect to the flag. This is the standard confounding problem, and the remedy is the standard one: include both.

2.6.4 Which structure each model class captures

The structures named in this chapter are the reason the book has the chapters it has. Table 2.2 lists each structure against the mechanism that represents it and the chapter where it appears. The table should be read as a specification: a model class is a good choice for a series when it represents that series' dominant structures, and the way to find out which those are is the diagnostic sequence of Section 2.6.5.

One row of the table deserves emphasis because it is the chapter's conceptual claim applied to model choice. Residual autocorrelation, treated in Chapter 6, is a symptom: it says that some state influencing load is not in the feature set. The autoregressive correction is a legitimate response, and so is finding the missing state. The latter is usually better when the state is identifiable, which is why a feature such as CD_t^{eff} of equation (2.11) often removes more residual correlation than an autoregressive term does.

2.6.5 A diagnostic sequence for a new series

Everything in this chapter can be checked on a new series in under an hour, and doing so before choosing a model is the most reliable way to avoid wasted effort. The sequence is:

1. Plot two weeks of the raw series. Look for the daily cycle, the weekend, obvious gaps, flat stretches that indicate a stuck meter, and any notches that indicate control events.

2. Plot the hour-of-day profile by day type, as in Figure 2.3. This gives the size of the calendar component and shows whether the day types chosen are the right ones.

3. Plot load against outdoor temperature, as in Figure 2.4, coloring points by hour of the day. Read off the balance temperatures, the two slopes, and whether the high-temperature end saturates.

4. Plot the autocorrelation function out to at least 200 lags, as in Figure 2.5, and the partial autocorrelation out to 30. Read off the lags worth using as features.

Table 2.2: The structures of an energy series and the model classes that represent them.

Structure	Represented by	Chapter
Lag-one persistence	the persistence forecast; a lagged load feature	3, 4
Daily and weekly cycle	seasonal persistence and similar-day averages; calendar features; day-type models	3, 4
Weather response with a kink	degree variables in a linear model; a tree ensemble, which finds the kink itself	4, 5
Saturation and interactions	tree ensembles, or explicit interaction and saturating features	5
Thermal memory	lagged and exponentially smoothed weather features	4
Residual autocorrelation	autoregressive and moving-average terms on the residual	6
Latent state carried over many steps	a recurrent network's hidden state; an explicit state-space model	7
Long-range and regime-dependent dependence	attention over a long input window	8
Nonstationarity and regime change	rolling-origin validation and retraining policy	9
Irreducible uncertainty	quantile and distributional forecasts	10
Common-mode fleet uncertainty	a joint forecast across devices; hierarchical reconciliation	11

5. Fit the simplest calendar-and-weather model and plot the autocorrelation of *its residual*. What remains names the missing structure.

6. Plot the series' monthly mean and standard deviation over as many years as exist. This shows trend, fleet growth and any structural break in the record itself.

Step 5 is the one most often skipped and the most informative. A residual whose autocorrelation is flat near zero says the calendar and weather terms have taken the predictable structure; a residual with a spike at lag 24 says the calendar treatment is too coarse; a residual with slow decay says a state variable, most often a thermal one, is missing.

Conceptual questions

"Why does yesterday's load help predict today's?"
A strong answer does not stop at the word autocorrelation, which names the observation rather than explaining it. It identifies the mechanisms. The same households are on the same schedules, so the causes of load at 17:00 yesterday are largely the causes of load at 17:00 today, which is a shared-cause argument rather than a persistence argument. The weather is autocorrelated over several days, so yesterday's temperature is informative about today's and yesterday's load encodes yesterday's temperature. The building stock has thermal memory of a few hours to a day, so stored heat genuinely carries across. The fleet composition and device availability change slowly, so yesterday's load encodes the level. And a good answer closes by saying what this implies operationally: that the lag-24 feature is standing in for an unobserved state, that it stops helping when any of the mechanisms is interrupted, as it is on a holiday or after a dispatched event, and that the way to find out whether the mechanism or the lag is doing the work is to check whether the residual still carries structure at lag 24.

2.7 What to retain

Definition 2.1 (Structure of an energy series)**.** An hourly energy series is the metered output of a physical system, $x_{t+1} = g(x_t, u_t, w_t)$ with $P_t = h(x_t) + \varepsilon_t$, and decomposes in practice into a level, a calendar component, a weather component, a control component and a residual. Its predictability comes from the mechanisms behind those components: schedules create the daily and weekly cycles, space conditioning creates a kinked and saturating weather response, thermal mass and device state create persistence, and aggregation over many devices removes idiosyncratic variation while leaving common-mode uncertainty untouched.

Table 2.3 collects the chapter's objects with the subsection that defines each.

Table 2.3: The objects used to describe the structure of a load series.

Object	Definition	Section
$L_t, S_t, W_t, C_t, \varepsilon_t$	level, calendar, weather, control and residual components of P_t	2.1.1
x_t, g, h	latent state, dynamics and measurement map	2.1.2
$\mu(k, j)$	hour-of-day profile for hour k and day type j	2.2.1
day type	a small grouping of days with the same shape	2.2.2
T^*_{cool}, T^*_{heat}	cooling and heating balance temperatures	2.3.1
CD_t, HD_t	cooling and heating degrees about those balance points	2.3.2
β_c, β_h	cooling and heating sensitivities, MW per degree	2.3.2
A, κ	saturating asymptote and scale of the cooling response	2.3.3
$\text{ACF}(k)$	$\text{Corr}(P_t, P_{t-k})$, read for peaks at 24 and 168	2.4.2
$\text{PACF}(k)$	correlation at lag k net of shorter lags; guides lag choice	2.4.3
$\tau = RC$	thermal time constant of a building	2.4.4
CD^{eff}_t	exponentially smoothed cooling degrees, weight $\alpha = 1 - e^{-\Delta t/\tau}$	2.4.4
σ_{idio}, σ_{com}	idiosyncratic and common-mode device variability	2.5.3
rebound	the post-event overshoot caused by synchronized restart	2.6.2

2.8 Exercises

Exercise 2.1. A fleet has balance temperatures $T^*_{\text{heat}} = 15°\text{C}$ and $T^*_{\text{cool}} = 22°\text{C}$, sensitivities $\beta_h = 1.2$ and $\beta_c = 1.8$ MW per degree, and a balance-band load of 86 MW. Using equations (2.5) and (2.6), compute the load at 8, 19, 27 and 34°C. Then suppose an analyst fits a single straight line in T_{out} to a year of such data and obtains a slope of 0.9 MW per degree. At which of the four temperatures does that line overpredict, and at which does it underpredict?

Exercise 2.2. Let $P_t = 90 + 18\cos(2\pi t/24) + \varepsilon_t$ with ε_t white noise of standard deviation 3 MW, and t in hours. Using equation (2.8), derive $\text{ACF}(k)$ in closed form and evaluate it at $k = 1, 6, 12, 24$. Compare the values with the upper panel of Figure 2.5 and state which features of that figure this series reproduces and which two it cannot, saying what would have to be added to the series to produce each.

Exercise 2.3. A fleet's buildings have a thermal time constant of $\tau = 4$ hours. Using equation (2.10), compute the fraction of the final load change realized 1, 4 and 12 hours after

a step in outdoor temperature. Then compute the exponential-moving-average weight α of equation (2.11) for hourly data, and state how many hours of past cooling degrees carry at least 5 % of the total weight. How would the answer change for a heavy masonry stock with $\tau = 10$ hours?

Exercise 2.4. Each thermostat has $P_{\mathrm{AC}} = 3.5\,\mathrm{kW}$ and duty cycle $D = 0.6$, so $\sigma_{\mathrm{idio}} = P_{\mathrm{AC}}\sqrt{D(1-D)}$. The shared weather forecast error contributes $\sigma_{\mathrm{com}} = 0.10\,\mathrm{kW}$ per device. Using equation (2.13), compute the standard deviation of the per-device fleet average for $N = 1$, 100, 10^4 and 10^6, and find the N at which the two terms are equal. Then, using Figure 2.8, explain in two or three sentences why equation (2.13) understates the variability of the recorded series during the two hours after an event is released.

Chapter 3

Baseline forecasts: persistence, seasonal naive and similar-day methods

3.1 The role of a baseline

Chapter 1 fixed what a forecast is: a statement about y_{t+h} made at issue time t using only the information set $\mathcal{I}_t$. Chapter 2 explained why energy series are predictable at all, by naming the physical and behavioral mechanisms that give them daily and weekly cycles, weather dependence and memory. Those two chapters say nothing about whether any particular model is worth running. That is the question of this chapter:

Given a candidate forecasting model and its measured error, how do we know whether the model is worth anything?

The answer is that an error cannot be judged on its own. It has to be compared with the error of a reference forecast so simple that nobody would claim credit for it. Those reference forecasts are the subject of the chapter. They are cheap, they take a line of code each, and in energy applications they are frequently strong enough that an elaborate model fails to improve on them.

Jargon: baseline model, naive forecast, persistence, seasonal naive, similar-day average, skill score, demand-response baseline

A *baseline model* (or *reference forecast*) is a deliberately simple forecast used as the yardstick for a more elaborate one. A *naive forecast* is a baseline that copies an observed value instead of computing anything: *persistence* copies the most recent observation, and the *seasonal naive* forecast copies the observation one seasonal cycle earlier (24 hours, or 168 hours for a weekly cycle). A *similar-day average* copies the mean of the same hour on several earlier days of the same type. The *skill score* is the fractional reduction in error that a model achieves relative to a named baseline. The *demand-response baseline* is a different object with the same name: the counterfactual load P_t^{base} that a fleet would have drawn had it not been dispatched.

Key idea

A forecast error has no meaning in isolation. It acquires meaning only as a ratio to the error of a simple reference forecast, measured at the same horizon, on the same out-of-sample period, for the same target.

This section separates the two unrelated things the word *baseline* denotes in energy forecasting, shows that a reported error figure is uninterpretable without a reference, and states the requirement that the reference be a credible one.

3.1.1 Two meanings of the word baseline

The first meaning is statistical. A baseline model is a simple forecasting rule used for comparison: persistence, the seasonal naive forecast, a similar-day average, a small weather regression. It is a property of the evaluation, not of the physical system, and its only purpose is to put a candidate model's error on a scale. Sections 3.2 to 3.5 use the word in this sense.

The second meaning is operational, and it was introduced in Chapter 1 as P_t^{base}: the load a fleet would have drawn at time t in the absence of any intervention. That quantity is part of the *target*, since flexibility is defined as $F_t = P_t^{\text{base}} - P_t^{\text{ctrl}}$. It is a counterfactual: on an hour in which the aggregator dispatched, the controlled load P_t^{ctrl} is metered and the baseline load P_t^{base} is not, and never will be.

Both meanings are standard and both appear in this book, as the notation chapter warns. They collide in Section 3.6, which is the interesting case: when the target itself is a counterfactual baseline, the choice of baseline model is no longer a question of evaluation hygiene but a direct contribution to the forecast error.

3.1.2 An error figure alone is uninterpretable

Suppose a report states that a long short-term memory network predicts the aggregator's hourly fleet load day-ahead with RMSE = 6.2 MW. Taken alone the number is unreadable. Three pieces of context change its interpretation completely.

If the seasonal naive forecast of Section 3.2.2 — a rule that copies yesterday's load at the same hour and has no parameters at all — achieves RMSE = 6.3 MW on the same period, then the network has reduced the error by 0.1 MW, or 1.6 %. The modeling effort, the training infrastructure, the inference latency and the monitoring burden have bought almost nothing, and the honest conclusion is that the fleet's day-ahead load is explained by its own daily periodicity.

If a gradient-boosted tree ensemble on lag and weather features achieves RMSE = 5.1 MW on the same period, the interpretation changes again: the network is now not only unnecessary but worse than a model that trains in seconds. The 6.2 MW figure, which looked like a result, turns out to describe a model that should be discarded.

> **Principle 3.1.** *A reported forecast error is uninterpretable unless the report also names the baseline it is compared with, the horizon at which both were evaluated, and the out-of-sample period on which both were measured.*

3.1.3 The baseline must be credible

The comparison can be gamed, and usually not deliberately. The analyst who builds the sophisticated model is also the one who chooses what to compare it against, and the weakest available baseline flatters the result most. Comparing a day-ahead model against persistence — which, as Section 3.4.1 shows, is close to useless at a 24-hour horizon — produces an impressive skill score that says nothing about whether the model is better than copying yesterday.

The requirement is therefore the opposite of the convenient one: the baseline should be the strongest simple alternative available, not the weakest. A weather-adjusted seasonal baseline of the kind built in Section 3.3.1 costs an afternoon and is a far more demanding reference than persistence. If the candidate model beats it, the result means something.

Principle 3.2. *Choose the baseline to be the strongest simple alternative, not the one that is easiest to beat. A skill score reported against a weak baseline measures the weakness of the baseline.*

3.2 Naive and seasonal baselines

The five rules in this section use nothing but past values of the series itself: no weather, no calendar, no fitted coefficients. They are the first things to compute on any new series, partly as references and partly because the pattern of which one wins is informative about the series.

Figure 3.1 shows four of them at work on one day. The horizontal axis is the hour of a Monday; the vertical axis is the aggregator's fleet load in megawatts. The solid black curve is the load that actually occurred, with the overnight trough near 05:00, a modest morning bump, an afternoon dip and the evening peak at 20:00 that Chapter 2 attributed to occupancy and cooling demand coinciding. The three colored curves are the forecasts that the persistence, lag-24 and lag-168 rules would have issued for that day. Their horizons are deliberately different, which is discussed below, and the subsections that follow take them one at a time.

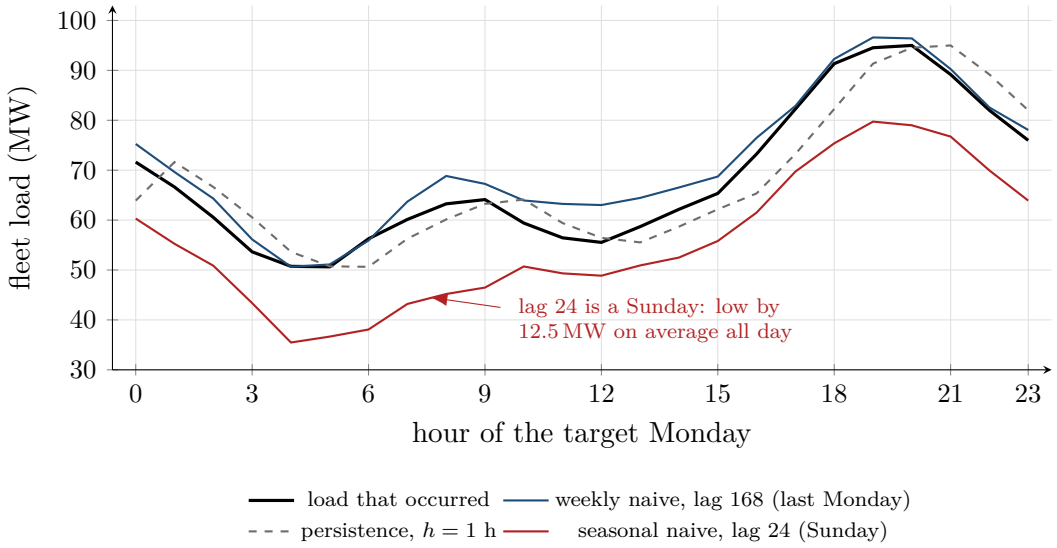


Figure 3.1: Four naive baselines on one day of the aggregator's fleet load. The series is synthetic, generated by `scripts/ch03_data.py` (seed 3) as three weeks of hourly load from a diurnal shape with a morning and an evening Gaussian bump, a weekday or weekend level factor of 1.00, 0.88 or 0.82, a cooling term $1.9\,\max(T_t - 18°\mathrm{C}, 0)$ in MW and an AR(1) noise process with coefficient 0.75 and innovation standard deviation 1.2 MW; the generating model is stated in full in the script's docstring. On this day the errors are RMSE = 3.58 MW for the lag-168 forecast, 5.23 MW for one-hour persistence, 9.80 MW for the four-hour moving average (not plotted) and 12.95 MW for the lag-24 forecast, whose bias is +12.49 MW because the previous day was a Sunday.

3.2.1 Persistence

The persistence forecast, also called the naive or random-walk forecast, predicts that nothing will change:

$$\hat{y}_{t+h} = y_t. \tag{3.1}$$

At 14:00 the fleet is drawing 82 MW, so the forecast for 14:15 is 82 MW. There is nothing to fit, nothing to store and nothing to monitor except the data feed.

It is hard to beat at short horizons, and the reason is quantitative rather than philosophical. Writing the forecast error as the sum of the increments the series will make between issue and target,

$$y_{t+h} - y_t = \sum_{k=1}^{h} \Delta y_{t+k}, \qquad \Delta y_s = y_s - y_{s-1}, \tag{3.2}$$

the persistence error is small whenever the increments are small relative to the level of the series — that is, whenever the horizon is short compared with the time the system needs to move. A fleet of thermostats cannot change its aggregate draw by much in fifteen minutes: the buildings have thermal time constants of hours, the occupants are not all making decisions at once, and the compressors that are running stay running for several minutes. Persistence is therefore excellent at fifteen minutes and usually poor at twenty-four hours, where the sum in equation (3.2) spans an entire diurnal cycle. The dashed gray curve of Figure 3.1 shows the one-hour version: it is the load curve shifted right by one hour, which tracks the level well and systematically lags every ramp.

Equation (3.1) is also the optimal forecast under a specific model, which is worth knowing because it says exactly when persistence cannot be improved. If the series is a random walk, $y_{t+1} = y_t + \varepsilon_{t+1}$ with ε independent of the past and mean zero, then $\mathbb{E}[y_{t+h} \mid \mathcal{I}_t] = y_t$ for every h, and no model of any complexity can do better. Energy series are not random walks — they are strongly periodic, which is exactly the structure persistence ignores — so beating persistence at long horizons is easy, and the right conclusion from a model that fails to is that the model is broken.

3.2.2 Seasonal persistence at lag 24

The seasonal naive forecast replaces "the latest value" with "the value one cycle ago", where the cycle is the 24-hour period of Chapter 2:

$$\hat{y}_{t+h} = y_{t+h-24}. \tag{3.3}$$

Yesterday at 17:00 the fleet drew 110 MW, so the forecast for today at 17:00 is 110 MW. The information requirement is visible in the subscript: $t + h - 24 \le t$ requires $h \le 24$ hours, so the rule is available for horizons up to one day and no further.

Two features deserve attention. First, at exactly $h = 24$ hours the seasonal naive forecast and persistence coincide, because $y_{t+h-24} = y_t$; the two rules are distinct only for $h < 24$, where persistence uses the most recent observation and equation (3.3) deliberately reaches back for a staler one that sits at the right phase of the daily cycle. Second, what the rule copies is the entire shape of the day: the trough, the ramps and the peak, at the right hours, without any model of them. It also copies the weather regime, since yesterday's temperature is usually close to today's.

Its failures follow from the same mechanism. Anything that differed yesterday is imported wholesale into the forecast. In Figure 3.1 the target day is a Monday and the previous day is a Sunday, on which the fleet's level factor was 0.82 rather than 1.00: the red curve is below the load at every hour of the day, with a bias of +12.49 MW in the forecast error and an RMSE of 12.95 MW — more than twice the error of one-hour persistence and more than three times that of the weekly rule. A weather change, a holiday or a dispatch event on the previous day does the same kind of damage, and Section 3.4.2 returns to the list.

3.2.3 Weekly persistence at lag 168

The remedy for the day-type problem is to copy a day of the same type, which for hourly data means reaching back one week:

$$\hat{y}_{t+h} = y_{t+h-168}, \qquad 168 = 24 \times 7. \tag{3.4}$$

Monday at 17:00 is forecast from last Monday at 17:00. Day type, which the lag-24 rule gets wrong on two transitions out of every seven, is now correct by construction, and so are holidays if the holiday happens to fall on the same weekday — which it generally does not, a point taken up in Section 3.3.2.

The gain is largest for loads whose weekday and weekend behavior differ most, which means commercial and industrial loads: an office building's Sunday carries almost no information about its Monday, while last Monday carries a great deal. The blue curve of Figure 3.1 is the lag-168 forecast for the synthetic residential fleet, and even there it is the best of the three, with RMSE = 3.58 MW against 12.95 MW for lag 24.

The cost is staleness. The lag-168 rule uses information a week old, so it imports a week-old weather regime. In a stable summer week that is harmless; across a cold front or at the start of a heat wave it is worse than yesterday's load, because the day type is right and the temperature is wrong by several degrees. The two naive rules fail on different things, which is the argument for carrying both as references and for the weather correction of Section 3.3.1.

3.2.4 Average of similar days

Both lag rules stake the whole forecast on one previous day, including whatever noise that day contained. Averaging over several same-weekday observations removes part of it:

$$\hat{y}_{t+h} = \frac{1}{K} \sum_{k=1}^{K} y_{t+h-168k}, \tag{3.5}$$

which for $K = 4$ is the mean of the same hour on the previous four weeks, all of them the same weekday as the target.

The variance argument is the usual one. If the K days were independent draws with a common mean and a per-day noise variance σ^2, the average would have variance σ^2/K, so four weeks would halve the noise contribution to the forecast error. The assumption of independence is the part that fails in practice, and it fails in a specific way: the component of each day's deviation that comes from the weather is shared between days within the same regime, and shared error does not average away. This is the common-mode uncertainty of Chapter 2 appearing in a baseline rather than in a fleet, and it means the realized improvement from $K = 4$ is smaller than $1/\sqrt{K}$ suggests.

Averaging also introduces a bias whenever the series is drifting. If the load is trending upward — a growing fleet, a warming season — then the mean of the last four same-weekday hours is centered on a point 2.5 weeks in the past and is therefore too low. The practical compromise is a small K, typically three to five, or a weighted average that puts more weight on recent weeks.

3.2.5 Moving average

The last naive rule averages over recent time rather than over previous weeks:

$$\hat{y}_{t+1} = \frac{1}{N} \sum_{i=0}^{N-1} y_{t-i}. \tag{3.6}$$

With $N = 4$ and the last four 15-minute loads equal to 80, 82, 83 and 85 MW, the forecast for the next interval is 82.5 MW.

Equation (3.6) is a low-pass filter, and it buys and sells the same thing. It buys noise reduction: metering noise and the short-period jitter of individual compressors cycling on and off are averaged down by a factor of N in variance. It sells phase accuracy: the mean of the last N observations is an estimate of the series at the center of that window, which is $(N-1)/2$

intervals in the past, so on any sustained ramp the forecast lags. In the numerical example the load is rising by about 1.7 MW per interval and the forecast of 82.5 MW is already 2.5 MW below the last observation.

Which effect dominates is a property of the series at the sampling interval in use, so the rule is a matter of measurement rather than taste: use a moving average when the period-to-period noise is large compared with the period-to-period ramp, and use persistence otherwise. On the hourly series of Figure 3.1 the ramps are large and the noise is small, and the four-hour moving average is accordingly much worse than persistence, RMSE = 9.80 MW against 5.23 MW. On one-minute feeder telemetry the ordering typically reverses.

3.3 Baselines that use domain knowledge

The rules of Section 3.2 know nothing about the physical system. A reference forecast is allowed to know something, and the three in this section do: they use the temperature sensitivity of the load, the calendar, and in the last case a fitted regression. They are harder to beat, which is the reason to use them.

3.3.1 Weather-adjusted baselines

The seasonal naive forecast fails across a weather change because it copies a load that was produced at a different temperature. The correction is to copy the load and then move it by the temperature difference times a sensitivity:

$$\hat{y}_{t+h} = y_{\mathrm{ref}} + s(\hat{T}_{t+h|t} - T_{\mathrm{ref}}), \tag{3.7}$$

where y_{ref} is the load on the reference day at the same hour, T_{ref} the temperature that prevailed then, $\hat{T}_{t+h|t}$ the temperature forecast for the target time as issued at t, and s the load sensitivity in megawatts per degree. Yesterday at 17:00 the fleet drew 100 MW at 30°C; tomorrow's forecast for 17:00 is 36°C; the sensitivity is about 2 MW per °C; the adjusted baseline is

$$\hat{P} = 100 + 2\,(36 - 30) = 112\,\mathrm{MW}. \tag{3.8}$$

Note that equation (3.7) uses the temperature *forecast*, not the temperature that will be observed. Substituting the observed value is the data leakage of Chapter 1: it produces a baseline that cannot be run operationally and that is unfairly strong in a backtest.

The sensitivity s is not a constant, and this is where a weather-adjusted baseline is most easily misapplied. Figure 3.2 plots the relationship the adjustment assumes. The horizontal axis is outdoor temperature and the vertical axis is the fleet's load at a fixed hour; the curve is flat below a balance temperature, here 18°C, and rises linearly above it with slope $s = 2$ MW per °C. The two marked points are the reference day at (30°C, 100 MW) and the forecast day at (36°C, 112 MW) of equation (3.8), and the vertical offset between them is the 12 MW adjustment. Below the balance temperature there is no cooling load to scale, so the same arithmetic applied to a forecast of 15°C would subtract $2 \times 15 = 30$ MW when the correct adjustment is $2 \times 12 = 24$ MW. The slope is valid in the regime it was estimated in. This is the hockey-stick nonlinearity of Chapter 2, and it is the reason cooling degrees rather than temperature are the variable to regress on, a point developed in Chapter 4.

3.3.2 Day matching and exclusions

Equation (3.5) averaged the previous K same-weekday hours without asking whether those days were suitable. Industry demand-response baselines, which have to survive a settlement dispute, instead specify a selection rule: from a pool of recent candidate days, exclude the unsuitable ones, then average the best K of what remains.

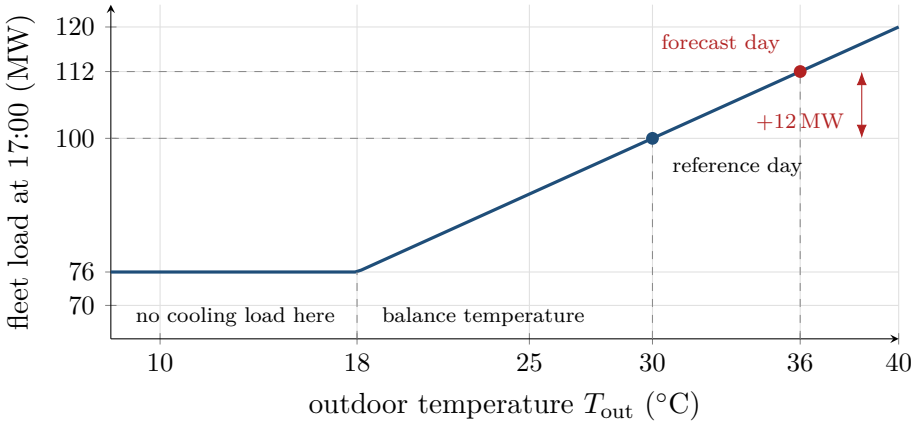


Figure 3.2: The relationship a weather-adjusted baseline assumes. The curve is $P_{\text{load}} = 76 + 2\,\max(T_{\text{out}} - 18°\text{C}, 0)$ in MW, a flat branch below a balance temperature of 18°C and a linear cooling branch above it with sensitivity $s = 2$ MW per °C; the parameters are illustrative and chosen so that the curve passes through the reference point of equation (3.8). The marked points are the reference day and the forecast day of that equation, and the arrow is the 12 MW adjustment.

Figure 3.3 shows the selection. The grid is a calendar: each row is a week, each column a day of the week, with weekend cells shaded gray. The target is the Monday of the current week, marked in green at the bottom left. The candidate pool is the Monday column of the five preceding weeks, outlined in blue. One candidate, the Monday three weeks before, is crossed out in red because the aggregator dispatched the fleet that afternoon, so its metered load is a controlled load and not a baseline load. The remaining four are averaged to form the forecast, which is why the pool reaches back five weeks to deliver $K = 4$ usable days.

Three kinds of exclusion matter in practice. Dispatch days are excluded for the reason just given, and Section 3.4.3 shows what happens when they are not. Public holidays are excluded because their load resembles a Sunday regardless of which weekday they fall on, and the same applies to the days immediately around them. Days with known data problems — a meter outage, a communications failure, a partial fleet — are excluded because the average is not robust to a single corrupted day. What remains can be ranked by weather similarity to the target, so that the K selected days are both the same day type and the closest in temperature; that ranking is what turns a similar-day average into a credible day-ahead reference.

3.3.3 A small weather and calendar regression

The strongest member of the baseline family is a linear regression on a handful of variables that are known to matter:

$$\hat{y}_{t+h} = \beta_0 + \beta_1 \text{CD}_{t+h} + \beta_2 \text{HD}_{t+h} + \sum_j \gamma_j \, \mathbf{1}[\text{hour} = j] + \sum_d \delta_d \, \mathbf{1}[\text{day type} = d], \tag{3.9}$$

with CD and HD the cooling and heating degrees of the notation chapter, computed from the temperature forecast $\hat{T}_{t+h|t}$, and the two indicator sums giving each hour of the day and each day type its own level. The model has a few dozen coefficients, fits in milliseconds by least squares, and is interpretable line by line.

It belongs in this chapter rather than in Chapter 5 because of how it is used. Nobody deploys equation (3.9) as the production forecaster for an aggregator; it is deployed as the reference that the production forecaster has to beat, and it is the rung of the hierarchy at which claims about machine learning start to become meaningful. It captures, in the crudest possible

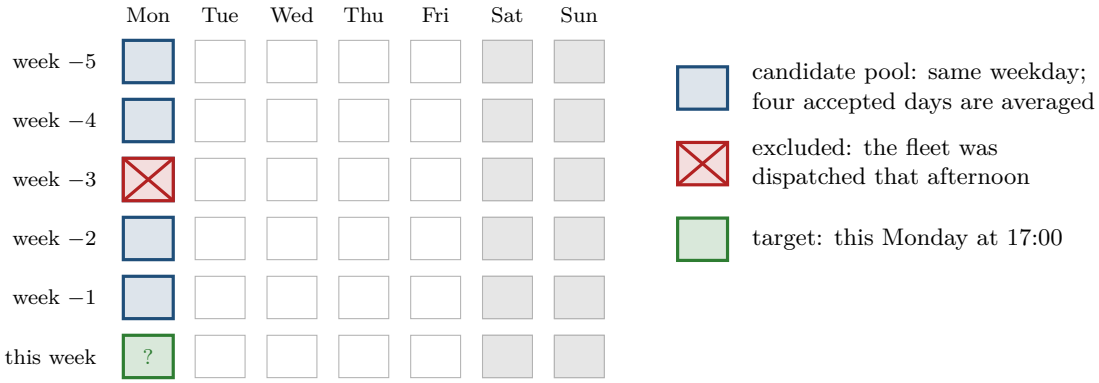


Figure 3.3: Similar-day selection for a Monday 17:00 forecast. Rows are weeks, columns are days of the week, and shaded columns are weekend days. The blue cells are the candidate pool of same-weekday days; the crossed cell is excluded because the fleet was dispatched that day, so its meter record is a controlled load; the green cell is the target. The pool spans five weeks so that four usable days remain for equation (3.5) with $K = 4$.

form, the three structures Chapter 2 identified as dominant: the daily profile, the day-type effect, and the weather dependence. A model that does not beat it is not capturing anything those three do not already explain.

3.4 Error sources and failure modes

Each baseline is wrong in a characteristic way, and knowing which way is what allows a baseline set to be chosen rather than guessed. This section takes the horizon first, because it dominates, then the three events that break the naive rules, and ends with the rule for matching baselines to horizons.

3.4.1 Error growth with the horizon

Figure 3.4 shows how the error of three baselines changes with the forecast horizon. The horizontal axis is the horizon h in hours, on a scale from fifteen minutes to a day; the vertical axis is the root-mean-square error in megawatts. The gray curve is persistence, the dashed curve is a four-period moving average, and the flat red line is the seasonal naive forecast at lag 24.

The persistence curve rises steeply and then flattens, which is the behavior equation (3.2) predicts: the error is the accumulated increment, so it grows with the horizon until the horizon exceeds the memory of the series, after which it saturates at roughly the standard deviation of the series itself. The moving-average curve starts higher, because of the phase lag of Section 3.2.5, and ends lower, because at long horizons the smoothing is worth more than the lag costs. The seasonal naive line is essentially flat, since copying yesterday's 17:00 load is the same operation whether it is done at 16:00 or at 09:00; its error depends on how similar yesterday was, not on how long ago the forecast was issued.

The two crossings are what the figure is for. Below about half an hour persistence is the best of the three. Beyond about two hours the flat seasonal line is below the rising persistence curve, and the ordering of the baselines has reversed. Any statement of the form "persistence is a strong baseline" is therefore a statement about a horizon, and a comparison that holds at fifteen minutes can be meaningless at a day.

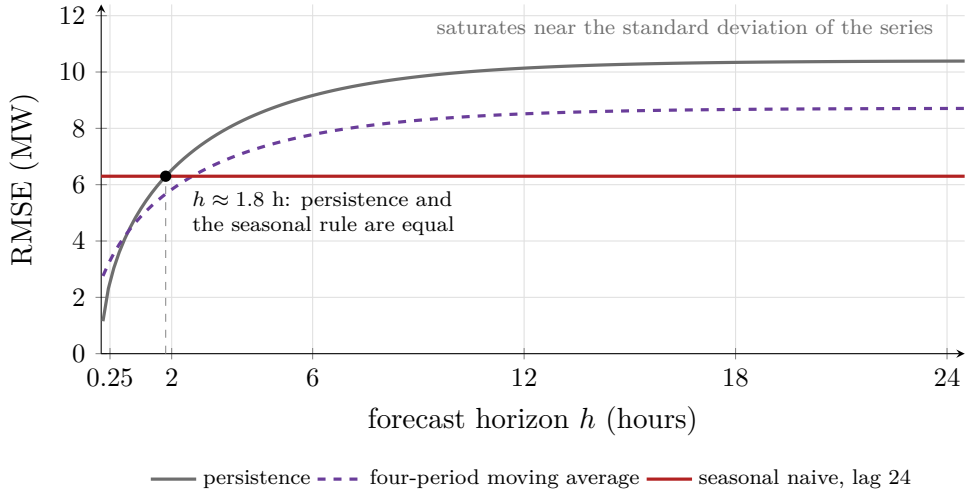


Figure 3.4: Illustrative error growth of three baselines with the forecast horizon. The curves are $\mathrm{RMSE}_{\mathrm{pers}}(h) = 10.4\sqrt{1 - e^{-h/4}}$, $\mathrm{RMSE}_{\mathrm{ma}}(h) = \sqrt{69.2\,(1 - e^{-h/4}) + 6.76}$ and $\mathrm{RMSE}_{\mathrm{snaive}}(h) = 6.3$, in MW with h in hours; the exponential form encodes a four-hour memory and the constant $6.76 = 2.6^2$ is the squared phase-lag penalty of the moving average. The parameters are illustrative; what is general is the shape, a rising and saturating persistence curve crossing a horizon-independent seasonal one.

3.4.2 Regime changes: weather, calendar and day type

Outside the horizon, the naive rules fail when the day they copy from differs from the day they forecast. Three mechanisms produce that difference, and each disables a specific rule.

A weather change. A cold front or the onset of a heat wave moves the cooling load by several megawatts between consecutive days, so the lag-24 rule imports the wrong temperature regime and the lag-168 rule imports one a week staler still. Only the weather-adjusted baseline of Section 3.3.1 and the regression of Section 3.3.3 have any defense, and theirs is limited by the accuracy of the temperature forecast they consume.

A calendar discontinuity. A public holiday has the load shape of a Sunday on a weekday. The lag-24 rule is wrong on the holiday and wrong again on the day after; the lag-168 rule is wrong on the holiday and, a week later, wrong once more when it copies the holiday into an ordinary day. Both failures are avoidable, since the calendar is known years in advance, and the fix is the exclusion list of Section 3.3.2 rather than a better model.

A day-type transition. The Sunday-to-Monday transition of Figure 3.1 is the routine case, costing the lag-24 rule a 12.5 MW bias on that fleet. The lag-168 rule handles it by construction, which is the division of labor between the two: lag 24 tracks weather and loses day type, lag 168 tracks day type and loses weather.

3.4.3 Control events contaminate the history

A failure peculiar to an aggregator's own data is that the historical record contains hours the aggregator interfered with. If the fleet was curtailed yesterday from 17:00 to 18:00, the metered load at 17:00 yesterday is P^{ctrl}, not P^{base}. Using it as a baseline for today therefore predicts a load that was suppressed by the dispatch, and the forecast inherits the full size of

that suppression as a bias. With a 45 MW curtailment on a 120 MW fleet, the lag-24 baseline for today is low by 45 MW, which is not a modeling error of any kind but a bookkeeping one.

The consequences compound, because the error has the wrong sign for the business. An understated baseline makes the next day's forecast load too low, which makes the estimated flexibility too small, which makes the aggregator bid conservatively. And the rebound hour after a curtailment has the opposite problem: the metered load from 18:00 to 19:00 is inflated by recovering buildings, so copying it overstates.

The remedy is procedural and should be built before any model. Every dispatch is recorded with its start, end and participating cohort; every affected hour, including the rebound hours, is flagged in the historical table; and every baseline rule — the lag rules, the similar-day average, and the training set of the regression — skips flagged hours. Figure 3.3 draws exactly this exclusion. The same flags are what make honest measurement and verification possible later, so the cost is paid once.

3.4.4 One baseline per horizon

Combining the horizon dependence of Section 3.4.1 with the failure modes above gives the practical rule. Table 3.1 lists three operational horizons with the information that dominates each, the baseline that is hardest to beat there, and what that baseline is blind to. The pattern across the rows is the one Chapter 1 anticipated: recent observations dominate at short horizons, because the system has not had time to move, and weather and calendar dominate at long ones, because the initial state has been forgotten.

Table 3.1: Which baseline is hardest to beat, by horizon.

Horizon	Dominant information	Strongest simple baseline	Blind to
15 min	the last few observations	persistence, equation (3.1); a short moving average on noisy feeds	any ramp already under way
3 h	recent load, the daily profile, the near-term weather forecast	a combination: recent load, the same hour yesterday, a temperature correction	changes of day type, dispatch in between
24 h	weather forecast, calendar, day type	weather-adjusted seasonal or weekly persistence, equations (3.4) and (3.7); the regression of equation (3.9)	behavior changes, holidays not in the exclusion list

Principle 3.3. *There is no single baseline for all horizons. A baseline set is chosen per horizon, and a model is compared with the strongest baseline at the horizon the model will actually be used at.*

3.5 Measuring improvement over a baseline

With a baseline set in place, the comparison can be made numerical. This section defines the skill score, works a comparison through on the aggregator's day-ahead problem, states why the simple competitor often wins outright, and collects the baselines into the ladder that a forecasting project climbs.

3.5.1 The skill score

The skill score expresses a model's error as a fractional reduction relative to a named baseline:

$$\text{Skill} = 1 - \frac{\text{RMSE}_{\text{model}}}{\text{RMSE}_{\text{baseline}}}. \tag{3.10}$$

If the baseline achieves $\text{RMSE}_{\text{baseline}} = 10\,\text{MW}$ and the model achieves $\text{RMSE}_{\text{model}} = 7\,\text{MW}$, then $\text{Skill} = 1 - 7/10 = 0.30$: the model reduces the root-mean-square error by 30 % relative to the baseline. The statement is considerably more informative than "the RMSE is 7", because it carries its own reference.

Four properties are worth stating. A skill of zero means the model and the baseline are equally accurate; a negative skill means the model is worse, and negative skill scores are common and should be reported rather than suppressed. A skill of one requires a perfect forecast, so in energy applications values above about 0.5 against a credible baseline are rare. The score is defined with respect to one metric, and the skill computed from MAE differs from the skill computed from RMSE because the two metrics weight large errors differently. Finally, the score is only comparable between studies that used the same baseline, horizon, target and evaluation period: four numbers that belong in the caption of every skill figure.

Skill scores against different baselines convert into one another. If a model has skill S_B against a common reference and another candidate has skill S_A against the same reference, the first model's skill measured against the second is

$$\text{Skill}(B \mid A) = 1 - \frac{1 - S_B}{1 - S_A}, \tag{3.11}$$

which follows from substituting the two definitions. The formula makes the inflation concrete: two models whose skills against persistence are 0.40 and 0.39 differ by only $1 - 0.60/0.61 = 0.016$ when compared directly.

3.5.2 A worked comparison

Table 3.2 collects the day-ahead results for the aggregator's hourly fleet load: four baselines and three candidate models, all evaluated at $h = 24$ hours over one summer season of out-of-sample hours. The first column names the forecaster, the second gives its RMSE, and the last two give its skill by equation (3.10) against two different references — persistence, and the seasonal naive rule.

Table 3.2: Day-ahead hourly fleet load, one summer season out of sample. The RMSE values for the gradient-boosted ensemble and the recurrent network are the ones quoted in Section 3.1.2; the remaining values are illustrative and consistent with them. Skill is computed from equation (3.10).

Forecaster	RMSE (MW)	Skill vs. pers.	Skill vs. lag 24
Persistence, eq. (3.1)	10.4	0.00	−0.65
Seasonal naive, lag 24	6.3	0.39	0.00
Weekly naive, lag 168	6.6	0.37	−0.05
Four-week similar-day average	5.9	0.43	0.06
Weather and calendar regression	5.4	0.48	0.14
Gradient-boosted ensemble	5.1	0.51	0.19
Recurrent network (LSTM)	6.2	0.40	0.02

Figure 3.5 plots the last two columns as grouped bars. The horizontal axis lists the six forecasters other than persistence itself; the vertical axis is skill, with zero drawn as a solid line

so that negative bars are visible. The dark bars are skill against persistence and the light bars are skill against the lag-24 rule.

The figure shows the whole argument of the chapter in one place. Every dark bar is substantial, because persistence is a weak reference at a 24-hour horizon, and a report that used only those bars would conclude that all six forecasters work. The light bars collapse: the recurrent network's skill falls from 0.40 to 0.02, so against the rule of copying yesterday the network has reduced the error by one part in fifty and the modeling effort is not justified. The weekly naive rule's bar goes negative, which on this residential fleet is the expected result and on a commercial fleet would not be. Only the regression and the gradient-boosted ensemble retain a meaningful margin, and the ensemble's is the larger: 0.19, about a fifth of the error of the naive rule removed. That ordering — a tree ensemble ahead of a recurrent network on tabular load data — is the ordering Chapters 5 and 7 will explain.

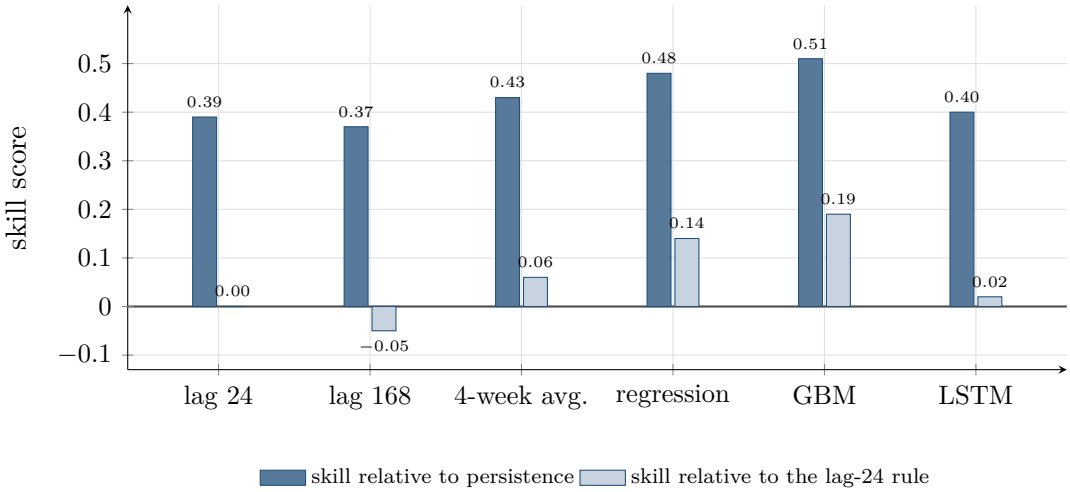


Figure 3.5: The skill columns of Table 3.2, drawn against two references. Dark bars are skill relative to persistence, light bars skill relative to the lag-24 seasonal naive rule, both from equation (3.10) at a 24-hour horizon. The same recurrent network scores 0.40 against one reference and 0.02 against the other.

3.5.3 Why the simple competitor often wins

The baselines in Table 3.2 are not only references; sometimes they are the right thing to deploy. Six properties explain why, and none of them is about accuracy.

A simple model has low variance: few parameters means less sensitivity to the particular sample it was fitted on, which matters most on short histories and for small cohorts. It is debuggable: when a forecast is visibly wrong at 17:00, a coefficient on cooling degrees can be inspected, and a recurrent network's hidden state cannot. It is fast, both to train and to evaluate, which is what allows a forecast to be refreshed every five minutes across thousands of cohorts. It is interpretable, which is what allows an operator to accept or override it. It is robust when data are sparse, since a newly enrolled cohort has no history for a sequence model to learn from but does have a day type and a temperature. And it is easy to monitor: a drift in one coefficient is a legible alarm.

Those properties change what the comparison should conclude. Suppose a transformer with two hundred million parameters achieves RMSE $= 5.0\,\mathrm{MW}$ and a gradient-boosted tree achieves $5.1\,\mathrm{MW}$ on the same evaluation. The transformer is more accurate by $0.1\,\mathrm{MW}$, which is $2\,\%$, and the tree is preferable for operations: it retrains nightly on a laptop, its failures are explicable to a control-room operator, and its $2\,\%$ disadvantage is smaller than the year-to-year

variation in the evaluation itself.

> **Principle 3.4.** *A complex model must earn its complexity. The margin it must clear is not zero but the operational value of the simpler model's variance, latency, interpretability and maintainability.*

3.5.4 The baseline hierarchy

The baselines form a ladder of increasing difficulty, drawn in Figure 3.6. Each block is one rung, labeled with the rule and the section that defines it, and the rungs rise from left to right in the order in which they should be built. The arrow above them is the direction of increasing difficulty to beat; the bracket below separates the four baseline rungs from the fifth, which is where the models of Chapters 5 to 8 live.

The ladder is a work plan as much as a taxonomy. Rung 1 and rung 2 cost minutes and immediately reveal gross errors in the data pipeline: if persistence beats the candidate model at a one-hour horizon, something is wrong with the model, and if the lag-24 rule beats it day-ahead, something is wrong with the features. Rung 3 costs an hour and requires the exclusion list, which has to exist anyway. Rung 4 costs an afternoon and is the first rung that consumes a weather forecast, so it is also the first to expose how much of the error is inherited from that forecast rather than produced by the load model. Only then is rung 5 worth starting, and its claim is reported against rung 4.

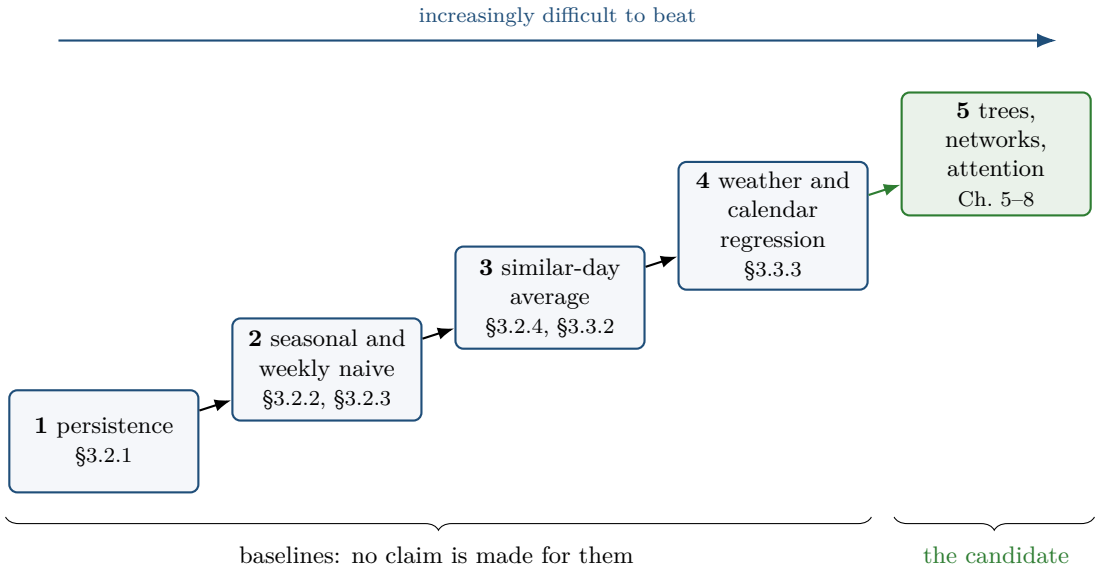


Figure 3.6: The baseline hierarchy. Each rung is a reference forecast, and each is harder to beat than the one before it; the sections and chapters that define them are named in the blocks. The practical rule is to build rungs 1 to 4 before rung 5 exists, and to report rung 5's result against rung 4.

3.6 Baselines for fleet flexibility

Everything so far treated the baseline as a device of evaluation. For the aggregator's actual target it is part of the quantity being predicted, and this section works through what that does to the error. The running example is the thermostat fleet of Chapter 1.

3.6.1 Baselines for the flexibility target

The target is the flexibility F_t, the load reduction the fleet could deliver, and it needs its own reference forecasts. The naive one copies the last comparable event:

$$\hat{F}_t = F \text{ observed at the same hour of the previous similar event,} \tag{3.12}$$

where "similar" means a comparable outdoor temperature, a comparable event duration and the same cohort. Equation (3.12) is the analog of seasonal persistence, and it inherits the same weakness: there may have been only a handful of events, and the nearest one may have occurred at a different temperature.

A stronger and still simple reference is a two-variable regression:

$$\hat{F}_t = \alpha + \beta_1 N_{\text{avail}} + \beta_2 T_{\text{out}}, \tag{3.13}$$

fitted on whatever event history exists. The two regressors are chosen from the physics rather than from a search. The number of available devices N_{avail} sets the scale: flexibility is roughly proportional to how many thermostats are enrolled, connected and not opted out. The outdoor temperature sets the intensity: it determines the duty cycle, so it determines what fraction of the fleet has a compressor running that could be switched off. Equation (3.13) can be fitted from a dozen events and is a demanding benchmark for any fleet-response model, which is the role it plays in Chapter 11.

3.6.2 The counterfactual problem

The deeper difficulty is that F_t is never observed. When the aggregator dispatches at 17:00 it meters P_t^{ctrl} and the baseline load P_t^{base} does not happen; when it does not dispatch, it meters P_t^{base} and P_t^{ctrl} does not happen. Only one of the two worlds is realized, so

$$F_t = P_t^{\text{base}} - P_t^{\text{ctrl}} \tag{3.14}$$

is an estimate even after the fact, and the baseline term in it must come from a model of the kind built in Sections 3.3.2 and 3.3.3.

The forecast error therefore has two sources. Writing $\varepsilon_{\text{base}}$ and $\varepsilon_{\text{ctrl}}$ for the errors of the baseline-load and controlled-load predictions,

$$\varepsilon_F = \varepsilon_{\text{base}} - \varepsilon_{\text{ctrl}}, \tag{3.15}$$

so an excellent controlled-load model does not by itself produce a usable flexibility estimate. The variance of the difference is

$$\operatorname{Var}(\varepsilon_F) = \operatorname{Var}(\varepsilon_{\text{base}}) + \operatorname{Var}(\varepsilon_{\text{ctrl}}) - 2\operatorname{Cov}(\varepsilon_{\text{base}}, \varepsilon_{\text{ctrl}}), \tag{3.16}$$

and the covariance term is usually positive and usefully large, because both predictions consume the same temperature forecast and are moved the same way by its error. This is the correlation Chapter 1 flagged in its worked example, seen from the error side: a shared weather error partly cancels in the difference, which is an argument for building the two models on the same inputs rather than independently.

3.6.3 A worked example on the thermostat fleet

Example 3.1 (The baseline dominates the flexibility error)**.** At 17:00 the fleet's unobserved baseline load is 120 MW and the metered controlled load is 80 MW, so the flexibility actually delivered is $F = 40$ MW. The aggregator's models predicted a baseline of 130 MW and a controlled load of 81 MW, giving $\hat{F} = 49$ MW.

By equation (3.15) the flexibility error is $\varepsilon_F = 10 - 1 = 9$ MW. The controlled-load model was off by 1 MW, or 1.3 %, which by any standard is excellent; the flexibility estimate was off by 9 MW, or 22.5 %, which would be reported as a failed forecast.

Figure 3.7 shows the arithmetic. The three groups on the horizontal axis are the baseline load, the controlled load and the flexibility; in each group the dark bar is the realized value and the light bar the predicted one, in megawatts. The first two groups look similar, with the predicted bars close to the realized ones. The third group, which is the difference of the first two, does not: the 10 MW baseline error and the 1 MW controlled-load error have combined into a 9 MW flexibility error.

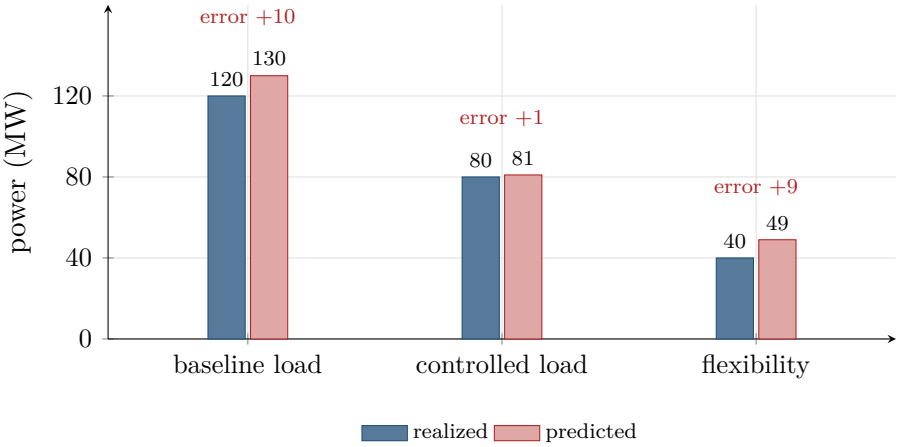


Figure 3.7: The worked example of Section 3.6.3. Dark bars are the realized baseline load, controlled load and flexibility; light bars are the predictions. The flexibility bars are the differences of the bars to their left, by equation (3.14), and the small controlled-load error plus the larger baseline error combine into the 9 MW flexibility error of equation (3.15).

The amplification in the example is not an accident of the numbers. Dividing equation (3.15) by F and keeping only the baseline term,

$$\frac{\varepsilon_F}{F} \approx \frac{P^{\text{base}}}{F} \cdot \frac{\varepsilon_{\text{base}}}{P^{\text{base}}}, \tag{3.17}$$

so a relative error in the baseline load is magnified by the ratio P^{base}/F, which here is $120/40 = 3$. A fleet that can shed a third of its load converts a 1 % baseline error into a 3 % flexibility error; a fleet that can shed a tenth converts it into a 10 % error. Flexibility is a small difference of two large numbers, and small differences of large numbers inherit amplified relative errors.

Principle 3.5. *When the target is a counterfactual difference, the baseline model is not a yardstick but a component of the forecast. Its error enters the answer amplified by P^{base}/F, so baseline estimation is the first problem to solve in flexibility forecasting, not the last.*

3.6.4 Reporting a result

The chapter's content reduces to a short checklist for presenting any forecasting result, and it is the checklist a reviewer probes for.

Conceptual questions

"You built an LSTM with RMSE *8 MW. Is that good?"*
A strong answer declines to say, on the grounds that the question is missing its reference, and then supplies the comparisons that would settle it. The number has to be set against persistence, against seasonal persistence at lag 24, and against a simple weather and calendar model, all evaluated at the same horizon and on the same out-of-sample periods; the skill score of equation (3.10) against each is what makes the 8 MW readable. The answer should add the conditional comparison: average error over a season can hide failure on the hours that matter, so the same baselines should be compared during heat waves, on holidays and on dispatch days, where an operator's decisions are actually exposed. Absolute RMSE on its own does not say whether the model adds value, and if the seasonal baseline is at 8.1 MW the correct conclusion is that it does not.

The checklist behind that answer is: name the target, the issue time and the horizon; name the baseline and show that it is a credible one; report the error of both on the same out-of-sample period, as an absolute number and as a skill score; break the comparison out for the operationally important conditions; and, when the target is flexibility, report the baseline-load error separately, because equation (3.17) says it will dominate.

3.7 What to retain

Definition 3.1 (Baseline forecast)**.** A sophisticated model is useful only if it beats a credible simple alternative, measured at the same horizon on the same out-of-sample data. The baseline set for an energy series should usually contain $\hat{y}_{t+h} = y_t$, $\hat{y}_{t+h} = y_{t+h-24}$, $\hat{y}_{t+h} = y_{t+h-168}$ and a small weather and calendar model. For flexibility forecasting the matter is more than methodological, because $F = P^{\text{base}} - P^{\text{ctrl}}$ makes the estimation of a baseline load part of the forecasting problem itself.

Table 3.3 collects the chapter's objects with the subsection that defines each.

3.8 Exercises

Exercise 3.1. Four consecutive 15-minute fleet loads are 80, 82, 83 and 85 MW. Compute the one-step forecasts given by persistence and by the four-period moving average of equation (3.6), and state the phase lag of the moving average in minutes. The next observed value is 88 MW: give the error of each rule. Then explain, using Section 3.2.5, what property of the series would have to change for the moving average to be the better rule.

Exercise 3.2. Using Table 3.2, compute the skill of the gradient-boosted ensemble against each of the four baselines. Then use equation (3.11) to obtain the recurrent network's skill against the weather and calendar regression from the two skills reported against persistence, compare it with the value computed directly from the RMSE column, and account for the small discrepancy between them. Which of the numbers you have computed would you put in a report, and why?

Exercise 3.3. A fleet's load at 17:00 follows the curve of Figure 3.2, with a balance temperature of 18°C and a sensitivity of 2 MW per °C above it. The reference day had 100 MW at 30°C. Compute the weather-adjusted baseline for a target day whose 17:00 temperature forecast is 15°C, using the curve. Then compute what equation (3.7) would give if the sensitivity were applied uniformly, and state the size and the sign of the resulting error.

Table 3.3: The objects of a baseline comparison.

Object	Definition	Section
persistence	$\hat{y}_{t+h} = y_t$; optimal for a random walk	3.2.1
seasonal naive	$\hat{y}_{t+h} = y_{t+h-24}$; available only for $h \leq 24$ h	3.2.2
weekly naive	$\hat{y}_{t+h} = y_{t+h-168}$; keeps day type, loses weather	3.2.3
similar-day average	mean of the same hour on K accepted same-type days	3.2.4
moving average	mean of the last N observations; lag $(N-1)/2$	3.2.5
weather-adjusted baseline	reference load plus $s(\hat{T}_{t+h\|t} - T_{\text{ref}})$	3.3.1
day matching	exclusion of dispatch, holiday and bad-data days, then weather ranking	3.3.2
weather and calendar regression	degree variables plus hour and day-type indicators	3.3.3
Skill	$1 - \text{RMSE}_{\text{model}}/\text{RMSE}_{\text{baseline}}$	3.5.1
baseline hierarchy	five rungs of increasing difficulty	3.5.4
ε_F	$\varepsilon_{\text{base}} - \varepsilon_{\text{ctrl}}$, with a usually positive covariance	3.6.2
amplification	relative baseline error enters F times P^{base}/F	3.6.3

Exercise 3.4. For the fleet of Section 3.6.3, the controlled-load prediction has an error standard deviation of 1 MW. Using equation (3.16), find the largest baseline-load error standard deviation that keeps the flexibility error standard deviation at or below 2 MW, first assuming the two errors are independent and then assuming a correlation of 0.9 between them. Compare the two answers and explain, in terms of Section 3.6.2, why building both models on the same temperature forecast helps.

Chapter 4

Feature engineering: lags, calendar, weather and state variables

4.1 The feature vector and the supervised table

Chapter 3 established that a model earns its complexity only by beating a credible simple alternative, and the simplest alternatives were built from one or two numbers: the last observation, the same hour yesterday, a temperature sensitivity. A model that is to do better needs more than one or two numbers, and it needs them arranged in a form a learning algorithm can consume. This chapter is about that arrangement, and its question is

Given everything genuinely known at the issue time, which finite vector of numbers should be handed to the model?

The raw material is a table of timestamps, meter readings and weather observations. No learning algorithm consumes a table of timestamps. The work of this chapter is to turn that table into a matrix of predictors and a vector of targets, with one row per issue time, in such a way that the row could have been constructed at the moment the forecast had to be issued and not one minute earlier.

Jargon: feature, feature vector, design matrix, lag, rolling statistic, one-hot encoding, degree day, as-of join

A *feature* is one number offered to the model as a predictor; the *feature vector* X_t collects the features for one issue time, and stacking the feature vectors of many issue times gives the *design matrix*. A *lag* is a past value of a series used as a feature, so P_{t-24} is the lag-24 feature of the load series. A *rolling statistic* is a summary of the last k observations, for example a 24-hour rolling mean. *One-hot encoding* replaces a categorical variable with one indicator column per category, so hour of day becomes 24 columns of which exactly one is 1. A *degree day* (here a degree *hour*) is the amount by which temperature exceeds or falls short of a reference, $\max(T - T^*, 0)$, accumulated over the period. An *as-of join* attaches to each row the most recent value of another table that had already arrived at that row's timestamp, rather than the value that table eventually holds.

Key idea

The feature vector X_t is the model's entire summary of the information set $\mathcal{I}_t$. Whatever is not in X_t the model cannot use, and whatever is in X_t the model will use — including, if the construction is careless, information that did not exist at time t.

This section sets up the object the rest of the chapter populates: the raw table, the map

from the information set to the feature vector, the supervised table that results, and what changes when several horizons are forecast at once.

4.1.1 The raw table is not yet a learning problem

Operational data arrive as a time-indexed table. Table 4.1 is three columns of it: a timestamp, the fleet load in megawatts, and the outdoor temperature in degrees Celsius. The first row records that at 08:00 the fleet drew 35 MW at 21 °C, the second that at 09:00 it drew 37 MW at 23 °C, and so on through the day.

Table 4.1: The raw operational table: one row per timestamp. The first two rows are the ones used throughout this chapter; the remainder are illustrative.

Timestamp	Load (MW)	Temperature (°C)
08:00	35	21
09:00	37	23
10:00	41	26
11:00	46	28
12:00	51	31

Three properties of this table block learning directly from it. It has no target column, because the quantity to be predicted lives in a later row rather than in the same row. It has no notion of a horizon, so nothing in it distinguishes a forecast made one hour ahead from one made twenty-four hours ahead. And its columns are not predictors of anything: a timestamp is a label, not a number with a meaning, and the load in a row is the quantity to be predicted at that row's time rather than a predictor of it. Feature engineering is the step that repairs all three: it manufactures a target column by shifting, fixes a horizon by construction, and replaces the timestamp with numbers that carry the calendar structure the timestamp only implies.

4.1.2 From the information set to the feature vector

Chapter 1 defined the information set $\mathcal{I}_t$ as everything genuinely available at issue time t: the metered load history, the weather observations and the weather forecasts that had been issued, the calendar, the device telemetry, and the record of the operator's own past actions. The information set is not a vector. It is a heterogeneous and growing collection, and a model cannot be fitted to it. Feature engineering is the choice of a map

$$X_t = \varphi(\mathcal{I}_t) \in \mathbb{R}^d, \qquad \hat{y}_{t+h} = f(X_t), \tag{4.1}$$

from that collection into a fixed-length real vector, after which the model f is fitted to the pairs (X_t, y_{t+h}). Two consequences follow immediately. The map φ is lossy, so a structure it discards is unavailable to f no matter how expressive f is; and the map is a modeling decision of the same weight as the choice of f, because the two compose. A linear f applied to a well-chosen φ routinely beats a flexible f applied to a poor one, which is the argument Chapter 5 makes in detail.

Figure 4.1 draws the composition. The left column lists the contents of the information set as they exist in storage; the middle column groups them into the five feature families this chapter works through; the right-hand box is the feature vector, which feeds the model and then the forecast. The arrow that does not appear in the figure is the one from the observed future, and its absence is the subject of Section 4.6.

A concrete instance of φ, and the one the rest of the chapter elaborates, is the eight-element vector

$$X_t = \big[\, P_{t-1},\ P_{t-2},\ P_{t-24},\ P_{t-168},\ T_t,\ T_{t-1},\ \mathrm{hr}_t,\ \mathrm{dow}_t \,\big], \tag{4.2}$$

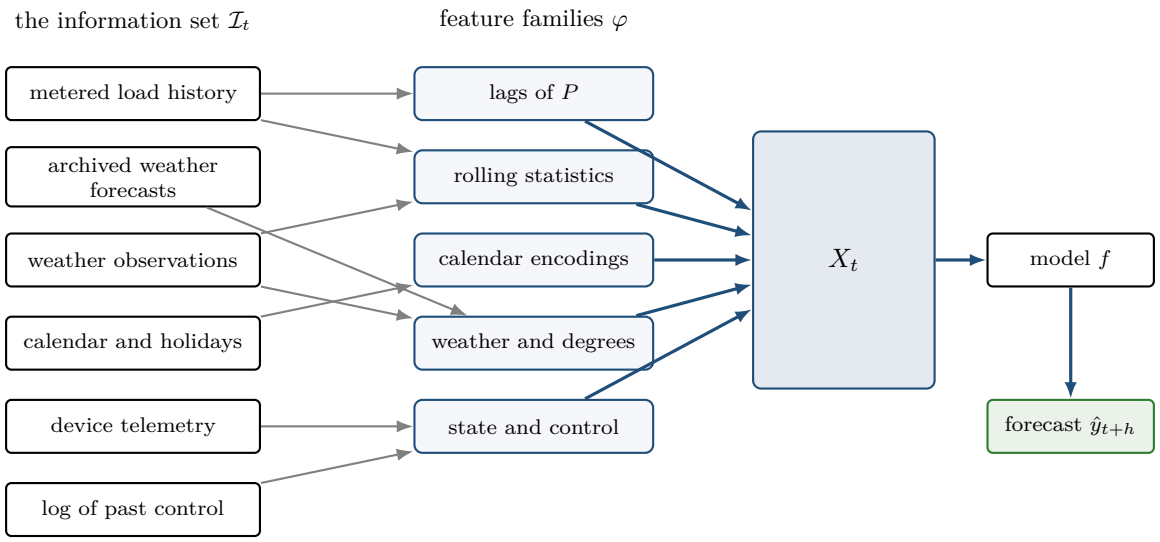


Figure 4.1: The composition of equation (4.1). The boxes on the left are the stored contents of the information set; the boxes in the middle are the five feature families of Sections 4.2 to 4.4; the shaded box is the feature vector, which is all the model ever sees. The map φ is lossy and is chosen by the modeler.

with P the fleet load, T the outdoor temperature, $\text{hr}_t \in \{0, \dots, 23\}$ the hour of day and $\text{dow}_t \in \{0, \dots, 6\}$ the day of week. Four lags of load carry the autocorrelation and the daily and weekly seasonality that Chapter 2 identified; two lags of temperature carry the immediate weather response and a little of the thermal memory; the two calendar entries carry the behavioral cycle. Every element is a number that existed at time t.

4.1.3 The supervised table

Applying φ at every issue time and pairing the result with the value h steps later produces the supervised table: rows indexed by issue time, feature columns on the left, one target column on the right. Figure 4.2 draws the construction for a short series and a horizon of two steps. The strip along the top is the raw series, one cell per time index. Each band below it is one row of the supervised table: the shaded group of cells on the left is the set of cells read into that row's features, the single cell on the right is that row's target, and the dashed arrow between them spans the horizon. The bands step one cell to the right as the issue time advances.

Written out, row t of the table is

$$\Big(\underbrace{\varphi(\mathcal{I}_t)}_{\text{features, known at } t} , \underbrace{y_{t+h}}_{\text{target, known at } t+h} \Big), \qquad t = t_{\min}, \dots, t_{\max}, \tag{4.3}$$

and the two ends of the range are set by the construction rather than by the data. The earliest usable issue time is the one at which the longest lag exists: with P_{t-168} in the feature set, the first seven days of any history produce no rows at all. The latest usable issue time is h steps before the end of the record, because later rows have no observed target. One series of N observations therefore yields about $N - 168 - h$ training rows, not N, and the count depends on both the longest lag and the horizon.

4.1.4 Several horizons at once

An operator rarely needs one horizon. The aggregator of Chapter 1 wants a correction for the next quarter hour, a plan for three hours ahead and a commitment for tomorrow evening, and equation (4.1) fixes a single h. Three constructions cover the general case.

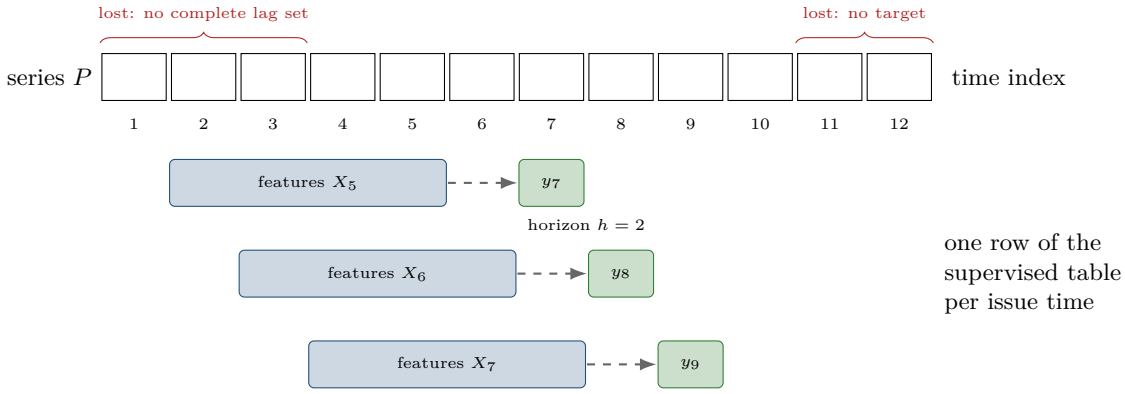


Figure 4.2: Unrolling a series into a supervised table. The top strip is the raw series; each band below is one row, with its feature cells shaded blue, its target cell shaded green and the horizon drawn as a dashed arrow. The braces mark the rows lost at each end: the first rows have no complete lag set, the last h rows have no target. The feature set shown is four consecutive lags, chosen to keep the drawing legible.

Direct, one model per horizon. Build one supervised table per horizon, each with its own target column y_{t+h}, and fit a separate f_h to each. The feature set may differ between them, which is the main attraction: at $h = 1$ the recent lags dominate, at $h = 24$ the weather forecast and the calendar dominate, and nothing forces the two models to share a predictor list. The cost is H models to fit and monitor.

Recursive, one model applied repeatedly. Fit a single one-step model and feed its own output back as the lag-1 feature of the next step. Only one model is needed, but errors compound along the recursion and the model is being applied to inputs it never saw in training, because at training time the lag-1 feature was an observation and at prediction time it is a forecast.

Horizon as a feature. Keep one model and add h itself as a column, so each issue time contributes H rows. The model can then share structure across horizons — the same temperature response applies at every h — at the price of assuming that a single functional form fits all horizons and of correlated rows, since the H rows of one issue time share their features.

Which construction is appropriate follows from how different the horizons are. In the direct scheme one fact is easy to overlook: as h grows, lags that were legal become uninformative rather than illegal. The lag P_{t-1} is always available at issue time t, but at $h = 24$ it is a value a full day before the target, and its weight in a fitted model falls accordingly. What changes with the horizon is the value of a feature, not its admissibility.

> **Principle 4.1.** *A feature is admissible if and only if its value existed and had arrived by the issue time t. Admissibility is a property of the feature and the issue time; usefulness is a property of the feature and the horizon. The two must not be confused.*

4.2 Lag and rolling features

Lags and rolling statistics are the features built from the target series itself. This section chooses which lags to include, states when a lag is available, and constructs the rolling summaries that compress a long history into a few numbers.

4.2.1 Lags at the orders that carry structure

The lag features of equation (4.2) are

$$P_{t-1}, \quad P_{t-2}, \quad P_{t-24}, \quad P_{t-168}, \tag{4.4}$$

and each order is there for a reason named in Chapter 2. Lags 1 and 2 carry the short-run persistence: the fleet's thermal state, the weather and the occupancy pattern all change slowly compared with an hour, so the last two observations locate the series' current level. Lag 24 carries the daily cycle: the same hour yesterday saw the same households on the same schedules under a similar weather regime. Lag 168 carries the weekly cycle, and does what lag 24 cannot, namely distinguish a Monday from a Sunday; for commercial and institutional load the weekly lag is frequently the stronger of the two, which is why Chapter 3 kept weekly persistence as a separate baseline.

The selection is not a matter of taste. The autocorrelation function of Chapter 2 shows peaks at 24, 48, 72 and 168 hours, and those peaks are the lag orders worth including; the partial autocorrelation function cuts off after a few lags, which says how many consecutive short lags add information once the earlier ones are present. A useful default for hourly load is the set $\{1, 2, 3, 24, 25, 48, 168\}$: three consecutive recent lags, the previous day's same and adjacent hour, the day before that, and the previous week. Lags 24 and 25 together let a model represent a small shift in the timing of the daily peak, which a single lag at 24 cannot.

Principle 4.2. *Lag selection follows the autocorrelation structure of the series, not convenience. Include the orders at which the autocorrelation function has peaks, and as many consecutive short lags as the partial autocorrelation function says are distinguishable.*

4.2.2 A lag must be available at the issue time

A lag of order k refers to the value at time $t - k$, which is in the past for every $k \geq 1$. Being in the past is necessary and not sufficient: the value must also have *arrived*. Interval meter data are commonly delivered on a delay, so at 14:00 the reading for 13:00 may not yet be in the database, and a feature set that assumes it is will work in a backtest and fail in production. Figure 4.3 shows the geometry. Time runs left to right with the issue time t marked by the solid vertical line and the target time $t + h$ to its right; the axis is broken twice so that lags 24 and 168 can be drawn alongside lags 1, 2 and 3. Each lag is drawn as an arrow pointing back from the issue time to the value it reads. Two regions are shaded. Everything to the right of t is inadmissible, because it has not happened. The narrow band immediately to the left of t is the latency window: those values exist but have not arrived, so they too are inadmissible at t, which in the figure rules out P_{t-1} and nothing else. Everything to the left of the latency window is admissible. The one arrow that crosses the issue time is the archived weather forecast $\hat{T}_{t+h|t}$, and it is legal because the object it reads — the forecast as issued at t — existed at t, even though the quantity it describes lies in the future.

The practical form of the rule is to build features against the *arrival* time rather than the nominal time. A feature set for a pipeline whose meter data land one hour late starts at P_{t-2}, and the backtest must do the same. Section 4.6.3 returns to this as one of the standard leakage mechanisms.

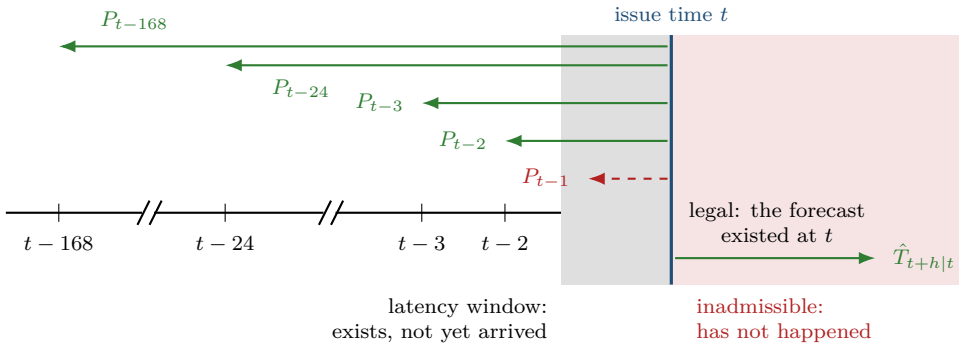


Figure 4.3: Which features are legal at the issue time. The time axis is broken twice so that the long lags fit; the solid vertical line is the issue time. Green arrows read values that exist and have arrived; the red dashed arrow reads a value that lies inside the metering latency window and is therefore not available at t even though it is in the past. The latency window is drawn one hour wide, which is illustrative; its true width is a property of the data pipeline.

4.2.3 Rolling statistics

Individual lags are noisy. A rolling statistic averages the noise away while retaining the information: over a trailing window of k observations,

$$\bar{P}_t^{(k)} = \frac{1}{k}\sum_{i=0}^{k-1} P_{t-i}, \qquad P_t^{\max,(k)} = \max_{0\le i<k} P_{t-i}, \qquad P_t^{\min,(k)} = \min_{0\le i<k} P_{t-i}, \tag{4.5}$$

together with the rolling standard deviation. The window must be trailing: it ends at t and never extends past it.

The three statistics describe different things. Figure 4.4 plots a week of synthetic hourly fleet load in the upper panel, with its 24-hour rolling mean and 24-hour rolling maximum drawn over it, and the driving temperature in the lower panel with its own 24-hour rolling maximum. In the upper panel the raw load oscillates through about 45 MW peak to trough every day; the rolling mean removes that oscillation almost completely and leaves a slowly varying level, which rises through the middle of the week and falls over the weekend; the rolling maximum traces the envelope of the daily peaks. A model given the rolling mean is given the level without the cycle, which is useful precisely because the cycle is already supplied by the calendar features of Section 4.3. The rolling maximum answers a different question, namely how extreme the recent past was, and that is the question a thermal system cares about.

The lower panel makes the point physically. The 24-hour rolling maximum of outdoor temperature is a proxy for stored building heat: a fleet whose buildings have seen 36 °C at some point in the last day carries heat in its walls and floors regardless of the temperature at the issue time, and it will draw more cooling power at 22 °C than an identical fleet that has been at 22 °C all day. That memory is the *RC* behavior of Chapter 2, and a rolling maximum is a crude but effective way to hand a memoryless model a summary of it. The same logic motivates rolling sums of degree hours over 6, 24 and 72 hours.

4.2.4 Differences, ratios and slopes

A third family of features built from the series is made of differences. The seasonal difference and the recent slope,

$$\Delta_{24}P_t = P_t - P_{t-24}, \qquad \Delta_{168}P_t = P_t - P_{t-168}, \qquad g_t = \tfrac{1}{3}\,(P_t - P_{t-3}), \tag{4.6}$$

say how today differs from yesterday, how this week differs from last week, and in which direction the load is currently moving. These are exactly the differencing operations that Chapter 6 uses

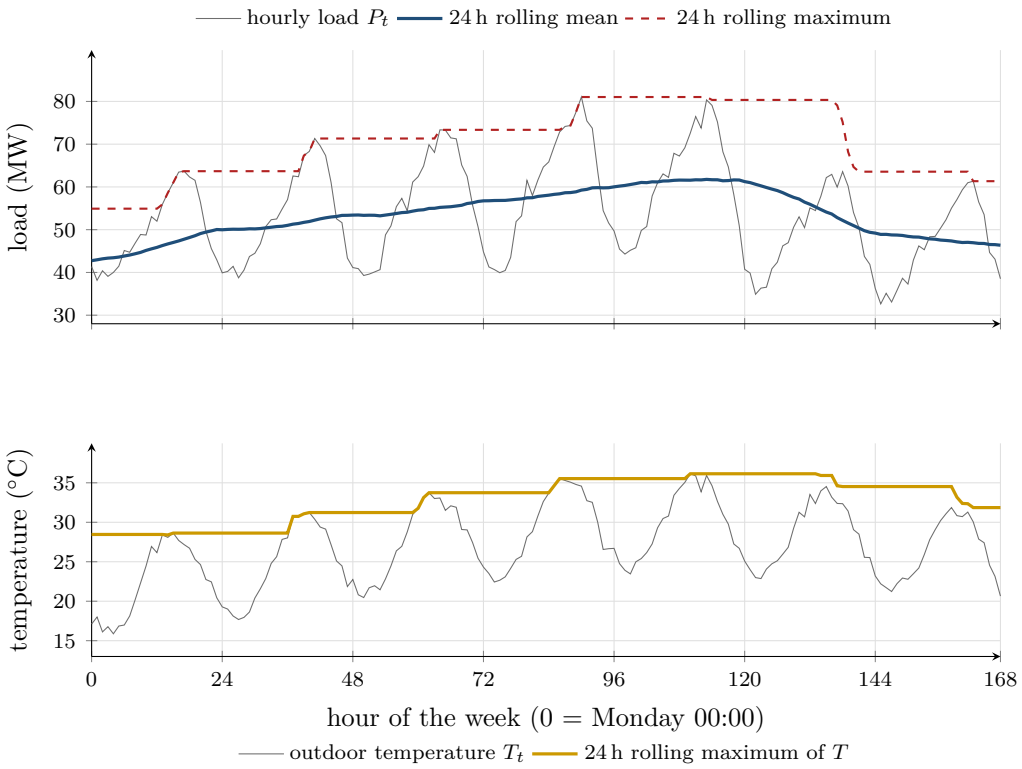


Figure 4.4: Rolling statistics on one synthetic week of hourly data, generated by `scripts/ch04_data.py` (seed 4) from $T_t = 26 + 6\sin[2\pi(t-9)/24] + 4\sin[2\pi(t-120)/216] + \nu_t$ and $P_t = w_d\,[\,40 + 2.0\,\mathrm{CD}_t + 14\,s(\mathrm{hr}_t)\,] + \varepsilon_t$, with $\mathrm{CD}_t = \max(T_t - 22, 0)$, $s(\mathrm{hr}) = \exp[-(\mathrm{hr}-18)^2/18] + \frac{1}{2}\exp[-(\mathrm{hr}-8)^2/8]$, $w_d = 1$ on weekdays and 0.85 at weekends, $\nu_t \sim \mathcal{N}(0, 0.6^2)$ and $\varepsilon_t \sim \mathcal{N}(0, 1.2^2)$. Values are illustrative. Upper panel: load with its 24-hour rolling mean and maximum. Lower panel: temperature with its 24-hour rolling maximum, the stored-heat proxy of Section 4.2.3.

to make a series stationary, used here as features rather than as a transformation of the target. Their value is that they are robust to slow drift: if enrollment grows or a tariff changes the level of the series, the differences are unaffected while the raw lags all shift together.

Ratios serve the same purpose multiplicatively, for example $P_t / \bar{P}_t^{(168)}$, the load relative to its weekly average, which is dimensionless and comparable across fleets of different size. Features of this kind are the main reason a model fitted on one fleet sometimes transfers to another.

4.2.5 The cost of adding lags

Lag features are cheap to compute and not free to include. Three costs recur. Successive lags are strongly correlated with one another — P_{t-1}, P_{t-2} and $\bar{P}_t^{(3)}$ carry nearly the same information — so a least-squares fit distributes weight among them arbitrarily and the fitted coefficients become unstable; this is the collinearity that motivates ridge regression in Chapter 5. Every lag order k added to the feature set destroys the first k rows of the table, which matters when the history is short. And a lag that is available in the backtest but unavailable in production, by the latency argument of Section 4.2.2, converts an apparent improvement into a deployment failure. The remedy for the first is regularization or feature selection, for the second a deliberate choice of the longest lag, and for the third the arrival-time discipline of Section 4.6.

4.3 Calendar and weather features

Calendar and weather features carry the structure that the series' own past cannot supply: the behavioral cycle imposed by clocks and schedules, and the physical driver that sets the level of thermal load. This section encodes the calendar, builds the degree variables that linearize the weather response, and states the rule governing weather features for the target time.

4.3.1 Calendar variables and one-hot encoding

The timestamp implies a set of integers: hour of day, day of week, day of year, month, week of year. None of them is a quantity. Hour 14 is not twice hour 7 in any sense that matters to load, so passing the raw integer to a linear model asserts a linear relationship between the clock and the load that does not exist. The standard repair is one-hot encoding: replace the integer by a vector of indicators,

$$\mathrm{hr}_t \longmapsto \big[\, \mathbf{1}\{\mathrm{hr}_t = 0\},\ \mathbf{1}\{\mathrm{hr}_t = 1\},\ \ldots,\ \mathbf{1}\{\mathrm{hr}_t = 23\} \,\big], \tag{4.7}$$

of which exactly one entry is 1. A linear model then fits one coefficient per hour, which is a free-form daily profile. Two details matter in practice. Only 23 of the 24 columns are free once an intercept is present, because the 24 indicators sum to one; dropping one column, or omitting the intercept, avoids an exactly singular design matrix. And 24 hour columns times 7 weekday columns is 168 separate coefficients if the interaction is encoded as a full product, which is usually more parameters than a short history can support — the common compromise is 24 hour columns plus a single weekend indicator, or 24 columns per day *type* with three types (weekday, weekend, holiday).

Holidays and schedules. A public holiday has the load shape of a Sunday on a day the weekday indicator calls a Tuesday, and a model with no holiday feature will be wrong on every one of them. The usual encoding is an indicator per holiday, or a single indicator plus indicators for the day before and the day after, since the shoulder days also differ. School calendars, local industrial shutdowns and major sporting events act the same way. These features come from a table maintained by hand and are a common source of silent error when the table runs out.

Clocks and time zones. Load responds to local time, because people and thermostats follow local clocks, while weather and power systems are usually recorded in Coordinated Universal Time. The practical convention is to store every timestamp in UTC and to derive the calendar features from the local-time rendering of it. Daylight saving transitions then produce one 23-hour day and one 25-hour day per year, in which a naive lag-24 feature points at the wrong hour of the previous day. The transitions are two days out of 365 and are frequently left unhandled; the resulting errors are concentrated in exactly those days and are easy to mistake for model inadequacy.

4.3.2 Cyclic encoding

One-hot encoding is expensive in columns and discards the fact that adjacent hours behave similarly. The alternative exploits the geometry of a clock. Hour of day is a point on a circle, and the pair

$$\mathrm{hr}_t \longmapsto \left(\cos\frac{2\pi\,\mathrm{hr}_t}{24},\ \sin\frac{2\pi\,\mathrm{hr}_t}{24}\right) \tag{4.8}$$

places that point in two dimensions, in which hours 23 and 0 are neighbors. Figure 4.5 compares the two encodings. The left panel is the unit circle of equation (4.8), with the 24 hours marked as points, a few of them labeled, and the horizontal and vertical coordinates labeled with the cosine and sine columns; the right panel is the raw integer encoding drawn as a number line from 0 to 23. In the left panel the chord from hour 23 to hour 0 has the same length as the chord from hour 0 to hour 1, both marked; in the right panel hour 23 and hour 0 are 23 units apart, the largest separation in the whole encoding, while hour 0 and hour 1 are one unit apart. The integer encoding therefore tells a model that midnight is maximally distant from the hour before it, which is false, and the cyclic pair tells it that they are adjacent, which is true.

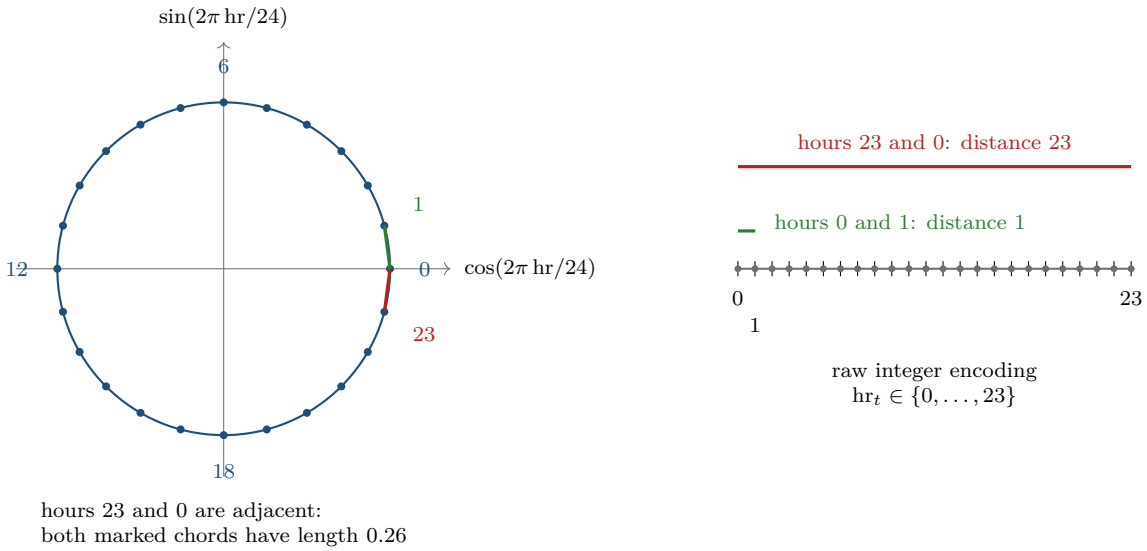


Figure 4.5: Two encodings of hour of day. Left: the cyclic pair of equation (4.8), with the 24 hours as points on the unit circle; the two marked chords, from hour 23 to hour 0 and from hour 0 to hour 1, both have length $2\sin(\pi/24) = 0.26$. Right: the raw integer encoding on a number line, in which the same two pairs of hours are 23 and 1 units apart. The cyclic encoding represents the wrap-around that the integer encoding breaks.

The same construction applies to any periodic index. Day of year becomes the pair $\cos(2\pi d/365)$ and $\sin(2\pi d/365)$, which gives a smooth annual seasonality in two columns instead of twelve month indicators, and day of week becomes a pair at period 7 when a smooth weekly shape is wanted.

Three qualifications bound the usefulness of the encoding. The pair represents only the first harmonic, so it describes a single smooth rise and fall per cycle and cannot represent the two-peaked daily profile of a residential fleet; adding the pair at double frequency, $\cos(4\pi\,\mathrm{hr}/24)$ and $\sin(4\pi\,\mathrm{hr}/24)$, recovers the second harmonic, and a few harmonics approximate any smooth profile. Where the profile is not smooth — a sharp switch at a tariff boundary, a loading-dock schedule — one-hot encoding represents it and the cyclic pair does not. And a tree-based model needs neither: a tree splits on thresholds of the raw integer, so it can isolate hours 17 to 20 with two splits and is indifferent to the fact that 23 and 0 are numerically far apart. Cyclic encoding is a device for models that are linear or smooth in their inputs, which includes linear regression and neural networks and excludes decision trees.

4.3.3 Degree variables and the shape of the weather response

Outdoor temperature is the dominant exogenous driver of thermal load, and the relationship between them is not linear. Below a balance temperature neither heating nor cooling runs and load is flat in temperature; above it, cooling load rises roughly linearly; below a lower balance temperature, heating load rises as temperature falls. The standard features that encode this shape are the degree variables

$$\mathrm{CD}_t = \max(T_t - T_c^*,\, 0), \qquad \mathrm{HD}_t = \max(T_h^* - T_t,\, 0), \tag{4.9}$$

cooling degrees and heating degrees about base temperatures T_c^* and T_h^*. Typical bases for residential load are near 18 °C for heating and 20 to 22 °C for cooling, and the right value for a particular fleet is estimated by fitting the kink location rather than assumed.

Figure 4.6 plots the relationship. The horizontal axis is outdoor temperature, the vertical axis is fleet load, the solid curve is the piecewise-linear response built from equation (4.9), and the dashed line is what a single linear term in temperature can represent over the same range. The solid curve is flat between the two base temperatures, the shaded band marks that dead zone, the two sloped arms are the heating and cooling regimes, and the vertical measure at −5 °C is the distance between the two. The dashed line passes through the middle of the range and is wrong everywhere it matters: it understates load at −5 °C by 40 MW, which is the measure drawn in the figure, overstates it at 19 °C by 12 MW, and understates the load at 40 °C by 17 MW. A linear model given T_t cannot represent the kink; a linear model given CD_t and HD_t represents it exactly, because the kink has been supplied by the feature rather than learned by the model.

Three refinements are routine. Degree variables can be squared, CD_t^2, to capture the saturation that occurs when every available compressor is already running. They can be accumulated, as rolling sums of degree hours over 6, 24 and 72 hours, which restores the thermal memory that the instantaneous value lacks. And they can be interacted with the hour of day, because a cooling degree at 03:00 and a cooling degree at 17:00 produce different load; the interaction $\mathrm{CD}_t \times \mathbf{1}\{\text{evening}\}$ is the kind of term Chapter 5 shows a linear model needs stated explicitly and a tree discovers on its own.

4.3.4 Humidity, irradiance and the rest of the weather

Temperature is not the only weather variable that moves thermal load. Relative humidity RH_t matters because an air conditioner removes latent as well as sensible heat, so the same dry-bulb temperature at 80 % humidity produces a longer compressor runtime than at 30 %; the physically motivated feature is a combined index such as wet-bulb temperature, the dew point, or an apparent-temperature formula, and the practical alternative is to include RH_t and the interaction $\mathrm{CD}_t \times \mathrm{RH}_t$. Solar irradiance drives both the solar heat gain through windows, which adds cooling load with a lag of an hour or two, and the output of behind-the-meter photovoltaic

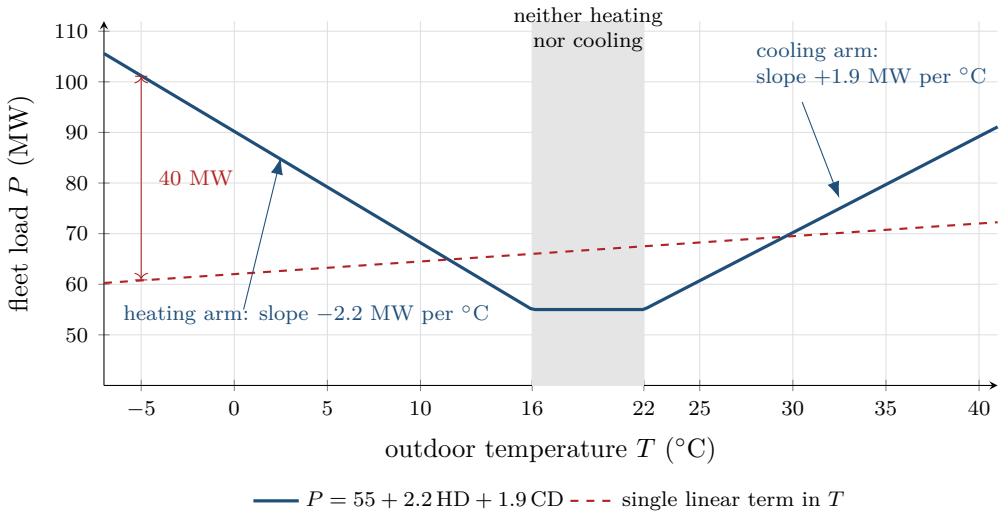


Figure 4.6: The hockey-stick load–temperature relationship and why a linear model needs the kink supplied. The solid curve is $P = 55 + 2.2\,\mathrm{HD} + 1.9\,\mathrm{CD}$ in MW, with $\mathrm{HD} = \max(16 - T, 0)$ and $\mathrm{CD} = \max(T - 22, 0)$ in degrees Celsius; the shaded band is the dead zone between the two base temperatures; the dashed line is $P = 62 + 0.25\,T$, representative of what one linear term in T can do over this range; the vertical measure at $-5\,^{\circ}\mathrm{C}$ is the 40 MW distance between them. All values are illustrative.

systems, which subtracts from metered load at the same instant; a fleet with significant rooftop generation needs irradiance as a feature of its own sign. Wind speed increases the building envelope's heat loss and therefore matters in winter. Cloud cover is a coarse substitute for irradiance when irradiance is unavailable.

The common error with this family is to add variables faster than the history can support them. A year of hourly data is about 8 760 rows; a feature set with four lags, three rolling statistics, 24 hour indicators, a weekday indicator, a holiday indicator and six weather variables with four interactions is already 44 columns, and every additional interaction multiplies. Section 4.5.3 takes up the selection question.

4.3.5 Weather features must be forecasts, not observations

The weather feature for the target time is the single most consequential choice in the chapter. A model that predicts load at 17:00 from the temperature at 17:00 is using a number that at the 14:00 issue time does not exist. The admissible feature is the forecast as issued, $\hat{T}_{t+h|t}$, read from the archive of forecasts rather than from the archive of observations.

Table 4.2 shows the same training row built both ways. The first column names the field, the second gives the value if the row is built from observations, the third the value if it is built from the forecast that existed at the issue time, and the last states whether the value was available at 14:00. The two builds differ by about two degrees Celsius in temperature and, through equation (4.9) with $T_c^* = 22\,^{\circ}\mathrm{C}$, by about two cooling degrees; at a sensitivity near 2 MW per cooling degree that is about 4 MW of load, which is a large fraction of a typical forecast error. The first build is not merely optimistic. It trains the model on a relationship that cannot be evaluated in production, so the coefficient it learns on temperature is the sensitivity of load to the *true* temperature, whereas the quantity available at run time is a forecast with its own error. Fitting to the as-issued column makes the model learn the sensitivity of load to the forecast, which is the sensitivity it will need.

The construction this requires is an archive of forecasts, keyed by both the issue time and the target time, and an as-of join against it. Three practical points follow. The archive must be kept,

Table 4.2: One training row for the 17:00 target, built from observations and from the forecast as issued at 14:00. Values are illustrative; cooling degrees use $T_c^* = 22\,^\circ\mathrm{C}$.

Field	Observed build	As-issued build	Available at 14:00?
Temperature at 17:00 (°C)	34.1	36.0	only the as-issued value
Cooling degrees at 17:00 (°C)	12.1	14.0	only the as-issued value
Implied load contribution (MW)	24.2	28.0	–
Source table	observations	forecast archive	–

because a weather provider's observation history is easy to obtain later and its forecast history is not; if no archive exists, a defensible fallback is to degrade the observations deliberately by adding noise with the provider's documented error statistics, and to say so. The forecast's own accuracy depends on its lead time, so the feature for $h = 1$ and the feature for $h = 36$ have different error characteristics even though both are called "forecast temperature". And weather forecast error is the irreducible floor of a weather-driven load forecast: no load model can be more certain about 17:00 than the weather input it was given.

4.4 State and control features for a controllable fleet

For a fleet the aggregator can dispatch, lags, calendar and weather are not enough. This section adds the features that describe what the devices are currently doing, how many of them can be reached, and what the operator has recently done to them.

4.4.1 Features as observable proxies for a latent state

Chapter 2 framed the fleet as a state-space system,

$$x_{t+1} = g(x_t, u_t, w_t), \qquad P_t = h(x_t), \tag{4.10}$$

in which x_t contains quantities that are not metered: the distribution of indoor temperatures across the fleet, the thermal energy stored in building mass, the compressor on-off pattern. If x_t were observable, forecasting would be mostly simulation. It is not, and the practical substitute is to give the model features that are correlated with the components of x_t and are measurable. A lag of load is such a proxy, because load is a function of the state; so is the rolling maximum of temperature, because it bounds the stored heat; so is the duty cycle, because it reports the fraction of the fleet currently drawing power. Seen this way, the whole chapter is an exercise in state reconstruction with a fixed, hand-chosen basis, and Chapter 7 replaces the hand-chosen basis with a learned one.

4.4.2 The fleet feature vector

For a thermostat fleet forecast at horizon h, a defensible feature vector is

$$X_t = \big[\, P_t,\ P_{t-1},\ P_{t-24},\ \hat{T}_{t+h|t},\ \mathrm{RH}_{t+h|t},\ N_{\text{avail}},\ D_t,\ \Delta T_t,\ r_{\text{hist}} \,\big], \tag{4.11}$$

with $\Delta T_t = T_{\text{in}} - T_{\text{set}}$ the average gap between indoor temperature and setpoint across the fleet and r_{hist} a summary of how the fleet has responded to past dispatch. The first three elements are the load lags of Section 4.2.1, the next two are the as-issued weather of Section 4.3.5, and the last four are new: they describe the fleet rather than the series. The remaining subsections take them in turn, and Table 4.3 collects all five families with their roles.

Table 4.3: The five feature families, what structure each supplies, and which chapters use them.

Family	Examples	Structure it supplies	Defined in
Lags	P_{t-1}, P_{t-2}, P_{t-24}, P_{t-168}	autocorrelation; daily and weekly seasonality	4.2.1
Rolling	24 h mean, max, min, standard deviation; degree-hour sums	recent level, recent extremes, thermal memory	4.2.3
Calendar	hour indicators or cyclic pair, weekday, holiday, day of year	the behavioral cycle imposed by clocks and schedules	4.3.1, 4.3.2
Weather	$\hat{T}_{t+h\|t}$, CD, HD, RH, irradiance, wind	the physical driver of thermal load, with its kink	4.3.3, 4.3.5
State and control	N_{avail}, D_t, ΔT_t, time since last event, r_{hist}	the fleet's internal condition and the operator's own actions	4.4.3 to 4.4.5

4.4.3 Availability and fleet composition

The number of devices available, N_{avail}, is the count of devices that are enrolled, powered, communicating and not opted out. It is a feature and not a constant: enrollment grows month over month, communication reliability varies with the cellular network, and customers opt out of events. A model fitted without it will attribute a step change in fleet load to whatever else moved at the same time, typically the weather, and will carry that error forward.

Two constructions make N_{avail} useful. The first is to normalize: a per-device target P_t/N_{avail} removes enrollment growth from the series entirely, so the model learns a device-level relationship and the fleet forecast is recovered by multiplying by the forecast device count. The second is to split the count by cohort — by climate zone, by equipment type, by tariff — because a fleet whose composition shifts behaves differently even at constant total count. Related composition features include the fraction of devices with a given thermostat model, the fraction on a time-of-use tariff, and the mean age of enrollment.

4.4.4 Duty cycle and the setpoint gap

The duty cycle D_t is the fraction of the fleet's compressors running at time t. It is the most direct observable summary of the fleet's thermal state: a duty cycle of 0.3 on a mild morning and a duty cycle of 0.9 in the afternoon of a heat wave describe fleets whose response to a curtailment command differs by a factor of three. For the load forecast it tells the model how much of the fleet's nameplate capacity is currently in use; for a flexibility forecast it is close to being the answer.

The setpoint gap $\Delta T_t = T_{\text{in}} - T_{\text{set}}$ measures headroom. A fleet whose indoor temperatures sit at the setpoint has the full comfort band available and can absorb a curtailment; a fleet whose indoor temperatures are already two degrees above the setpoint, because the last event depleted them or because the equipment is undersized for the weather, cannot. The gap is therefore the state variable that distinguishes two fleets with identical load and very different capability, which is the distinction Chapter 11 develops. Both features require telemetry: a fleet whose thermostats report only aggregate energy has neither, and the forecaster must then infer them from the aggregate it does report.

4.4.5 Control history and rebound

Features that describe what the operator recently did are necessary because the fleet's response depends on its own recent history. The usual set is the time since the last dispatch event, the cumulative duration of events in the last 24 hours, an indicator for the rebound window following an event, and the historical response ratio

$$r_{\text{hist}} = \frac{\text{verified response}}{\text{commanded response}}, \tag{4.12}$$

the fraction of the commanded reduction the fleet has actually delivered in comparable conditions. The rebound indicator earns its place because the hours after a curtailment carry anomalously high load: the buildings that were allowed to warm must be cooled again, and compressors that would normally be cycling run continuously. A model without a rebound feature treats those hours as unexplained error, and because they follow the operator's own actions, the error is systematic rather than random.

These features also make the target partly endogenous. The control variable u of Chapter 1 enters the feature vector, so the forecast is conditional on a dispatch history and, if the planned action is also included, on a dispatch plan. The practical consequence for the feature table is that the control log must be joined into it with the same care as the weather archive: what matters is the command that was actually issued at the time, and the record of what was commanded must not be overwritten by a later correction.

4.5 Target transformations and feature selection

The target column is as much a modeling choice as the feature columns, and once the feature list is long it has to be pruned. This section transforms the target so that the model learns a correction rather than a level, and then chooses which features to keep.

4.5.1 Forecasting a change rather than a level

Nothing requires the target column to hold y_{t+h} itself. Two alternatives are standard:

$$z_{t+h} = y_{t+h} - y_t \qquad \text{or} \qquad z_{t+h} = y_{t+h} - y_{t+h-24}, \tag{4.13}$$

the change from the current value, and the change from the same hour yesterday. The forecast is recovered by adding the subtracted term back, $\hat{y}_{t+h} = y_t + \hat{z}_{t+h}$. In this form the model is predicting the error of a baseline from Chapter 3: the first target makes it predict the error of persistence, the second the error of seasonal persistence. Three things change. The baseline is obtained for free and cannot be beaten by less than nothing, since a model that predicts $\hat{z} = 0$ reproduces the baseline exactly. The target is usually much closer to stationary, because the level and most of the seasonality have been removed, which is the differencing idea of Chapter 6 applied to the target column. And a slow drift in the level of the series — enrollment growth, a tariff change — no longer appears in the target at all.

The transformation is not free of cost. A level forecast with error σ and a change forecast with error σ are not equally good, because the change target has a smaller variance to begin with, so the skill score of Chapter 3 must be recomputed against the same baseline in the same units before the two are compared. For a linear model the change target is close to equivalent to including y_t among the features with its coefficient fixed at one, so the choice is really about whether that coefficient should be fitted or imposed.

4.5.2 Normalizing the target

Two further transformations are common. Dividing by the device count,

$$\tilde{y}_{t+h} = \frac{y_{t+h}}{N_{\text{avail}}}, \tag{4.14}$$

turns a fleet target into a per-device target and removes enrollment growth, as Section 4.4.3 noted; the fleet forecast is then $\hat{P}_{t+h} = \hat{N}_{\text{avail},t+h}\,\hat{\tilde{y}}_{t+h}$, which makes the dependence on the device-count forecast explicit instead of hiding it. Taking logarithms, $\log y_{t+h}$, converts multiplicative effects into additive ones and is appropriate when the error scales with the level, which it often does for load. The log transformation carries a trap: the exponential of the mean of the log is not the mean of the level, so back-transforming a log-scale forecast gives a median rather than a mean and underestimates the expectation by a factor that depends on the residual variance. If the decision downstream needs an expectation, the correction must be applied; if it needs a median or a quantile, as the commitment problems of Chapter 10 often do, no correction is needed.

4.5.3 Choosing which features to keep

A feature list assembled from the five families of Table 4.3 runs to tens of columns, many of them near-duplicates. Four selection methods cover most practice. Lasso regression, the ℓ_1 penalty of Chapter 5, drives coefficients of redundant features exactly to zero and so performs selection as part of fitting. Permutation importance measures how much validation error increases when one column is randomly shuffled, which is a direct estimate of how much that column was contributing. A tree ensemble reports a built-in importance, the total reduction in the splitting criterion attributable to each feature. And forward selection adds features one at a time, keeping each only if it improves rolling-origin validation error, which is the protocol of Chapter 9.

Figure 4.7 shows an importance ranking of the kind these methods produce. The horizontal axis is the share of total importance and the bars are features, ordered from most to least important; the colors group the bars by the family each feature belongs to. The ranking in the figure is illustrative, but its shape is typical: a few lag and weather features account for most of the total, the calendar and fleet-state features contribute moderately, and a long tail of features contributes almost nothing.

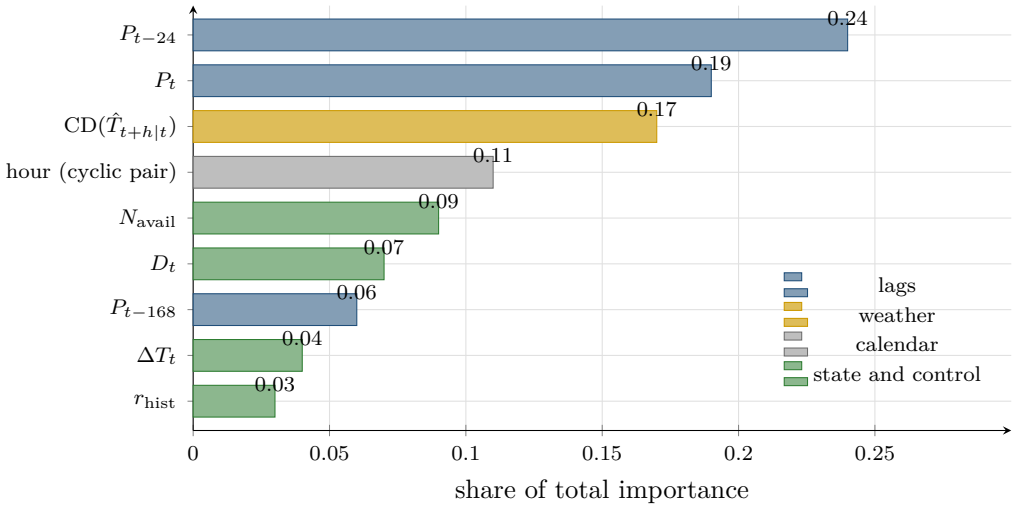


Figure 4.7: An illustrative feature-importance ranking for a thermostat fleet forecast at $h = 3$ hours, with the nine features of equation (4.11). Bars are shares of total importance and sum to one; colors group features by the families of Table 4.3. The values are illustrative and are not measurements.

Two cautions apply to any such ranking. Importance is not causality: a feature can rank highly because it is correlated with the true driver, and removing the true driver can leave the proxy's importance intact. And correlated features split their importance between them, so two near-duplicate lags each at 0.08 may be jointly worth more than a single feature at 0.14; dropping the lower-ranked member of a correlated pair is safe, dropping both is not.

4.5.4 How many features the history supports

The row count of Section 4.1.3 bounds the column count. Two years of hourly data give about 17 500 rows before truncation and about 17 300 after a 168-hour longest lag, which supports a few tens of features comfortably and a few hundred only with regularization. The binding constraint is usually not the nominal row count but the number of *effective* observations: consecutive hourly rows are strongly correlated, so 17 300 hourly rows carry nothing like 17 300 independent observations about an annual effect, and a feature that varies only with the season — a month indicator, say — has really been observed twice. This is the reason a seasonal feature set should be validated on held-out seasons rather than on held-out hours, which is the argument Chapter 9 makes in full.

4.6 Leakage, latency and feature hygiene

Every rule in this chapter reduces to one: a feature may depend only on what existed at the issue time. This section states the rule precisely, enumerates the mechanisms by which it is violated, and describes the artifact that makes the rule enforceable.

4.6.1 Two clocks, and the admissibility rule

Every datum in an operational store has two timestamps. Its *valid time* is the instant the datum describes: the hour a meter reading covers, the target hour a weather forecast refers to. Its *arrival time* is the instant the datum became available in the store. For an observation the arrival time is later than the valid time, by the metering latency. For a forecast the arrival time is *earlier* than the valid time, which is the whole point of a forecast. Confusing the two is the root of every leakage mechanism in Section 4.6.2. The rule is

$$X_t = \varphi\big(\{\, d \in \mathcal{D} : \mathrm{arrival}(d) \le t \,\}\big), \tag{4.15}$$

that the feature vector is a function of the data whose arrival time is at or before the issue time, with no condition at all on valid time. Equation (4.15) admits the forecast $\hat{T}_{t+h|t}$, whose valid time is in the future and whose arrival time is in the past, and excludes the observation T_{t+h}, whose valid time and arrival time are both in the future. It also excludes the observation T_{t-1} when the metering latency exceeds one hour, which equation (4.2) quietly assumed away.

> **Principle 4.3.** *Build features against arrival time, not valid time. A feature is admissible when every datum it reads had arrived by the issue time; the period a datum describes is irrelevant to its admissibility.*

4.6.2 Five leakage mechanisms

Leakage is a violation of equation (4.15), and in practice it arrives by one of five routes. Table 4.4 names them, with the symptom each produces and the fix. The common symptom is a validation error that is too good and that does not survive deployment, and the common cause is a table join that silently used the latest available value of something instead of the value available at the row's issue time.

Table 4.4: Five mechanisms by which equation (4.15) is violated.

Mechanism	What goes wrong	Symptom	Fix
Observed weather	T_{t+h} used where $\hat{T}_{t+h\|t}$ was the only available value	validation error far below operational error	as-of join against a forecast archive
Centered window	a rolling statistic whose window extends past t	short horizons look almost perfect	trailing windows only
Global statistics	the scaler's mean and standard deviation computed over the whole data set, test period included	small, uniform optimism; worse with few rows	fit the scaler on the training fold alone
Target-derived features	a feature computed from the target day, such as that day's peak or daily mean	near-zero error on the leaked target	derive features only from data before t
Revised data	meter readings or forecasts later corrected in place, so the store no longer holds what was available	silent; the backtest cannot be reproduced	append-only storage with arrival timestamps

Figure 4.8 draws the first three on a timeline. Time runs left to right, the issue time t is the solid vertical line, the target time $t + h$ is marked, and each horizontal bar is the span of data one feature reads. The top bar is a legal trailing 24-hour rolling mean, which ends at the issue time. Below it, a centered 24-hour window reaches twelve hours past the issue time and is therefore illegal; an observed temperature at the target time sits entirely in the future and is illegal; and a normalization computed over the full data set spans everything, including data from after the target, and is illegal in a subtler way, since it leaks only summary statistics rather than individual values. The vertical line is the only thing that distinguishes the legal bar from the illegal ones, and it is exactly the line a careless table join erases.

4.6.3 Metering latency and missing lags

Two operational details remain, and both change the row set rather than the column set. The first is the latency of Section 4.2.2: a feature set must be built at the latency the production pipeline actually delivers, which means the backtest has to simulate that latency rather than assume instant data. The practical device is to make the latency a parameter of the feature-building function and to run the backtest at the production value, so that a change in the pipeline is a change in one number.

The second is the handling of missing lags. A row whose features are incomplete is usually dropped, and dropping it is not neutral: the set of dropped rows depends on the longest lag and on the horizon, so two models compared at different horizons may have been fitted and evaluated on different periods. If one horizon drops the rows after an outage and another does not, the comparison between them is contaminated. The disciplined practice is to fix one row set — the intersection of the rows available at every horizon under comparison — and to evaluate every model on it, and to record how many rows were dropped and why. Imputation is permitted and must itself be causal: a gap may be filled from values before it, never by

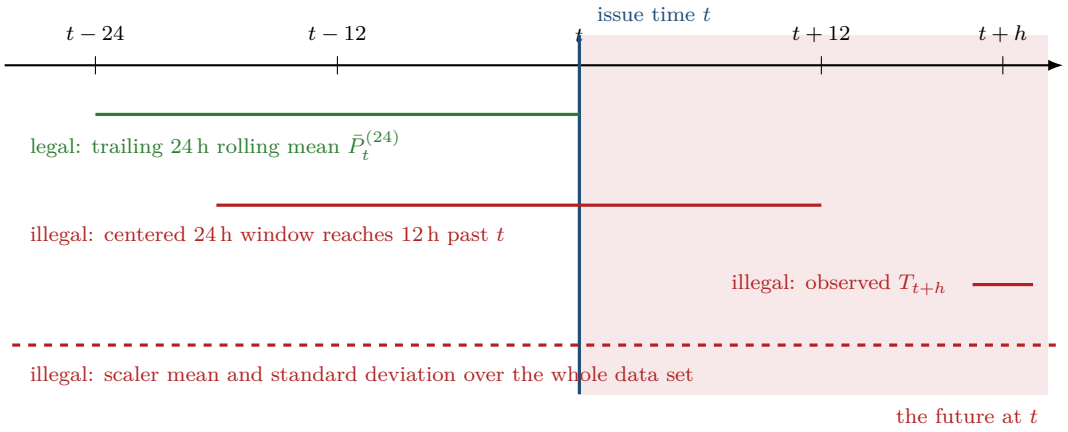


Figure 4.8: Three leakage mechanisms on a timeline. Each bar is the span of data one feature reads; the solid vertical line is the issue time and the shaded region is the future relative to it. Only the top bar ends at or before the issue time, so only the top bar is admissible under equation (4.15).

interpolating across it, because interpolation reads the future.

4.6.4 The feature table as a contract

The artifact that makes all of this enforceable is a written schema for the feature table. Table 4.5 is an excerpt: each row is one feature, with its source table, the valid-time offset relative to the issue time, the arrival latency, the units, and the value to use when the source is missing. A schema in this form answers the questions that otherwise get answered by reading code. Which features are forecasts, and therefore carry a weather provider's error? Which features would be unavailable if the meter feed were delayed by another hour? What is the units convention, so that a megawatt column and a kilowatt column are not summed?

Table 4.5: Excerpt of a feature-table contract. The offset is the valid time relative to the issue time t; the latency is how long after the valid time the datum arrives. Negative offsets are past values; positive offsets are necessarily forecasts.

Feature	Source	Offset	Latency	Units	If missing
`load_lag2`	meter feed	-2 h	1 h	MW	last known value
`load_roll24_mean`	meter feed	$[-25, -2]$ h	1 h	MW	drop the row
`cd_fcst`	forecast archive	$+h$	issued at t	°C	previous issue
`hour_sin`, `hour_cos`	calendar	$+h$	none	–	never missing
`n_available`	enrollment store	0 h	15 min	count	last known value
`duty_cycle`	device telemetry	0 h	5 min	–	cohort median
`mins_since_event`	control log	0 h	none	min	10 080 (one week)

The second requirement is that one function build the table. If training features are produced by an analysis script and serving features by production code, the two will diverge, and the divergence will show up as an unexplained gap between validation error and operational error. The same function, called with a different issue time, must produce both; the contract of Table 4.5 is then a description of that function rather than an aspiration.

Conceptual questions

"What features would you give a model forecasting thermostat fleet load three hours ahead, and which of them must be forecasts rather than observations?"
A strong answer names the five families and then separates them by clock. From the series itself: the current load and one or two recent lags, the same hour yesterday and last week, and a 24-hour rolling mean to carry the recent level. From the calendar: hour of day, encoded as indicators or as a cyclic pair depending on the model class, a weekday or day-type indicator, and a holiday indicator. From the weather: the temperature at the target hour converted to cooling and heating degrees, humidity, and a 24-hour rolling maximum of temperature as a stored-heat proxy. From the fleet: the number of devices available, the duty cycle, and the mean gap between indoor temperature and setpoint. From the control record: the time since the last event, an indicator for the rebound window, and the historical response ratio. The answer then states which of these must be forecasts. Everything with a valid time at or before the issue time is an observation, and everything with a valid time at the target hour must be a forecast: the temperature and humidity at $t + 3$, hence the degree variables derived from them, and — if the fleet's composition moves on that timescale — the device count. The weather features must be read from the archive of forecasts as issued at the issue time, not from the observation history, because the model will be run against forecasts and must therefore learn the sensitivity of load to a forecast. The calendar features at the target hour are known exactly and need no forecast, because a clock is not a random variable.

4.7 What to retain

Definition 4.1 (Feature engineering)**.** Feature engineering is the choice of the map $X_t = \varphi(\mathcal{I}_t)$ that compresses everything known at the issue time into a fixed-length vector, built from lags, rolling statistics, calendar encodings, degree variables and device-state features. Its one inviolable constraint is that every datum φ reads must have *arrived* by the issue time, which makes the weather feature for the target hour a forecast as issued rather than the observation that eventually occurred.

Table 4.6 collects the chapter's objects with the subsection that defines each.

4.8 Exercises

Exercise 4.1. A fleet has three years of hourly load history, 26 280 rows. The feature set contains lags 1, 2, 3, 24, 25, 48 and 168, a 24-hour rolling mean, and 24 hour indicators. How many rows does the supervised table of Section 4.1.3 contain at $h = 1$, and how many at $h = 24$? Now suppose the meter feed arrives with a two-hour latency. Which lag features must be dropped, and how does the row count change?

Exercise 4.2. Using the hockey stick of Figure 4.6, compute CD and HD with $T_c^* = 22\,^\circ\mathrm{C}$ and $T_h^* = 16\,^\circ\mathrm{C}$ for temperatures -2, 12, 19, 28 and 37 °C, and evaluate $P = 55 + 2.2\,\mathrm{HD} + 1.9\,\mathrm{CD}$ at each. Then evaluate the dashed linear model $P = 62 + 0.25\,T$ at the same five temperatures and report the five errors. At which temperature is the linear model least wrong, and why is that the least useful place to be right?

Exercise 4.3. Write out the cyclic pair of equation (4.8) for hours 22, 23, 0 and 1, and compute the Euclidean distance between consecutive pairs. Compare with the distances under the raw integer encoding. Then add the second-harmonic pair at period 12 and state what shape of daily profile the four columns together can represent that the first pair alone cannot.

Table 4.6: The objects of a feature table.

Object	Definition	Section
φ	the map $\mathcal{I}_t \mapsto X_t$; lossy, and a modeling choice	4.1.2
P_{t-k}	lag of order k; orders chosen from the autocorrelation function	4.2.1
$\bar{P}_t^{(k)}$, $P_t^{\max,(k)}$	trailing rolling mean and maximum over k hours	4.2.3
$\Delta_{24} P_t$	seasonal difference used as a feature	4.2.4
one-hot hour	24 indicators, 23 free given an intercept	4.3.1
cyclic pair	$(\cos 2\pi\,\mathrm{hr}/24,\ \sin 2\pi\,\mathrm{hr}/24)$; makes hours 23 and 0 adjacent	4.3.2
CD_t, HD_t	$\max(T_t - T_c^*, 0)$, $\max(T_h^* - T_t, 0)$: the kink, supplied	4.3.3
$\hat{T}_{t+h\|t}$	weather forecast for the target time as issued at t	4.3.5
N_{avail}, D_t, ΔT_t	devices available, duty cycle, indoor-setpoint gap	4.4.3, 4.4.4
r_{hist}	verified response divided by commanded response	4.4.5
z_{t+h}	change target, $y_{t+h} - y_t$ or $y_{t+h} - y_{t+h-24}$	4.5.1
valid, arrival time	the period a datum describes; when it became available	4.6.1
leakage	violation of $\text{arrival}(d) \le t$; five mechanisms	4.6.2
feature contract	schema of source, offset, latency, units, default	4.6.4

Exercise 4.4. A colleague reports a validation root-mean-square error of 2.1 MW for a day-ahead fleet forecast, against 6.4 MW for seasonal persistence. The feature table contains a column `temp_target_hour` built by joining the observation history on target time, and a column `load_roll24` built with a window centered on the target hour. Using Table 4.4, name the two mechanisms at work, say which is likely to be the larger, and describe the two corrected columns and what you expect the validation error to become relative to the baseline.

Chapter 5

Regression and tree models: linear models, regularization and gradient boosting

5.1 Supervised learning of the forecast function

Chapter 4 built the feature vector X_t from the information set $\mathcal{I}_t$: lagged load, forecast weather, calendar variables, device states and fleet statistics, arranged so that one row of a table corresponds to one issue time and the target is the value y_{t+h} at the target time. Chapter 3 supplied the numbers any model must beat. What remains is the transition from the feature vector to the number the operator receives, and the question of this chapter is

which function should map X_t to $\hat{y}_{t+h}$, and what structure can each candidate function represent that the previous one could not?

The chapter answers it by starting with ordinary linear regression and relaxing its assumptions one at a time: first by penalizing the size of its coefficients, then by replacing the weighted sum with a partition of the feature space, then by adding many such partitions sequentially. Each step buys the ability to represent some structure the previous step could not, and each step costs something: interpretability, stability outside the training range, or the amount of data needed to fit the model honestly.

Jargon: supervised learning, design matrix, regularization, ensemble, gradient boosting, learning rate

Supervised learning is the fitting of a function f to pairs (X_t, y_{t+h}) drawn from history, so that $f(X_t)$ predicts y_{t+h} at issue times not used in fitting. The *design matrix* $\mathbf{X}$ is the table of those feature rows, one row per issue time, one column per feature. *Regularization* is the addition of a penalty on model complexity to the training error, so that the fitted model is not the one that reproduces history most exactly. An *ensemble* is a collection of models whose predictions are combined; averaging independently fitted models reduces variance, and fitting them in sequence, each to what the previous ones got wrong, is *gradient boosting*. The *learning rate* η is the fraction of each new model's correction that is actually applied.

Key idea

A forecasting model is a choice about how much structure the modeler specifies and how much the data is asked to discover. A linear regression on engineered features has its structure specified almost entirely by the modeler; a boosted ensemble of trees has considerably more of it learned from data.

This section sets up the supervised problem on the aggregator's thermostat fleet, reads the fitted coefficients as physical statements, evaluates one forecast by hand, and states what the fitting procedure actually optimizes.

5.1.1 The simplest supervised forecasting model

Take the concrete task of forecasting the aggregator's fleet load at 17:00 from information available in the early afternoon. The feature vector of Chapter 4, reduced to four entries for clarity, is

$$X_t = \big[\, P_t,\ P_{t-24},\ \hat{T}_{17:00},\ N_{\text{avail}} \,\big], \tag{5.1}$$

the current fleet load, the load at the same hour yesterday, the temperature forecast for the target hour as issued at t, and the number of thermostats available. The simplest function of these four numbers is a weighted sum,

$$\hat{P}_{17:00} = \beta_0 + \beta_1 P_t + \beta_2 P_{t-24} + \beta_3 \hat{T}_{17:00} + \beta_4 N_{\text{avail}}, \tag{5.2}$$

which is ordinary linear regression: the prediction is a weighted sum of the predictors plus a constant. The five numbers $\boldsymbol{\beta} = [\beta_0, \ldots, \beta_4]^{\mathsf{T}}$ are what fitting determines.

Two properties of equation (5.2) are worth naming before anything is fitted. It is linear in the *coefficients*, which is what makes the fit a solved problem rather than a search; and it is linear in the *features*, which is a restriction on the shapes it can represent and the restriction the rest of the chapter removes. The second is not a restriction on the shapes it can represent as a function of the raw measurements, because a feature may itself be a nonlinear function of a measurement. That distinction is the subject of Section 5.2.1.

5.1.2 What the coefficients mean

Suppose the model is fitted on a season of history and returns

$$\hat{P}_{17:00} = 10 + 0.25\, P_t + 0.40\, P_{t-24} + 1.8\, \text{CD}_{17:00} + 0.0005\, N_{\text{avail}}, \tag{5.3}$$

where the raw temperature of equation (5.2) has been replaced by the cooling-degree feature of Chapter 4,

$$\text{CD}_{17:00} = \max(\hat{T}_{17:00} - 22^{\circ}\text{C},\ 0), \tag{5.4}$$

measured in degrees above a 22°C base. Every coefficient of equation (5.3) carries a unit and reads as a physical statement.

The lag coefficients. The 0.40 on yesterday's same-hour load is dimensionless: holding the other predictors fixed, one megawatt more load yesterday at 17:00 is associated with about 0.4 MW more predicted load today at 17:00. The 0.25 on the current load says the same about the present level of the fleet. The two together say that a little over half of today's level is explained by persistence in one form or another, which is the quantitative version of the statement that the seasonal-persistence baseline of Chapter 3 is hard to beat.

The weather coefficient. The 1.8 has units of megawatts per degree: each cooling degree above the 22°C base adds 1.8 MW of fleet load. This is the aggregate of a few hundred thousand compressors running a little longer per hour, and it is a number an engineer can check against the fleet's installed capacity.

The availability coefficient. The 0.0005 has units of megawatts per device, that is 0.5 kW per available thermostat. It is not the connected load of an air conditioner, which is several kilowatts, but the product of that load with a duty cycle and with the fraction of devices actually cooling. A coefficient of this kind is useful precisely because it can be audited for plausibility.

The ability to read the fitted model this way is the main practical argument for keeping a linear model in the stack even when a more flexible model forecasts better. The coefficients are the model's statement of what it believes about the fleet, in units an operator can dispute.

5.1.3 A worked forecast

Place the issue time at 14:00 on a hot afternoon with

$$P_t = 90\,\mathrm{MW}, \qquad P_{t-24} = 105\,\mathrm{MW}, \qquad \hat{T}_{17:00} = 32^{\circ}\mathrm{C}, \qquad N_{\mathrm{avail}} = 80\,000. \tag{5.5}$$

The temperature forecast gives $\mathrm{CD}_{17:00} = \max(32 - 22, 0) = 10$ cooling degrees, and equation (5.3) evaluates term by term,

$$\hat{P}_{17:00} = \underbrace{10}_{\text{intercept}} + \underbrace{22.5}_{0.25\times 90} + \underbrace{42.0}_{0.40\times 105} + \underbrace{18.0}_{1.8\times 10} + \underbrace{40.0}_{0.0005\times 80\,000} = 132.5\,\mathrm{MW}. \tag{5.6}$$

Figure 5.1 shows the five terms as a bar chart. The horizontal axis is the contribution of each term in megawatts, the five upper bars are the terms of equation (5.6) in the order they appear there, and the lowest bar is their sum. The picture makes two points about this particular forecast. Half of the predicted load, the 42.0 and the 22.5 MW terms, comes from the two lag features and is therefore available without any weather input; the weather term contributes 18.0 MW, about 14 % of the total, which is the part of the forecast exposed to an error in the temperature forecast. The engineered features of Chapter 4 have been mapped directly to a number an operator can act on, and the mapping is auditable line by line.

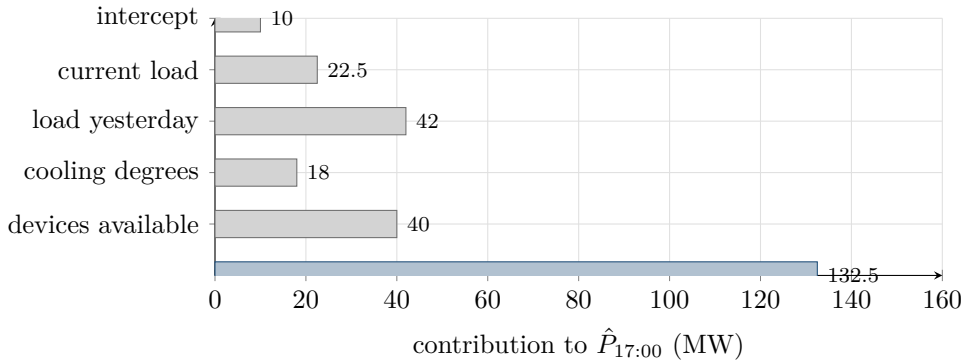


Figure 5.1: The five terms of the worked forecast of equation (5.6), in megawatts, with their sum in the bottom bar. The inputs are those of equation (5.5) and the coefficients those of the fitted model of equation (5.3); both are illustrative.

5.1.4 What fitting optimizes

Stack the history into a design matrix whose i-th row is $x_i^{\mathsf{T}} = [1, x_{i1}, x_{i2}, \ldots, x_{ip}]$ and whose target vector is y. The model's predictions for the whole history are $\hat{y} = \mathbf{X}\boldsymbol{\beta}$ and its residuals are $e = y - \mathbf{X}\boldsymbol{\beta}$. Ordinary least squares chooses the coefficients that make the sum of squared historical errors as small as possible,

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - x_i^{\mathsf{T}}\boldsymbol{\beta})^2 = \arg\min_{\boldsymbol{\beta}} \|y - \mathbf{X}\boldsymbol{\beta}\|_2^2, \tag{5.7}$$

which has the closed-form solution

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}y \tag{5.8}$$

whenever $\mathbf{X}^{\mathsf{T}}\mathbf{X}$ is invertible. Training, in this model class, means nothing more than evaluating equation (5.8).

Three remarks follow from writing the objective down explicitly, and all three return later in the chapter. The criterion is the squared error on *history*, so a model class flexible enough to drive it to zero will do so without that being an achievement: this is the overfitting of Section 5.3.3. The criterion is symmetric in the sign of the error, which Chapter 10 shows is the wrong criterion when over-forecasting and under-forecasting have different operational costs. And the inverse in equation (5.8) is the point at which correlated features cause trouble, because $\mathbf{X}^{\mathsf{T}}\mathbf{X}$ becomes nearly singular when two columns carry almost the same information, which is the subject of Section 5.2.4.

5.2 Linear models and regularization

This section examines where the weighted sum of equation (5.2) is an excellent model and where it is not, and then adds the penalty terms that make it usable when the feature set contains many overlapping columns.

5.2.1 Why a linear model can be sufficient

Suppose the true data-generating relationship for a thermostat fleet were

$$P_t = 50 + 1.7\,\mathrm{CD}_t + 10\,I_{\text{evening}} + 0.6\,P_{t-1} + \varepsilon_t, \tag{5.9}$$

with I_{evening} an indicator that the hour lies in the evening peak. Equation (5.9) is already a linear regression, and no amount of model flexibility can improve on recovering its four coefficients. The important observation is that the relationship is linear in CD_t and I_{evening}, which are themselves nonlinear functions of temperature and of the timestamp. The nonlinearity has been moved out of the model and into the features.

This is what good feature engineering buys: a simple model can represent domain knowledge it would otherwise have to discover. Giving the model $\mathrm{CD}_t = \max(T_t - 22^{\circ}\mathrm{C}, 0)$ spares it the task of discovering that cooling load has a threshold near 22°C; giving it I_{evening} spares it the task of discovering that evenings exist. Both facts are known from the physics of air conditioning and from the shape of household occupancy, and a model that is told them spends its data budget on the coefficients instead of on the shape.

5.2.2 Where a single linear term fails

The real temperature response of a thermostat fleet has a characteristic shape. Below about 22°C almost no cooling load exists. From 22 to 32°C the load increases as compressors run longer each hour. Around 35°C the response is strong and nearly linear in temperature. At the extremes the fleet saturates: compressors approach continuous operation and cannot draw more power however much hotter it gets.

Figure 5.2 plots that shape together with two linear fits. The horizontal axis is the temperature at the target hour and the vertical axis is fleet load; the thick curve is the physical response, the straight line is the least-squares fit of a model whose only weather feature is the raw temperature T, and the kinked line is the fit of a model given the cooling-degree feature of equation (5.4). The straight-line fit is wrong in a systematic way at both ends: it predicts about 43 MW of load at 15°C, where the true response is 50 MW of non-cooling load and no cooling at all, and it under-predicts by about 6 to 7 MW through the range from 33 to 36°C

where the system peaks. A single coefficient $\beta_3 T$ cannot represent a response that is flat over part of its range and steep over another.

The cooling-degree fit removes most of that error, because the kink in $\max(T - 22, 0)$ is exactly the kink the physics has. It still misses the saturation above 36°C, since one cooling-degree term grows without bound; capturing the upper bend as well requires a quadratic term, a second knot, or a spline basis. Each of those is available, and each of them is a piece of nonlinear structure constructed by hand. The limitation of the linear model is therefore not that it cannot represent the response, but that somebody has to specify the shape in advance.

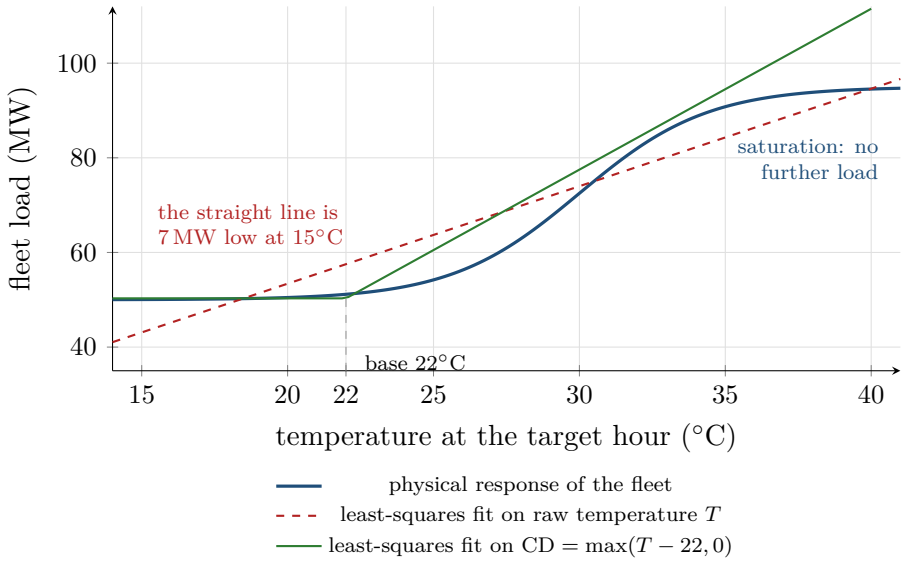


Figure 5.2: The load–temperature response and two linear fits. The physical response is drawn as $P(T) = 50 + 45/(1 + e^{-(T-30)/2.2})$ MW, which is flat below 22°C, steep near 30°C and saturating above 36°C. The dashed line is $P = 12.2 + 2.06\,T$, the least-squares fit of a raw-temperature model on a uniform grid over 15 to 40°C; the kinked line is $P = 50.3 + 3.4\,\mathrm{CD}$ with $\mathrm{CD} = \max(T - 22, 0)$. All values are illustrative.

5.2.3 Interactions must be specified

A second kind of structure the weighted sum cannot represent by itself is an interaction: a predictor whose effect depends on the value of another predictor. Temperature matters differently at 03:00 and at 17:00, because at 03:00 the building is unoccupied, the setpoint is relaxed and the solar gain is zero. In the linear model the effect of temperature is the single number β_1 whatever the hour, so the fitted β_1 is a compromise between the two regimes.

The remedy is to add the product of the two features as a third feature,

$$P = \beta_0 + \beta_1 T + \beta_2 I_{\text{evening}} + \beta_3 (T \times I_{\text{evening}}), \tag{5.10}$$

after which the temperature slope is β_1 outside the evening and $\beta_1 + \beta_3$ during it. Linear models can represent interactions, then, provided the interaction is named. The difficulty is combinatorial: with p features there are $p(p-1)/2$ pairwise products and many more higher-order terms, so a feature set of twenty columns offers 190 candidate pairs, of which a handful matter and the rest add variance. Choosing among them by hand does not scale, and choosing among them by search on the training set re-introduces the overfitting that equation (5.7) invites. Tree models, taken up in Section 5.3, handle interactions differently and without being told which ones to look for.

5.2.4 Ridge regression

The feature sets of Chapter 4 contain many columns that carry overlapping information. A thermostat-fleet model may be given P_{t-1}, P_{t-2}, P_{t-3}, P_{t-24} and P_{t-25} at once; a load series with the autocorrelation structure of Chapter 2 makes consecutive lags nearly identical. When two columns of $\mathbf{X}$ are nearly proportional, $\mathbf{X}^\mathsf{T}\mathbf{X}$ in equation (5.8) is nearly singular, and its inverse amplifies small changes in the data into large changes in $\hat{\boldsymbol{\beta}}$. The fitted coefficients become unstable: refitting on a slightly different window can reverse their signs.

Ridge regression adds a penalty on the squared size of the coefficients to the least-squares objective,

$$\hat{\boldsymbol{\beta}}^{\,\text{ridge}} = \arg\min_{\boldsymbol{\beta}} \left[\sum_{i=1}^{n} (y_i - x_i^\mathsf{T}\boldsymbol{\beta})^2 \;+\; \lambda \sum_{j=1}^{p} \beta_j^2 \right], \tag{5.11}$$

with the intercept left unpenalized and the features standardized so that the penalty treats them comparably. The weight $\lambda \geq 0$ sets the exchange rate between fitting history and keeping the coefficients small; $\lambda = 0$ recovers ordinary least squares and large λ drives every coefficient toward zero. In closed form the solution is $(\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\mathsf{T}y$, which is invertible for any $\lambda > 0$: the penalty repairs the conditioning problem directly.

5.2.5 What the penalty does to a collinear pair

The effect is clearest on two features that are nearly the same column. Suppose the model is given both P_{t-1} and a two-period rolling mean of the load, which contains P_{t-1} as half of itself. Ordinary least squares may return $\beta_1 = 8$ and $\beta_2 = -7.4$. Those coefficients are not nonsense: because the two features move together, a large positive weight on one and a nearly equal negative weight on the other reproduces the training data about as well as a moderate weight on each, and the optimizer has no reason to prefer the moderate solution. It is, however, a fragile fit, and it asserts something physically absurd, namely that recent load has a large negative effect on future load.

Ridge regression resolves the ambiguity by preferring the solution of smallest squared norm among those that fit nearly as well. For this pair it returns something like $\beta_1 = 0.35$ and $\beta_2 = 0.28$. Figure 5.3 traces the two coefficients as λ increases. The horizontal axis is the penalty weight, the vertical axis is the coefficient value, the two solid curves are β_1 and β_2, and the dotted curve is their sum. Reading the figure from left to right: at $\lambda = 0$ the two coefficients are 8 and -7.4 and nearly cancel; as the penalty grows they collapse rapidly onto 0.35 and 0.28; and throughout, their sum stays close to 0.6. The sum is what the data identifies, since the two features are nearly the same column and only their combined effect is visible; how that sum is divided between them is almost unconstrained, and the penalty supplies the missing preference. A little training fit has been traded for stability, which is what regularization means.

5.2.6 Lasso and feature selection

Replacing the squared penalty by the sum of absolute values gives the lasso,

$$\hat{\boldsymbol{\beta}}^{\,\text{lasso}} = \arg\min_{\boldsymbol{\beta}} \left[\sum_{i=1}^{n} (y_i - x_i^\mathsf{T}\boldsymbol{\beta})^2 \;+\; \lambda \sum_{j=1}^{p} |\beta_j| \right]. \tag{5.12}$$

The change of penalty changes the character of the solution. The absolute-value penalty has a constant slope all the way to zero, so for a feature whose contribution to the fit is smaller than that slope the optimum sits exactly at $\beta_j = 0$ rather than at a small nonzero value. The lasso therefore performs feature selection as part of fitting: a model offered a hundred candidate features may come back with seventeen active ones and the rest set to zero.

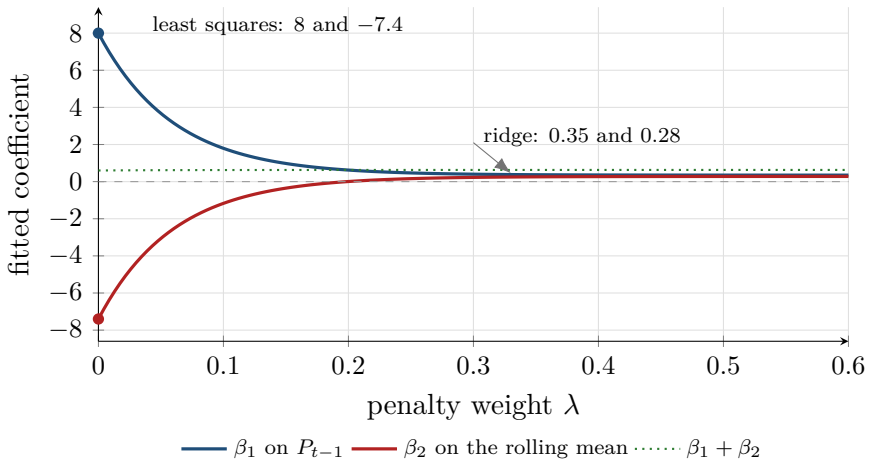


Figure 5.3: Ridge coefficient paths for two nearly identical features. The curves are $\beta_1(\lambda) = 0.35 + 7.65\,e^{-\lambda/0.06}$ and $\beta_2(\lambda) = 0.28 - 7.68\,e^{-\lambda/0.06}$, chosen to pass through the least-squares values 8 and -7.4 at $\lambda = 0$ and the ridge values 0.35 and 0.28 at moderate penalty; the dotted curve is their sum. Values are illustrative and the shape is the one equation (5.11) produces on a collinear pair.

For operational work the distinction between the two penalties matters less than the presence of one. Ridge is the natural choice when the features are genuinely collinear and all of them belong in the model, which is the usual situation with lagged load; the lasso is the natural choice when the feature set is large and mostly irrelevant and a short list is wanted. What matters in both cases is the object being minimized,

$$\text{training error} \;+\; \lambda \times \text{model complexity}, \tag{5.13}$$

and the fact that λ is chosen on data not used for fitting, by the rolling-origin procedure of Chapter 9, never on the training set itself.

Principle 5.1. *Regularization replaces the question "which coefficients reproduce history best" with "which coefficients reproduce history well while staying small". The second question has a stable answer when the features overlap; the first does not.*

5.3 Decision trees and ensembles

A tree abandons the weighted sum entirely and replaces it with a partition of the feature space. This section defines the object, states why its shape suits energy data, shows why one tree is not usable on its own, and introduces averaging as the first repair.

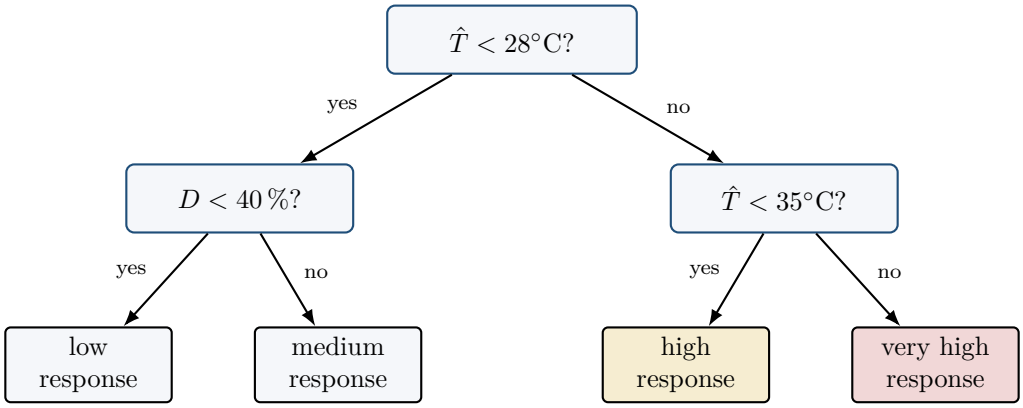
5.3.1 Partitioning the feature space

Consider forecasting the flexibility of a thermostat fleet, which Chapter 1 defined as the change in load the fleet could deliver on request, and which depends strongly on the outdoor temperature and on how hard the compressors are already working. A decision tree represents that dependence by asking a sequence of questions about single features. A first split asks whether the temperature is below 28°C. In the cool branch a second split asks whether the duty cycle is below 40 %, separating a fleet with little running load from one with moderate running load. In the warm branch a second split asks whether the temperature is below 35°C, separating a hot afternoon from an extreme one.

Figure 5.4 draws that tree. The boxes are the splits, each labeled with the question it asks; the edges are labeled with the answer taken; and the four rounded boxes at the bottom are the terminal regions, each carrying one predicted response level. The structure to notice is that nothing in the diagram is a coefficient. In place of $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$ the model is

$$f(X) = \sum_{j=1}^{J} c_j \, \mathbf{1}\{X \in R_j\}, \tag{5.14}$$

a sum over J disjoint regions R_j of the feature space, with $\mathbf{1}\{\cdot\}$ the indicator function and c_j the constant predicted inside region j. The regions are axis-aligned boxes, since each split tests one feature against one threshold, and the constants are the mean of the training targets that fall in each box. A tree approximates $f(X)$ by a piecewise-constant function, and fitting it means choosing the splits.



Each path from the root to a leaf is one region R_j of equation (5.14).

Figure 5.4: A decision tree for thermostat-fleet flexibility. The two levels of splits test the forecast temperature $\hat{T}$ and the fleet duty cycle D against thresholds; the four leaves are the terminal regions and each carries one constant prediction. The thresholds are illustrative.

Example 5.1 (A one-variable tree)**.** The simplest useful tree splits on temperature alone and predicts a flexibility of about 10 MW when $\hat{T} < 25°\text{C}$, 25 MW when $25°\text{C} \le \hat{T} < 32°\text{C}$, and 45 MW when $\hat{T} \ge 32°\text{C}$. Nothing in the fitting procedure was told that cooling load has thresholds; the two cut points were chosen because they reduced the squared error of the piecewise-constant fit more than any other pair. The tree discovered the thresholds without being given a nonlinear equation.

5.3.2 Why the shape suits energy data

Four properties of structured energy data make equation (5.14) a natural functional form.

Thresholds. Physical behavior changes at values, not smoothly: above about 30°C a substantial fraction of air conditioners move from cycling to near-continuous operation, and the fleet's response to a curtailment command changes character at that point. A split is exactly a threshold.

Interactions. Temperature may matter only when the hour is past 15:00 and the available device count exceeds 50 000. A tree represents this without being told: the temperature split simply appears below the hour split on one branch and not on the other. Figure 5.5 draws the

resulting partition of the plane spanned by temperature and hour of day. The horizontal axis is the forecast temperature, the vertical axis is the hour, each rectangle is one terminal region, and the number inside each rectangle is the constant it predicts. The region below hour 15 is undivided in the temperature direction above 28°C, which is the interaction made visible: the same temperature difference changes the prediction by 18 MW in the evening and not at all in the morning. A linear model reproduces this only after the product of the two features has been constructed by hand, as in equation (5.10).

Nonlinearity without a basis. The staircase of Example 5.1 approximates a curved response with no polynomial, spline or knot placement. The approximation is crude locally and improves as splits are added.

Mixed feature types. The feature vector of Chapter 4 mixes continuous weather variables, integer calendar variables, counts of devices, fractions such as the duty cycle and categorical day types. A split compares one feature with one threshold or one category, so no scaling, encoding or units reconciliation is needed. In particular, the cyclic encoding of the hour that a linear model requires is unnecessary, because a tree can simply split the hour twice.

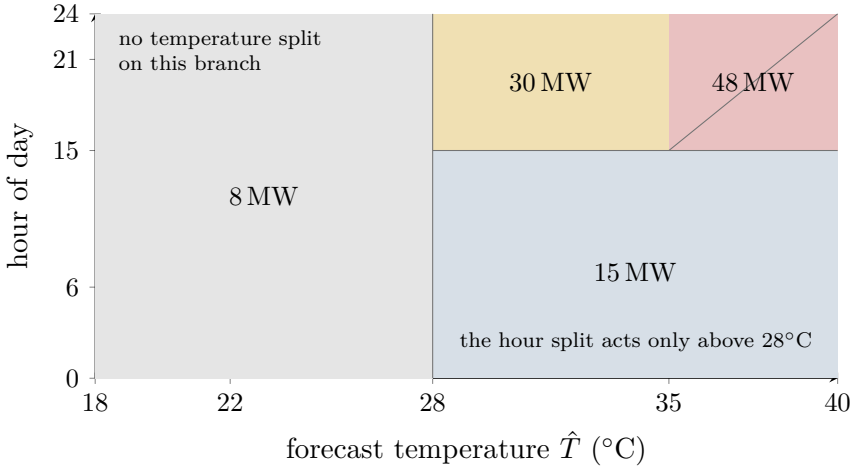


Figure 5.5: The partition of the temperature–hour plane produced by a tree with three splits: temperature at 28°C, hour at 15:00 on the warm branch only, and temperature again at 35°C on the warm evening branch. Each rectangle is a terminal region R_j of equation (5.14) and carries one constant. Values are illustrative.

5.3.3 One tree is not enough

A single tree fitted to its own training set is a high-variance estimator: a slightly different sample of history produces a noticeably different tree, because changing a split near the root changes everything below it. Two models that differ this much under resampling cannot both be close to the truth.

The mechanism is the one equation (5.7) invites. Nothing in the fitting criterion stops a tree from continuing to split, and each additional split reduces the training error. Left unconstrained, a tree carves history into regions containing a handful of observations each and asks questions of the form “is the temperature below 31.73°C, the humidity below 64.2 % and the available device count below 61 284?”. The thresholds carry that many digits because they are placed to separate individual historical rows. Such a tree reproduces the training set almost exactly and predicts poorly out of sample: it has memorized rather than generalized. This is overfitting,

and the two standard remedies are to constrain the tree, by limiting its depth or requiring a minimum number of observations per leaf, and to average many trees instead of trusting one.

5.3.4 Random forests

Averaging is the first of the two ensemble constructions. Build M trees $T_1(X), \dots, T_M(X)$ and predict their mean,

$$\hat{y} = \frac{1}{M} \sum_{m=1}^{M} T_m(X). \tag{5.15}$$

For the average to help, the trees must differ, and a random forest makes them differ deliberately in two ways: each tree is fitted to a bootstrap resample of the observations, and at each split it may choose among only a random subset of the features. The second device is the more important one in energy data, because without it every tree would put the dominant weather variable at the root and the trees would be nearly identical.

The statistical effect is a variance reduction. If the individual trees had independent errors of variance σ^2, the average would have variance σ^2/M; because their errors are in fact positively correlated, the reduction is smaller than that, and the residual variance is set by the correlation between trees rather than by their number. The decorrelation devices are what make the average worth taking. In practice one tree emphasizes the forecast temperature, another the load at the same hour yesterday, another the duty cycle and another the available device count; the trees make different mistakes, and the ensemble is more robust than any member. What the average does not do is remove bias: a forest of shallow trees is a smoother estimator than one tree, but still a piecewise-constant one.

5.4 Gradient boosting

The second ensemble construction builds the trees in sequence rather than independently, each one fitted to what the ensemble so far still gets wrong. This section states the recursion, works a four-point example by hand, examines the role of the learning rate, and explains why the construction suits the feature sets of this book.

5.4.1 The additive model fitted to residuals

Start from a constant prediction, the mean of the training target,

$$\hat{y}^{(0)} = \bar{y}, \tag{5.16}$$

and compute what that model gets wrong,

$$r^{(1)} = y - \hat{y}^{(0)}. \tag{5.17}$$

Fit a small tree $T_1(X)$ to the residual vector $r^{(1)}$ rather than to the target, and add a fraction of its prediction to the model,

$$\hat{y}^{(1)} = \hat{y}^{(0)} + \eta\, T_1(X), \tag{5.18}$$

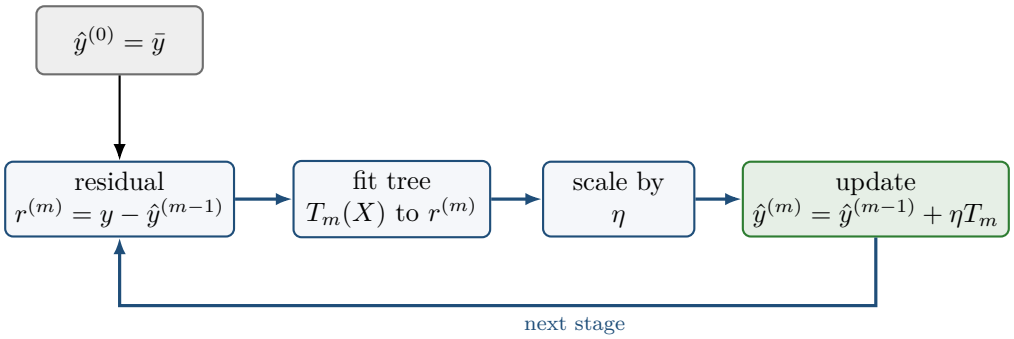
with $0 < \eta \leq 1$ the learning rate. Recompute the residual against the new model and repeat. After m stages the model is

$$\hat{y}^{(m)} = \hat{y}^{(m-1)} + \eta\, T_m(X) = \bar{y} + \eta \sum_{k=1}^{m} T_k(X), \tag{5.19}$$

an additive model in which every term after the first is a tree fitted to the error of its predecessors. The name of the method comes from the observation that, for squared-error loss, the residual

$y - \hat{y}^{(m-1)}$ is the negative gradient of the loss with respect to the prediction, so each stage is one step of gradient descent in function space; other loss functions replace the residual by their own gradient and leave the rest of the construction unchanged.

Figure 5.6 draws the loop. The boxes along the top are the four operations of one stage: form the residual, fit a tree to it, scale the tree by η, add it to the model; the arrow returning to the left closes the loop, and the text beneath records what the first four trees of a thermostat-fleet model might each be correcting. Each new tree asks what systematic error the current model still makes. Tree 1 finds that hot days carry higher load than the constant. Tree 2 finds that the model, now weather-aware, still under-predicts hot weekday evenings. Tree 3 finds that it over-predicts when device availability is low. Tree 4 finds that it misses the recovery of load after a recent dispatch event. The sequence is a progressive correction of mistakes, and the order in which the corrections appear is itself informative about the fleet.



What successive trees correct in a thermostat-fleet model: (1) hot days carry higher load; (2) hot weekday evenings are still under-predicted; (3) low device availability is over-predicted; (4) the load rebound after a recent dispatch event is missing.

Figure 5.6: One stage of gradient boosting, equations (5.16) to (5.19). The model is initialized at the mean; each stage fits a tree to the current residual, scales it by the learning rate and adds it to the model.

5.4.2 A four-point example

The recursion is small enough to run by hand. Take four observations with targets $y = [20, 30, 40, 50]$, in megawatts, ordered from the coolest to the hottest day. The initial model is the mean, $\hat{y}^{(0)} = [35, 35, 35, 35]$, and the first residual is $r^{(1)} = [-15, -5, 5, 15]$. A tree with a single split on temperature can separate the two cool days from the two warm ones and predicts the group means of the residual, $T_1 = [-10, -10, 10, 10]$. With a learning rate $\eta = 0.5$ the applied correction is half of that, $[-5, -5, 5, 5]$, and the updated model is $\hat{y}^{(1)} = [30, 30, 40, 40]$. The remaining residual is $r^{(2)} = [-10, 0, 0, 10]$.

Table 5.1 lists the six vectors. Two features of the arithmetic are worth stating. The first stage removed the part of the error that a single temperature split can describe, and what remains is error *within* the two groups, which the next tree must split on some other feature, or on temperature again at a finer threshold, to reach. And the correction was deliberately incomplete: had η been 1, the model would have jumped to $[25, 25, 45, 45]$ and the residual to $[-5, 5, -5, 5]$, which is smaller in magnitude but has had its sign pattern scrambled by one aggressive step. Taking half of each step leaves the next tree a residual that still has interpretable structure.

Table 5.1: One boosting stage on four observations, in megawatts, with $\eta = 0.5$. The columns follow equations (5.16) to (5.19).

Quantity	day 1	day 2	day 3	day 4
target y	20	30	40	50
initial model $\hat{y}^{(0)} = \bar{y}$	35	35	35	35
residual $r^{(1)}$	−15	−5	5	15
tree prediction $T_1(X)$	−10	−10	10	10
applied correction $\eta\, T_1(X)$	−5	−5	5	5
updated model $\hat{y}^{(1)}$	30	30	40	40
residual $r^{(2)}$	−10	0	0	10

5.4.3 Learning rate and number of trees

The learning rate and the number of stages are not independent. The total correction applied is $\eta \sum_k T_k$, so halving η roughly doubles the number of trees needed to fit the same structure. The reason to prefer a small η with many trees over a large η with few is that each tree is fitted to a residual computed from noisy data, so part of what it captures is noise; applying only a fraction of each tree limits how much of that noise enters the model, and averaging over many small corrections leaves a smoother function.

Figure 5.7 shows the trade-off. The horizontal axis is the number of trees, the vertical axis is the validation error in arbitrary units, and the two curves are a large and a moderate learning rate; the marked points are the minima. The large learning rate descends quickly, reaches its best validation error after about 270 trees and then deteriorates as further trees fit noise. The moderate rate descends more slowly, needs about 980 trees, and reaches a lower minimum. A learning rate ten times smaller again would need some 3 900 trees and reach slightly lower still, at a training cost that is usually not worth paying. The practical procedure is to fix η at a small value, add trees while the validation error falls, and stop when it does not: the number of trees is chosen by early stopping on held-out data, by the rolling-origin construction of Chapter 9, and never on the training error, which decreases monotonically and therefore carries no information about when to stop.

Production implementations. Gradient boosting as described is a few lines of arithmetic around a tree fitter. The libraries in common use, among them XGBoost and LightGBM, implement the same construction with engineering added: penalties on leaf values and on the number of leaves, efficient algorithms for finding splits on large data, explicit handling of missing values so that incomplete telemetry rows need not be dropped, monotonicity and interaction constraints that let physical knowledge be imposed, subsampling of rows and columns per stage, and control of the learning rate. The useful mental model remains equation (5.19): many small trees, fitted in sequence, each correcting the errors of the ones before.

5.4.4 Why boosting suits these features

Consider the feature set a thermostat-fleet model is actually given after Chapter 4,

$$X_t = \left[\, P_{t-1},\ P_{t-24},\ P_{t-168},\ \hat{T},\ \mathrm{RH},\ \mathrm{hour},\ \mathrm{weekday},\ N_{\mathrm{avail}},\ D,\ T_{\mathrm{in}} - T_{\mathrm{set}} \,\right], \tag{5.20}$$

ten columns mixing lagged load, weather, calendar and fleet state. Four properties of the mapping from this vector to the target decide which model class fits.

The relationships are nonlinear, as Figure 5.2 showed. They are threshold-based, since HVAC behavior changes at particular temperatures and the comfort band is a hard limit. They

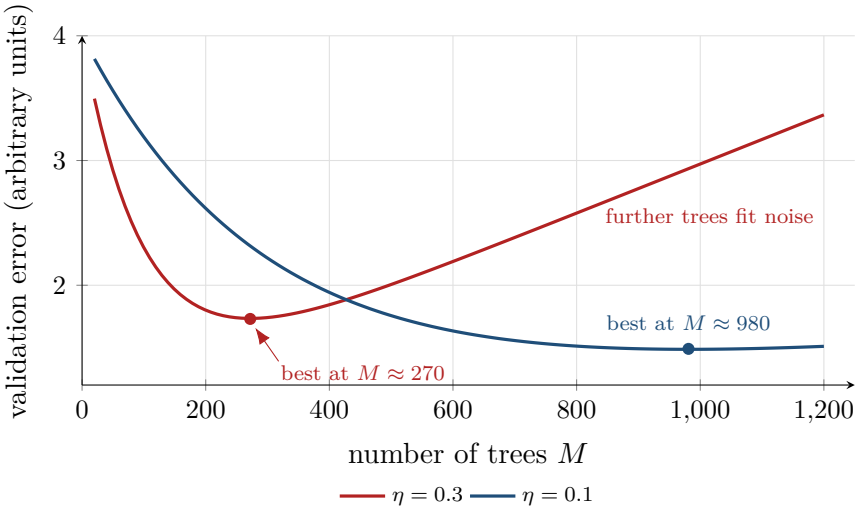


Figure 5.7: Validation error against the number of trees for two learning rates. The curves are $E(M;\eta) = 1 + 3\,e^{-\eta M/30} + 0.012\,\eta^{1.5} M$ in arbitrary error units, a shape chosen to show the two competing effects: the exponential term is the structure the ensemble captures and the linear term is the noise it accumulates. Values are illustrative.

are highly interactive: temperature, hour, availability and duty cycle qualify each other, as Figure 5.5 illustrated. And once P_{t-1}, P_{t-24} and P_{t-168} are present, the target is not strongly dependent on history beyond them, because the lag features have already summarized the daily and weekly structure that Chapter 2 found in the autocorrelation. Nonlinear, threshold-based, interactive and not long-memory is precisely the region of problem space in which boosted trees are competitive, which is why they are the model to beat on data of this shape rather than one option among many.

5.5 Choice of model class

The remaining question is which of the three classes to use, and the answer depends on where the structure comes from. This section compares a linear model with a boosted ensemble on a response whose shape is known, states the price of flexibility, compares both with a sequence model, and sets out the order in which the classes should be tried.

5.5.1 A kinked response, fitted two ways

Let the true relationship between forecast temperature and fleet flexibility be the piecewise-linear function

$$F(T) = \begin{cases} 5, & T < 24^\circ\mathrm{C}, \\ 5 + 3\,(T - 24), & 24^\circ\mathrm{C} \le T < 34^\circ\mathrm{C}, \\ 35, & T \ge 34^\circ\mathrm{C}, \end{cases} \tag{5.21}$$

in megawatts: no response below 24°C, a linear ramp of 3 MW per degree through the middle range, and saturation at 35 MW. Suppose the training record spans 20 to 35°C.

Figure 5.8 shows three objects against temperature: the true response of equation (5.21), the least-squares line fitted to it using raw temperature as the only feature, and the staircase a boosted ensemble produces. The line is $F = -47.7 + 2.36\,T$; it is 5.5 MW too low at 20°C, where the true response is flat at 5 MW, 4 MW too high at the lower knee, and about 2 MW too low at 33°C, because one slope must serve a flat region, a ramp and a plateau. The staircase tracks the true function to within 3.8 MW everywhere, having placed its cut points where the function

bends. Neither model was told that equation (5.21) has knees at 24 and 34°C; the boosted ensemble located them from data, and the linear model would have needed them supplied as $\max(T - 24, 0)$ and $\max(T - 34, 0)$ features. This is the contrast the chapter has been building toward: in a linear model the structure is largely specified by the modeler, and in a boosted tree ensemble considerably more of it is learned from the data.

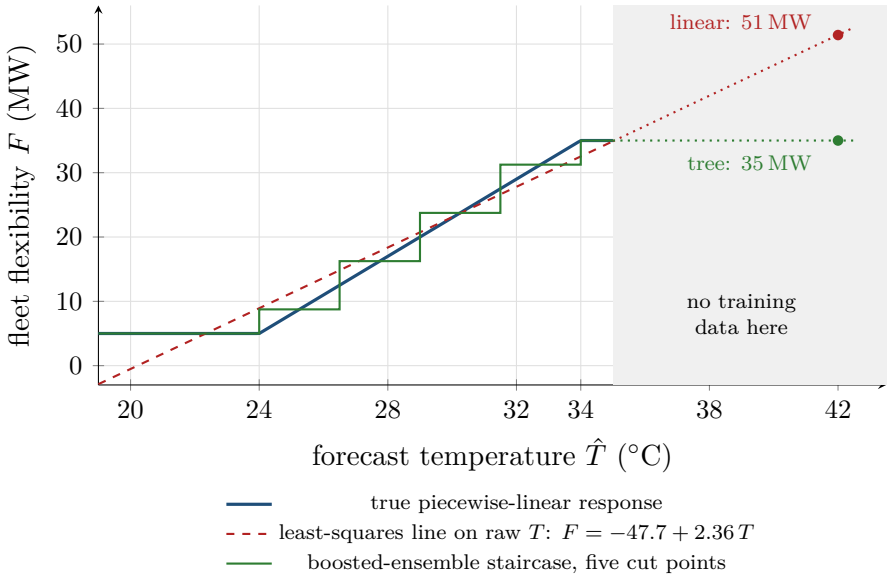


Figure 5.8: A piecewise-linear response fitted by a linear model and by a boosted ensemble. The training range is 20 to 35°C; the shaded band to the right has no training data. The line is the least-squares fit of equation (5.21) on a one-degree grid over the training range, and the staircase takes the mean of the true function over each of six bins with cut points at 24, 26.5, 29, 31.5 and 34°C. The dotted extensions are the two models' predictions at 42°C.

5.5.2 The price of flexibility

Flexibility in a model class is not free, and four costs recur. A flexible model can fit noise, which is the overfitting of Section 5.3.3. It behaves badly outside the range of its training data. It is harder to interpret, since equation (5.14) summed over several hundred trees has no coefficient to read. And it can discover relationships that are artifacts of the sample, which a coefficient-based model would have exposed as a physically implausible number.

The extrapolation cost deserves the detail, because it is the one that produces operational failures. Suppose the model was trained on temperatures from 20 to 35°C and tomorrow's forecast is 42°C. The two dotted extensions in Figure 5.8 show what each model does. The linear model continues its fitted trend and predicts 51 MW; the boosted ensemble predicts 35 MW, the value of its highest terminal region, because equation (5.14) is constant outside the outermost cut point. In effect the tree says that 42°C looks like the hottest cases it has seen, and no tree can say anything else.

Which prediction is better is not a statistical question. If the underlying physics continues the trend, for example a cooling load that keeps rising with temperature because compressors still have headroom, the linear model's extrapolation is approximately right and the tree's is 16 MW low. If the physics saturates, as equation (5.21) does and as a fleet running at full duty cycle must, the tree's flat extension is right and the linear model over-promises by the same margin. Neither answer is supported by the data, since there is no data in that range; the choice between them is a modeling assumption, and it should be made explicitly rather than inherited from the model class. Chapter 9 returns to this under the heading of distribution

shift, where the practical remedies are to monitor whether live feature values fall inside the training range and to blend in a physically parameterized model when they do not.

Principle 5.2. *A tree ensemble interpolates within the convex hull of its training features and is constant outside it. A model class is therefore also a choice of extrapolation behavior, and the choice is invisible until the day it matters.*

5.5.3 Tree models and sequence models

The third comparison is with the recurrent networks of Chapter 7. Suppose the model is already given P_{t-1}, P_{t-2}, P_{t-24}, P_{t-168}, the forecast weather $\hat{T}_{t+h}$, the hour, the weekday and the fleet's device state. Much of the temporal structure of the series has then been summarized explicitly, by hand, in the columns of equation (5.20). What is left is the problem $\hat{y} = f(X_t)$ with complex nonlinear interactions among those columns, and that is a problem boosted trees handle very well.

A recurrent network attacks a harder problem than it needs to in that situation. It must learn which parts of the history matter, learn a representation of that history in its hidden state $\mathbf{h}_t$, and learn the nonlinear mapping from the representation to the target, all from the same data. When the relevant history has already been engineered into features, the first two of those three tasks have been done, and the network's extra machinery buys little while costing data, training time and diagnosability.

The sequence model becomes the better choice when the target depends on a trajectory whose relevant parts are not known in advance,

$$y_{t+h} = f\big(x_t,\ x_{t-1},\ \ldots,\ x_{t-100}\big), \tag{5.22}$$

with no small subset of those hundred lags carrying the information. The physical example in this book is the thermal path of a building: morning temperature drives solar gains, which drive HVAC cycling, which charges the building's thermal mass, which sets the afternoon indoor state, which determines the evening flexibility. The chain runs through the whole day, and choosing by hand to include x_{t-3}, x_{t-8} and x_{t-17} is a crude substitute for it. A sequence model can learn a latent summary of the trajectory, which is the capability Chapter 7 develops.

Table 5.2 collects the comparison across the three classes. The rows are the properties that decide the choice and the columns are the classes; the entries are the statements the preceding sections established. The pattern down the table is a single trade: as the structure moves from the modeler to the data, the model represents more shapes and behaves worse in the places where the data is absent or misleading.

5.5.4 A practical model hierarchy

The comparison implies an order of work rather than a choice made once. Table 5.3 lists four levels: the persistence and seasonal-persistence baselines of Chapter 3, a linear or ridge model on engineered features, a gradient-boosted ensemble, and a sequence model. The second column names the model, the third says what capability the level adds over the one above it, and the fourth names what it costs. The discipline the table enforces is to ask, at each level, what incremental accuracy the added complexity buys against the level above, measured on held-out data by the procedure of Chapter 9. A level that does not buy a material improvement is not adopted, and the baseline at the top is never skipped, because the skill score of Chapter 3 is defined relative to it.

Table 5.2: The three model classes of this chapter compared on the properties that decide the choice.

Property	Linear or ridge	Boosted trees	Sequence model
Source of structure	specified by the modeler through features	learned from data, splits and depth	learned from data, including the history representation
Nonlinearity	only through engineered features	piecewise constant, refined by stages	smooth and learned
Interactions	must be named, equation (5.10)	automatic, by nesting splits	automatic
Thresholds	must be supplied as CD or splines	native, each split is a threshold	learned, less sharply
Extrapolation	follows the fitted trend	constant beyond the outermost cut	unreliable
Data required	least	moderate	most
Interpretability	one coefficient per feature, auditable	importances and partial effects only	opaque
Typical failure	bias where the response bends	memorizing noisy splits, flat extrapolation	overfitting with too little data
Wins when	the response is nearly linear in good features	features are rich, structured and interactive	long trajectories carry information the features miss

Table 5.3: The order in which model classes are tried, and what each level adds.

Level	Model	What it adds	What it costs
Model 0	persistence, seasonal persistence	the reference every later model is scored against	nothing; no weather or state information used
Model 1	linear or ridge regression on X_t	weather, calendar and state enter the forecast; coefficients are auditable	nonlinear shapes must be engineered
Model 2	gradient-boosted trees	thresholds and interactions found from data	interpretability, extrapolation, tuning effort
Model 3	recurrent or attention model	a learned representation of a long trajectory	data volume, training time, diagnosability

5.5.5 A worked flexibility forecast

The target in the aggregator's business is usually flexibility rather than load, and the same machinery applies with a different feature vector and a different target. To forecast the reduction available at 17:00 the features are

$$X_t = \big[\, \hat{T}_{17:00},\ N_{\text{avail}},\ D,\ T_{\text{in}} - T_{\text{set}},\ F_{\text{previous events}},\ \text{event duration},\ \text{hour} \,\big], \tag{5.23}$$

combining the weather forecast at the target hour, the fleet's availability and mean duty cycle, the average gap between indoor temperature and setpoint, the response delivered in previous events, the duration the request must be sustained, and the hour.

A boosted ensemble fitted to historical event records may produce a first tree with the following path. Is the forecast temperature above 31°C? If not, the predicted response is modest, because few compressors are running and there is little load to remove. If so, is the mean duty cycle above 60 %? If not, the response is medium. If so, is fleet availability high? If so, the predicted response is high. Every split in that path is physically sensible, and none of them was specified: the ensemble found them because they reduce the squared error of the piecewise-constant fit. That is the appeal of the class on this data, and Section 5.6.3 states the limit of what such a fit establishes.

5.6 Diagnostics after fitting

A fitted model is the beginning of the work rather than the end. This section sets out what to inspect once the error metric has improved, how to read the importances a tree ensemble reports, and what a predictive fit does not establish.

5.6.1 Residual inspection

The reporting of a single improved error number is not a diagnosis. The object to examine is the residual series,

$$e_t = y_t - \hat{y}_t, \tag{5.24}$$

and the question is whether it has any structure left. A residual that is unpredictable noise means the model has extracted the available signal; a residual that depends on anything the model was given, or on anything it was not given, means the opposite. Table 5.4 lists five patterns worth testing for, what each indicates and the remedy each implies. The rows are ordered by how often they appear in fleet forecasting work.

Two of the five rows deserve a comment. Residual autocorrelation, the second row, is the reason the next chapter exists: if weather, calendar and state features are all present and the errors are still correlated in time, then temporal structure remains that the feature vector did not capture, and an autoregressive model of the residual is the direct remedy. Residuals that spike during dispatch events, the third row, are a different kind of failure: the model of the fleet's *response to control* is weak, which matters more than the average error because those are exactly the hours in which the forecast is used.

5.6.2 Feature importance and its limits

Tree ensembles report an approximate importance for each feature, computed from how much the splits on that feature reduced the training error, or from how much the prediction changes when the feature's values are permuted. Figure 5.9 shows a plausible ranking for the thermostat-fleet model of equation (5.20). The horizontal axis is the share of the total importance and the bars are the features, ordered from the largest. The forecast temperature, the recent duty cycle and

Table 5.4: Residual diagnostics for a fitted fleet model, with the remedy each pattern implies.

Pattern in e_t	What it indicates	Remedy
correlated with temperature	the weather nonlinearity is not fully captured	add a knot, a spline or a split; check the saturation region
correlated in time, $\mathrm{ACF}(k) \neq 0$	temporal structure remains in the residual	model the residual dynamics, Chapter 6
spikes during dispatch events	the control-response part of the model is weak	separate event hours; add event features and a response model
grows with the horizon h	one model is being asked to serve several horizons	fit per horizon, or add h as a feature
collapses on extreme days	the live features lie outside the training range	distribution-shift monitoring, Chapter 9

the device availability dominate; the load at the same hour yesterday and the indoor-setpoint difference are moderate; humidity and the month are small.

The ranking is useful in three concrete ways. It tells the data engineer which inputs must be reliable, since a 20 % importance on a feature means a feed outage in that feature degrades the forecast materially. It tells the modeler where to invest in features, since the three large bars are also the three whose representation can most easily be improved. And it provides a sanity check against physics: if the month had come out above the forecast temperature, something would be wrong with the training set.

What the ranking does not establish is causation. Importance measures association in the training sample, conditional on the other features present. A variable can be highly predictive without influencing the outcome at all: the duty cycle is high *because* it is hot, so some of its apparent importance is borrowed from temperature, and removing temperature from the feature set would raise the duty cycle's importance without changing any physics. Correlated features also split their importance arbitrarily between them, in the same way the collinear pair of Section 5.2.5 split their coefficients.

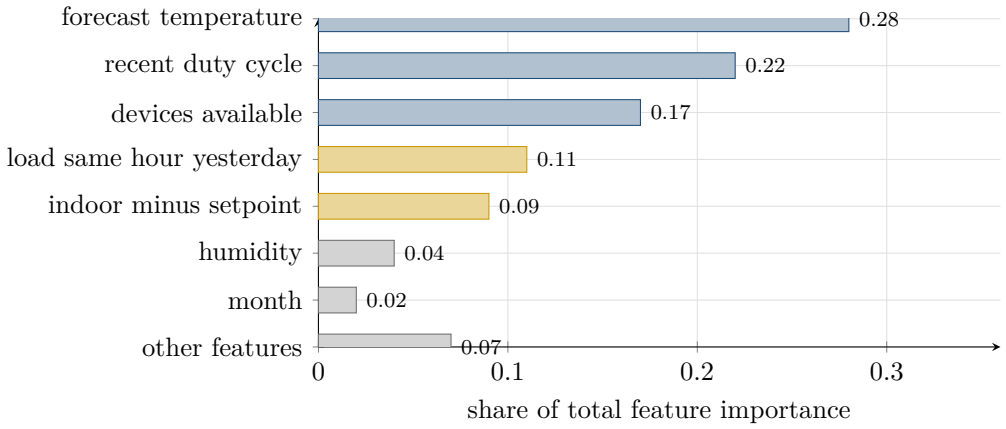


Figure 5.9: Feature importances for a boosted model of thermostat-fleet flexibility, on the feature set of equation (5.20). The three shadings separate the large, moderate and small groups discussed in Section 5.6.2. The shares are illustrative and sum to one.

5.6.3 Prediction is not causal response

The limit of the previous subsection has an operational version that matters for the aggregator's core business. Suppose a model is fitted to historical event records and learns

$$F = f(u,\ T,\ x), \tag{5.25}$$

with u the control action issued, T the outdoor temperature and x the fleet state. The historical control actions were not assigned at random: the operations team issued larger setpoint adjustments only on extremely hot days, when the grid needed the capacity. In the training data, therefore, u and T are strongly coupled.

A model fitted to that record predicts well on days that resemble the record, and it cannot separate two explanations of a large observed response: that the response was large because the control action was large, or that it was large because the day was extremely hot. The distinction is the difference between a forecast and a counterfactual. The forecast question is what the fleet will deliver tomorrow under the dispatch policy the operator has always used, and equation (5.25) answers it. The counterfactual question is what the fleet would deliver tomorrow under a dispatch the operator has never tried, and no amount of boosting answers that from this data; answering it requires either experimental variation in u or a physical model in which u enters the dynamics explicitly. The point returns when the measurement and verification of a controlled response is treated in its own right.

> **Principle 5.3.** *Gradient boosting solves prediction. It does not solve causal inference, and a feature importance is not a causal effect. Any statement of the form "the response was larger because the command was larger" requires either randomized dispatch or a model of the physics.*

Conceptual questions

"What model would you start with for thermostat load forecasting?"

A strong answer starts below the modeling question. It establishes the persistence and seasonal-persistence baselines first, because they are the reference the skill score is computed against, and then proposes two models: a regularized linear model and a gradient-boosted-tree model, both on the same feature vector of lagged load, archived weather forecasts, calendar variables and fleet-state variables. It gives the reason for the boosted model, that it handles nonlinear weather responses and feature interactions on structured data without those being specified in advance, and names the condition for going further: move to a sequence model only if held-out validation shows that predictive information remains in longer sequences which the engineered lag and state features fail to summarize.

Conceptual questions

"Why might gradient boosting outperform an LSTM for DER fleet load?"

The answer is a chain of seven statements. The available variables are rich and structured, combining weather, calendar, device state and lagged response. Much of the temporal information is representable explicitly, through P_{t-1}, P_{t-24} and P_{t-168}. What remains after those lags is nonlinearity, interactions and thresholds. Boosted trees handle exactly those three. They need less data and less training effort than a sequence model. They are easier to diagnose and to operate, since a split and an importance can be inspected. And the sequence model becomes the better choice only when a long sequential history carries information the engineered features fail to summarize, which is a question validation can settle rather than a matter of preference.

5.7 What to retain

Definition 5.1 (The structure trade)**.** A linear model predicts $\hat{y} = \mathbf{X}\boldsymbol{\beta}$ with its structure specified by the modeler through the features, and regularization replaces its objective with training error plus λ times a complexity penalty; a tree partitions the feature space into regions and predicts a constant in each, a forest averages such partitions to reduce variance, and gradient boosting builds them in sequence through $\hat{y}^{(m)} = \hat{y}^{(m-1)} + \eta\, T_m(X)$ so that each tree corrects the error the others leave. For structured energy data, a boosted ensemble on well-engineered features is one of the models any more sophisticated method, a sequence model included, has to beat.

Table 5.5 collects the chapter's objects with the subsection that defines each.

Table 5.5: The objects of this chapter.

Object	Definition	Section
$\hat{y} = \mathbf{X}\boldsymbol{\beta}$	linear model: prediction as a weighted sum of features	5.1.1
$\hat{\boldsymbol{\beta}}$	least-squares coefficients, $(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}y$	5.1.4
CD feature	engineered kink that lets a linear model bend	5.2.2
interaction term	product of two features; the effect of one depends on the other	5.2.3
ridge penalty	$\lambda \sum_j \beta_j^2$: stabilizes collinear coefficients	5.2.4
lasso penalty	$\lambda \sum_j \lvert\beta_j\rvert$: sets some coefficients to zero	5.2.6
R_j, c_j	terminal region of a tree and the constant it predicts	5.3.1
random forest	$\frac{1}{M}\sum_m T_m(X)$: variance reduction by averaging	5.3.4
boosting update	$\hat{y}^{(m)} = \hat{y}^{(m-1)} + \eta T_m(X)$ on the residual	5.4.1
η	learning rate; trades against the number of trees	5.4.3
extrapolation rule	linear model continues its trend, tree stays constant	5.5.2
model hierarchy	baseline, linear, boosted, sequence; adopt by increment	5.5.4
e_t	residual series, inspected for remaining structure	5.6.1
feature importance	split-based association, not a causal effect	5.6.2

5.8 Exercises

Exercise 5.1. Evaluate the fitted model of equation (5.3) for $P_t = 102\,\text{MW}$, $P_{t-24} = 96\,\text{MW}$, $\hat{T}_{17:00} = 27°\text{C}$ and $N_{\text{avail}} = 72\,000$. Then compute the change in the forecast caused by a 2°C error in the temperature forecast, and the change caused by a 5 % error in the available device count. Which input deserves the tighter data-quality requirement, and does that answer depend on the temperature level?

Exercise 5.2. Two standardized features satisfy $x_2 = x_1$ in the training data, and the least-squares fit returns $\beta_1 = 8$, $\beta_2 = -7.4$ while ridge returns $\beta_1 = 0.35$, $\beta_2 = 0.28$. Show that both fits give the same prediction whenever $x_2 = x_1$. Then compute each model's predicted change when a live observation has x_1 larger than x_2 by 0.5, and state the ratio of the two changes.

Exercise 5.3. Continue Table 5.1. A second tree fitted to $r^{(2)} = [-10, 0, 0, 10]$ predicts $[-10, 0, 0, 10]$; with $\eta = 0.5$, compute $\hat{y}^{(2)}$ and $r^{(3)}$. If every later tree reproduces the current residual exactly, how many further stages are needed before the largest residual falls below 1 MW, and what does that imply about the relation between η and the number of trees in Figure 5.7?

Exercise 5.4. A model of equation (5.21) is trained on temperatures from 20 to 35°C and asked for a forecast at 42°C. Using Figure 5.8, state the linear model's prediction and the boosted ensemble's prediction. Then write down the physical assumption under which each is the better answer, and propose a monitoring rule that would have flagged the forecast as an extrapolation before it was issued.

Chapter 6

Autoregressive models, ARIMA and the dynamics of forecast residuals

6.1 The residual as the starting point

Every model built so far in this book has the form $\hat{y}_{t+h} = f(X_t)$, with X_t holding recent load, the same hour on previous days, weather, calendar variables, device state and control history. Chapter 4 built that vector and Chapter 5 fitted nonlinear functions to it. The question this chapter asks is what remains afterward:

After the available predictors have been used, is there still temporal structure in the model's errors, and what should be done about it?

The answer decides whether the autoregressive family of models is useful here. If the errors of a fitted model are unpredictable from their own past, the model has extracted the temporal information in the data and nothing in this chapter will improve it. If the errors are predictable from their own past, something has been left on the table, and the vocabulary for describing and exploiting what is left is the vocabulary of autoregression, differencing and moving averages.

Jargon: residual, autocorrelation, autoregression, white noise, stationarity, exogenous

A *residual* is the realized forecast error of a fitted model, $e_t = y_t - \hat{y}_t$; the words residual and error are used interchangeably here, with "residual" preferred when the quantity is being analyzed rather than scored. *Autocorrelation* is the correlation of a series with a lagged copy of itself, $\mathrm{Corr}(y_t, y_{t-k})$; applied to residuals it measures how much one error says about the next. A series is *white noise* when it has zero mean, constant variance and zero autocorrelation at every nonzero lag; this is the ideal shape of a residual series. *Autoregression* means regressing a series on its own past values. A series is *stationary* when its statistical properties do not drift in time, which is the condition autoregressive estimates rely on. An *exogenous* variable is a predictor from outside the series itself: temperature, calendar, the number of devices available.

Key idea

A fitted model's residuals are themselves data. If they still correlate with their own past, the model is leaving temporal information unexplained, and the usual cause is a slowly varying physical state the feature vector does not contain.

This section introduces the residual series as an object of study and states the condition a good residual series satisfies. The rest of the chapter supplies the models that describe residuals when the condition fails.

6.1.1 A run of underpredictions

Consider a gradient-boosted model of an aggregator's thermostat fleet load, fitted with the feature vector

$$X_t = \big[\, T_t,\ \text{hour},\ \text{weekday},\ P_{t-24},\ N_{\text{avail}} \,\big], \tag{6.1}$$

which is a reasonable minimum: current outdoor temperature, two calendar variables, yesterday's load at the same hour, and the number of enrolled devices currently connected. On one afternoon the model's hourly errors $e_t = P_t - \hat{P}_t$ are $+3$, $+4$, $+5$, $+4$, $+6$ and $+5\,\text{MW}$ at 13:00 through 18:00.

Figure 6.1 plots that run. The horizontal axis is the hour of the day, the vertical axis is the residual in megawatts, each bar is one hourly error, and the dashed line marks zero error. Two features of the picture matter. Every bar is on the same side of zero, so the model underpredicts for six consecutive hours; and the bars have similar heights, so the size of the error is also persistent, not just its sign. A residual series that looked like white noise would cross the dashed line repeatedly and show no relation between one bar and the next.

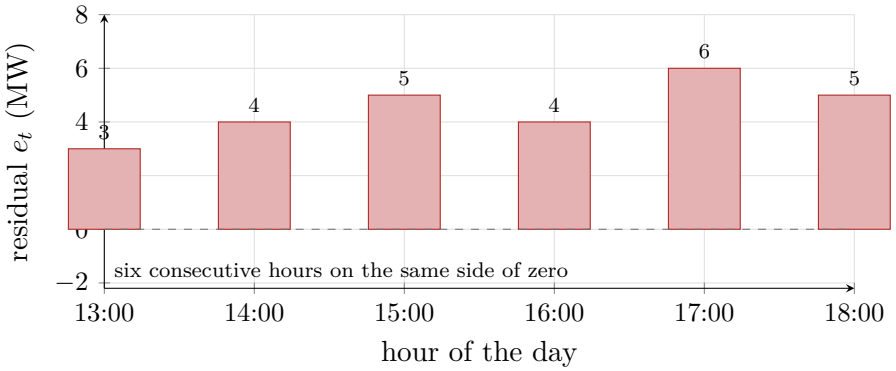


Figure 6.1: The residual run of Section 6.1.1. Bars are the hourly residuals $e_t = P_t - \hat{P}_t$ of a boosted model fitted with the features of equation (6.1); the dashed line is zero error. The values $+3, +4, +5, +4, +6, +5\,\text{MW}$ are the ones quoted in the text.

A run like this is not a scoring problem but a diagnostic. The mean absolute error over these six hours is $4.5\,\text{MW}$, which a report would record as a modest number; the informative fact is not the magnitude but the arrangement. Six independent fair coin flips land on the same side with probability $2^{-5} \approx 3\,\%$, so the pattern is not plausibly chance. Something in the afternoon's physics is pushing the load above what the model expects, and the model does not know what it is.

6.1.2 The condition a residual series should satisfy

The property to aim for is

$$\operatorname{Corr}(e_t,\ e_{t-k}) \approx 0 \qquad \text{for every lag } k \geq 1 \text{ of practical interest}, \tag{6.2}$$

where "practical interest" means the lags a forecaster could actually exploit: one step, a few steps, one day, one week. The reasoning behind equation (6.2) is an argument about information rather than about statistics. Suppose e_t predicts e_{t+1} with correlation 0.7. At time t the residual e_t is known, because y_t has been observed and $\hat{y}_t$ was computed; so a quantity available at issue time carries information about an error the model is about to make. A forecast that ignores available information is not the best forecast obtainable from the information set $\mathcal{I}_t$, which is the standard set in Chapter 1.

The size of the loss is easy to quantify under a linear correction. If the residual series has variance σ_e^2 and lag-one correlation ρ_1, then subtracting the best linear prediction $\rho_1 e_{t-1}$ from the next residual leaves variance

$$\operatorname{Var}(e_t - \rho_1 e_{t-1}) = \sigma_e^2 \,(1 - \rho_1^2), \tag{6.3}$$

so $\rho_1 = 0.7$ corresponds to a 51 % reduction in error variance, or 30 % in root mean square error, obtainable without a single new input. That potential gain is the central motivation for autoregressive models, and equation (6.3) is also the reason the diagnostic matters more as ρ_1 grows: at $\rho_1 = 0.2$ the recoverable variance is 4 %, which is rarely worth a second model.

Principle 6.1. *Residual autocorrelation is a statement about the model, not about the data. It says that information already inside the information set is not being used.*

6.2 Autoregressive models

This section defines the autoregressive model, interprets its coefficient, extends it to several lags, identifies it with ordinary regression on lagged values, and then states why it is insufficient on its own for an energy load series.

6.2.1 The first-order autoregressive model

The first-order autoregressive model, written AR(1), predicts the current value of a series from the immediately preceding value,

$$y_t = c + \phi\, y_{t-1} + \varepsilon_t, \tag{6.4}$$

with c an intercept, ϕ the autoregressive coefficient, and ε_t a white-noise innovation: the part of y_t that could not have been anticipated from the past of the series. The forecast implied by equation (6.4) is obtained by dropping the innovation, whose expected value is zero, $\hat{y}_t = c + \phi\, y_{t-1}$.

As an instance, take $y_t = 20 + 0.8\, y_{t-1} + \varepsilon_t$ with $y_{t-1} = 100$. The forecast is $\hat{y}_t = 20 + 0.8 \times 100 = 20 + 80 = 100$. The arithmetic also exposes a property of the model worth naming immediately. The value 100 reproduces itself, because it is the fixed point $\mu = c/(1-\phi) = 20/0.2 = 100$: the unconditional mean of the process. An AR(1) series is a series that is pulled back toward μ at a rate set by ϕ and pushed away from it by the innovations.

6.2.2 The meaning of the coefficient

The single coefficient ϕ controls how long a disturbance survives. Writing the model in deviations from the mean, $\tilde{y}_t = y_t - \mu$, equation (6.4) becomes $\tilde{y}_t = \phi \tilde{y}_{t-1} + \varepsilon_t$, and in the absence of further innovations a deviation decays geometrically,

$$\tilde{y}_k = \phi^k\, \tilde{y}_0. \tag{6.5}$$

Three regimes follow. At $\phi = 0$ the previous value contributes nothing and the series is white noise about its mean. At $\phi = 0.9$ persistence is strong: a deviation retains 90 % of its size after one step and still 43 % after eight. At $\phi = -0.5$ the sign alternates, so a positive deviation tends to be followed by a negative one, which is the signature of an overshooting or oscillating process.

Figure 6.2 draws equation (6.5) for those three values of ϕ. The horizontal axis counts steps after a single 10 MW deviation, the vertical axis is the surviving deviation in megawatts, the

dashed line is zero deviation, and the three marked paths are the three coefficients. The $\phi = 0.9$ path decays slowly and never changes sign; the $\phi = 0.5$ path is at 0.6 MW after four steps; the $\phi = -0.5$ path crosses zero at every step while shrinking at the same rate as the $\phi = 0.5$ path. The half-life of a deviation is $\ln(1/2)/\ln|\phi|$, which is 6.6 steps at $\phi = 0.9$ and one step at $|\phi| = 0.5$.

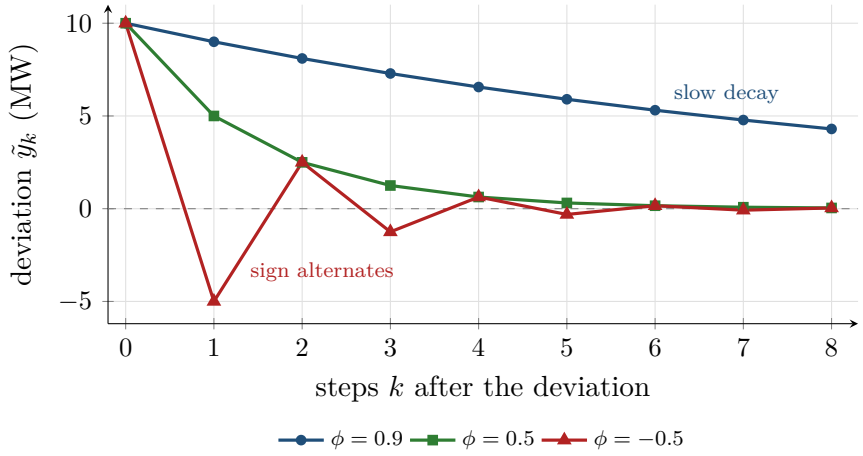


Figure 6.2: Decay of a single deviation under AR(1), from equation (6.5) with $\tilde{y}_0 = 10$ MW: $\tilde{y}_k = \phi^k \tilde{y}_0$ for $\phi = 0.9$, 0.5 and -0.5. Markers are the integer steps $k = 0, \ldots, 8$; the dashed line is zero deviation. Values are illustrative.

For energy demand, positive coefficients at short lags are the normal case, and the reason is physical rather than statistical. The states that determine load are thermal masses, occupancy patterns and device duty cycles, all of which change gradually relative to a fifteen-minute or hourly sampling step. A building that is warm now is warm in fifteen minutes; a fleet whose compressors are mostly running now has mostly running compressors in fifteen minutes. Negative short-lag coefficients do occur, and when they do they usually indicate a control loop overshooting or a metering artifact rather than a property of demand.

6.2.3 The autoregressive model of order p

Extending the model to p lags gives AR(p),

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t, \tag{6.6}$$

of which a third-order instance is

$$y_t = 5 + 0.6\, y_{t-1} + 0.2\, y_{t-2} + 0.1\, y_{t-3} + \varepsilon_t. \tag{6.7}$$

The coefficients in equation (6.7) decline with lag, which is the usual pattern for a smoothly persistent series: most of the predictive content sits at lag one, with the remaining lags supplying corrections. Their sum, 0.9, is the long-run multiplier of the process, and the unconditional mean is $c/(1 - \sum_i \phi_i) = 5/0.1 = 50$.

Two practical points follow from equation (6.6). First, the order p is a modeling choice rather than a property of the data, and raising it always improves the in-sample fit while eventually degrading the forecast; Chapter 9 supplies the machinery for choosing it honestly. Second, the lags need not be consecutive. An hourly load model will often use lags 1, 2, 24 and 168 and leave the rest at zero, because the physical mechanisms that create dependence at one hour, one day and one week are different mechanisms. A model of that shape is formally AR(168) with 164 coefficients fixed at zero, which is a useful reminder that the order is bookkeeping and the chosen lags are the substance.

6.2.4 Autoregression as regression on lagged values

Equation (6.6) is a linear regression whose predictors happen to be past values of the target. Setting $X_t = [\, y_{t-1},\ y_{t-2},\ y_{t-3} \,]$ and fitting by least squares gives

$$\hat{y}_t = \beta_0 + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \beta_3 y_{t-3}, \tag{6.8}$$

which is equation (6.7) with the coefficients renamed. Nothing in the fitting procedure distinguishes the two. Any of the estimators of Chapter 5 can be applied to a lag matrix, and a boosted ensemble fitted to $[\, y_{t-1}, y_{t-24}, y_{t-168} \,]$ is an autoregressive model with a nonlinear link.

The difference between the two descriptions is one of emphasis, and the emphasis has consequences. A regression practitioner treats lagged values as columns of a design matrix and asks about fit and generalization. A time-series practitioner treats them as a dynamic system and asks four further questions: whether the series is stationary, whether the residuals are autocorrelated, which lags are structurally relevant, and what the fitted coefficients imply about how long a disturbance survives. Those questions are the subject matter of this chapter, and they are worth asking of a lag-feature regression too. The practical warning is that ordinary standard errors and significance tests assume uncorrelated residuals; if the residuals of equation (6.8) are autocorrelated, the coefficients may still forecast usefully while the reported uncertainties are wrong.

6.2.5 Why autoregression alone is insufficient for load

An energy load series is not driven only by its own history. The fleet load of an aggregator responds to outdoor temperature, to the calendar, to how many devices are connected and enrolled, and to the control actions the aggregator itself has dispatched. A purely autoregressive model sees none of those variables, and it compensates by absorbing their effects into its lag coefficients.

The failure this produces is specific: the model confuses cause with persistence. If temperature changes slowly, then P_{t-1} is a good proxy for T_{t-1}, which is a good proxy for T_t, so a fitted AR(1) will report a large ϕ and forecast well on ordinary days. What it has learned is $P_t \approx P_{t-1}$, with the temperature dependence hidden inside that relation rather than represented. The model then fails on exactly the days that matter. A cold front arriving overnight changes T_t by eight degrees; load changes with it; and the autoregressive model, which knows only that load was high yesterday, carries yesterday's level forward. The same argument applies to a holiday, to a sudden change in the number of available devices, and to a dispatch event, each of which breaks the proxy relation the model is relying on.

The conclusion is not that autoregression is useless but that it is a component. Autoregressive structure captures the persistence of states the model cannot see; exogenous variables capture the drivers it can. The next subsection combines them.

6.2.6 Autoregression with exogenous variables

Adding predictors to equation (6.6) gives the model usually written ARX, autoregressive with exogenous inputs,

$$y_t = c + \sum_{i=1}^{p} \phi_i \, y_{t-i} + \boldsymbol{\beta}^{\mathsf{T}} X_t + \varepsilon_t, \tag{6.9}$$

in which the first sum carries temporal persistence and the second term carries the physical and calendar drivers. For the thermostat fleet a compact and defensible instance is

$$P_t = c + \phi_1 P_{t-1} + \phi_{24} P_{t-24} + \beta_1 T_t + \beta_2 N_{\text{avail}} + \varepsilon_t, \tag{6.10}$$

with four interpretable coefficients. The term $\phi_1 P_{t-1}$ represents short-term persistence of the fleet's thermal and duty-cycle state; $\phi_{24} P_{t-24}$ represents the daily cycle through yesterday's value at the same hour; $\beta_1 T_t$ represents the cooling demand created by outdoor temperature; and $\beta_2 N_{\text{avail}}$ scales the whole fleet by how many devices are actually connected.

Equation (6.10) is the smallest model in this chapter that a practitioner would be willing to run, and it is worth noticing what it still lacks. It is linear in temperature, whereas Chapter 4 showed the load–temperature relation to be V-shaped and interacted with hour of day; and it has no term for the aggregator's own dispatch history, so a curtailment and the rebound that follows it appear to the model as unexplained innovations. Those two gaps are what a boosted ensemble with engineered features repairs, which is why the hybrid of the next section is built on the ensemble rather than on equation (6.10).

6.3 Autoregressive residual correction

Rather than replacing a machine-learning model with an autoregressive one, the two can be composed: the ensemble predicts the load, and an autoregressive model predicts the ensemble's error. This section builds that composition, works a numerical example, explains physically why residual autocorrelation arises, and draws the distinction between compensating for a missing state and representing it.

6.3.1 Correcting a model with its own residuals

Let $\hat{y}_t^{\text{ML}} = f(X_t)$ be the prediction of the fitted machine-learning model and $e_t = y_t - \hat{y}_t^{\text{ML}}$ its realized residual. Treat the residual series as a time series in its own right and fit an autoregressive model to it. If the fitted relation is

$$e_t = 0.7\, e_{t-1} + \varepsilon_t, \tag{6.11}$$

then the forecast of the next residual is $\hat{e}_t = 0.7\, e_{t-1}$, and the corrected forecast is the sum

$$\hat{y}_t = \hat{y}_t^{\text{ML}} + \hat{e}_t. \tag{6.12}$$

Three conditions make the construction legitimate. The residual e_{t-1} must be computable at issue time, which requires that y_{t-1} has been metered and delivered; the residuals used to fit equation (6.11) must come from out-of-sample predictions, because in-sample residuals of a flexible model are artificially small and artificially uncorrelated; and the correction must be validated by horizon, for the reason developed in Section 6.6.5.

6.3.2 A worked correction

Take the fleet at 16:00 with a forecast required for 17:00. The boosted model returns $\hat{P}_{17:00} = 100\,\text{MW}$. The most recent residual, computed once the 16:00 meter reading arrived, is $e_{16:00} = +8\,\text{MW}$: the fleet drew 8 MW more than the model expected. The residual series of this fleet has historically satisfied $e_t \approx 0.6\, e_{t-1}$, so

$$\hat{e}_{17:00} = 0.6 \times 8 = 4.8\,\text{MW}, \qquad \hat{P}^{\text{corr}}_{17:00} = 100 + 4.8 = 104.8\,\text{MW}. \tag{6.13}$$

The two halves of equation (6.13) do different jobs. The tree ensemble has already accounted for weather, calendar and reported device state, and it supplies the 100 MW. The autoregressive correction accounts for the short-term bias that those features do not explain, and it supplies the 4.8 MW. Neither half can produce the other: the ensemble cannot see its own error, and the autoregressive term has no idea what the temperature is.

This is the practical hybrid, and its appeal is that it is cheap. The correction is one coefficient fitted to a residual series, it requires no new data source, and it can be disabled by setting one number to zero if the residual structure disappears. Its limitation is equally plain: the correction is extrapolating a bias, so it is only as good as the assumption that the bias persists over the horizon being forecast.

6.3.3 Why residual autocorrelation arises

The mechanism is almost always a missing state. Suppose the model predicts

$$P_t = f(T_t,\ \text{hour},\ \text{weekday}), \tag{6.14}$$

while the physical load also depends on the thermal state of the buildings, the aggregate temperature of their mass, $x_t = T_{\text{mass},t}$. That state is not in equation (6.14) and is not metered. It also changes slowly, because the thermal time constant of a building envelope is hours rather than minutes.

Those two facts together produce the run of Figure 6.1. If the building mass is warmer than the model's implicit assumption, the air conditioners run longer than the model expects, so the load is higher than predicted. Fifteen minutes later the mass is still warmer, so the load is still higher; thirty minutes later the same. The missing state imprints its own time constant on the error series, and a slowly varying state produces slowly varying, therefore autocorrelated, residuals. The general reading is worth stating as a rule.

Principle 6.2. *Residual autocorrelation is usually the signature of a persistent state the model does not represent. The time constant of the residual correlation is an estimate of the time constant of the missing state.*

6.3.4 Diagnosing a flexibility model's residuals

The same analysis applied to a flexibility forecast is where the diagnostic earns its place, because the missing states are nameable. Suppose a boosted model predicts fleet flexibility, $\hat{F}_t = f(X_t)$, and the autocorrelation function of its out-of-sample residuals is

$$\rho_1 = 0.65, \qquad \rho_2 = 0.42, \qquad \rho_3 = 0.18. \tag{6.15}$$

The pattern in equation (6.15) is a decay over roughly three steps, so the missing information is short-lived state rather than a persistent level error. Four candidates fit that description: indoor temperature, which is incompletely observed across the fleet; recent HVAC cycling, which determines how much of the load is curtailable right now; recent dispatch events, which deplete the available response; and rebound state, which follows a dispatch and raises load while reducing flexibility.

Two responses are available. Option A is to engineer better state features: add the observed indoor temperatures where they exist, add a duty-cycle estimate, add the time since the last dispatch and its depth. Option B is to add an autoregressive correction on the residuals as in equation (6.12). Option A should be investigated first, for two reasons. It usually reduces the error more, because a feature that represents the state works at every horizon whereas a residual correction decays with horizon; and the investigation itself is informative, because discovering which added feature removes the autocorrelation identifies the physical mechanism that was missing.

6.3.5 Compensation is not explanation

The distinction behind that recommendation is worth isolating. If the residuals are autocorrelated because indoor temperature was omitted, then adding e_{t-1} as a predictor may well improve accuracy, and it does so by using the error as a noisy, lagged, one-dimensional proxy for the omitted state. It does not identify the state, it cannot extrapolate it, and it inherits none of its physics. Three consequences follow. The corrected model gives no insight into what drives the fleet, so it cannot be interrogated when it fails. It breaks under regime change, because the relation between the proxy and the state was estimated under historical conditions. And it degrades with horizon, because the proxy is a lagged quantity while the state keeps evolving.

An autoregressive term can therefore compensate for a missing state without explaining it. That is an acceptable trade when the state is genuinely unobservable, and a poor one when the state is merely unmeasured through inattention.

6.3.6 The hybrid forecaster

Putting the pieces together gives the composite forecast

$$\hat{y}_t = f_{\mathrm{ML}}(X_t) + f_{\mathrm{AR}}(e_{t-1}, e_{t-2}, \dots), \tag{6.16}$$

whose two terms separate the structure that is explained from the structure that is merely described. Figure 6.3 shows the arrangement. The boxes on the left are the two input groups: the feature vector built from weather, calendar, device state and load history, and the recent residual history. The upper path sends the features through the gradient-boosted ensemble to the base prediction; the lower path sends the residuals through the autoregressive correction; the two meet at a summing node that emits the final forecast. The dashed arrow closes the loop: once the meter reading for t arrives, the realized residual is appended to the residual history that the lower path will use at $t + 1$.

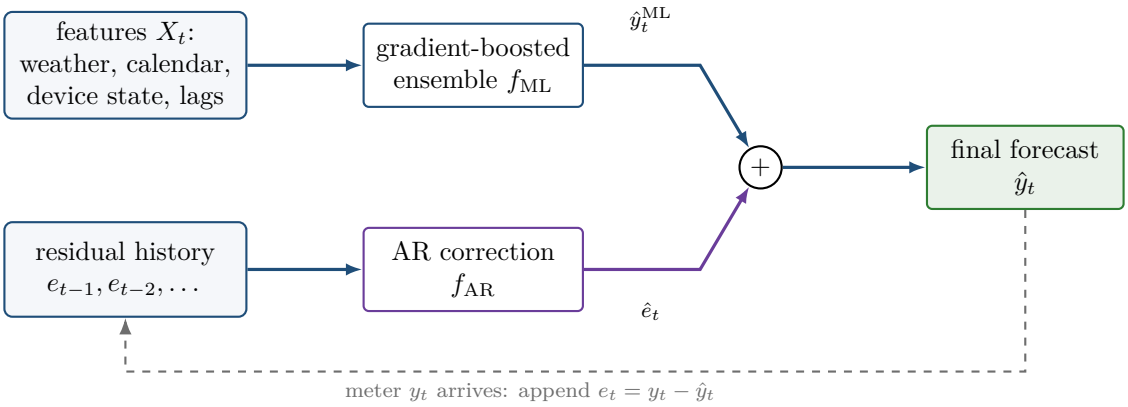


Figure 6.3: The hybrid forecaster of equation (6.16). Features enter a boosted ensemble to give the base prediction; the recent residual history enters an autoregressive correction; the sum is the issued forecast. The dashed path is the update that makes the next residual available.

The architecture has one further merit beyond accuracy: it is auditable. The two terms of equation (6.16) can be logged separately, so an operator can see at a glance how much of a given forecast is explanatory and how much is a bias correction. A correction term that grows month over month is evidence that a feature is missing, which turns the hybrid into a monitoring instrument as well as a forecaster.

6.4 Stationarity, differencing and the ARIMA family

Autoregressive estimates assume that the series being modeled has stable statistical properties. This section defines that condition, shows the two transformations used to obtain it, and assembles the ARIMA family from the autoregressive, integrated and moving-average parts.

6.4.1 Stationarity

A series is stationary, in the weak sense used throughout applied forecasting, when two conditions hold:

$$\mathbb{E}[y_t] = \mu \text{ constant in } t, \qquad \operatorname{Cov}(y_t,\ y_{t-k}) = \gamma_k \text{ depending on } k \text{ but not on } t. \tag{6.17}$$

In words: the level does not drift, and the dependence between two observations is governed by how far apart they are, not by when they occurred. Strict stationarity is a stronger statement about the whole joint distribution, and is not needed here.

The condition matters because equation (6.6) has coefficients that do not depend on t. Fitting a single set of ϕ_i to a series whose level is moving forces those coefficients to absorb the movement. Consider the random walk with drift

$$y_t = y_{t-1} + 5 + \varepsilon_t, \tag{6.18}$$

which rises by about 5 units per step forever and has no unconditional mean at all. Fitting AR(1) to a sample of equation (6.18) returns $\hat{\phi}$ close to one, and the model then reports enormous persistence: a deviation that never decays. The reported persistence is an artifact of the trend, and forecasts built on it inherit the artifact, becoming confidently wrong as soon as the drift rate changes.

Energy load series routinely violate equation (6.17). They contain annual and daily seasonality, so the mean depends on t; they contain long-run growth or decline in the customer base; an aggregator's fleet series contains enrollment growth, which changes the level month over month; and weather drives episodes of elevated demand that last days. The raw series is therefore not a safe input to an autoregressive fit, and the standard remedy is to transform it.

6.4.2 Differencing

First differencing replaces the series by its step-to-step changes,

$$\Delta y_t = y_t - y_{t-1}. \tag{6.19}$$

A fleet load of 80, 84, 87, 89 MW at 14:00 through 17:00 becomes $\Delta y = [\,4,\ 3,\ 2\,]$ MW: the modeling target is no longer the level but how much the load changes from one step to the next. Applied to the drifting series of equation (6.18), differencing gives $\Delta y_t = 5 + \varepsilon_t$, which is stationary; the drift has become a constant mean and the trend has been removed exactly.

Figure 6.4 shows the transformation on a constructed example. The upper panel plots the level against the step index: it rises from 80 to 122 MW with no stable mean, and a horizontal line could not be drawn through it. The lower panel plots the first difference of the same series against the same axis, together with its mean of 3.5 MW as a dashed line: the differenced series fluctuates about that fixed level and shows no trend. The axes are deliberately different in scale, because the point of the figure is the qualitative change, from a series with no mean to a series with one.

Differencing is not free. It amplifies measurement noise, because the difference of two noisy values has twice the noise variance of one; it removes the level information the forecast may actually need, so a differenced forecast must be integrated back; and over-differencing a series that was already stationary introduces negative autocorrelation that was not there. The number of differences d is therefore a decision, and for most energy series $d \in \{0, 1\}$.

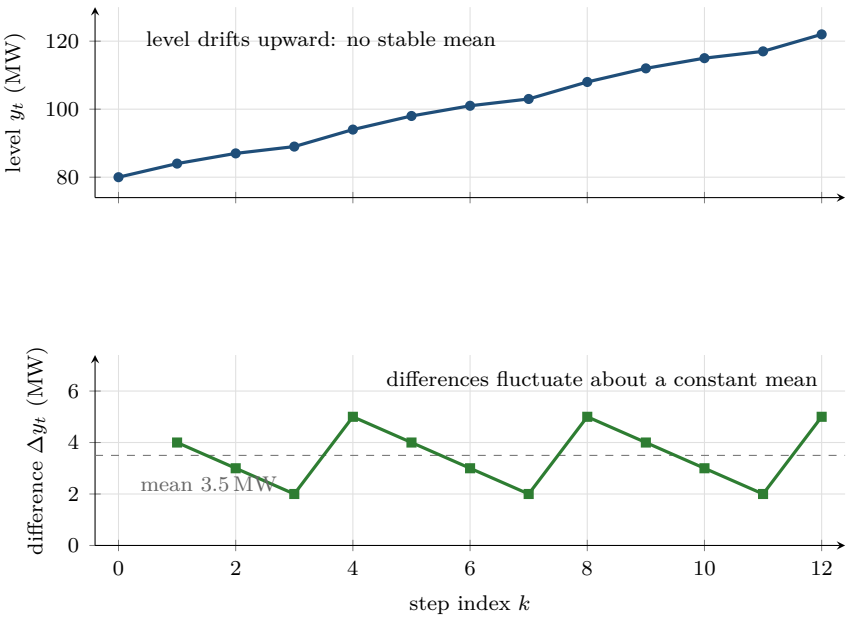


Figure 6.4: First differencing. Upper panel: a load series whose first four values are the $80, 84, 87, 89$ MW of Section 6.4.2 and whose later increments cycle through $4, 3, 2, 5$ MW. Lower panel: its first difference $\Delta y_t = y_t - y_{t-1}$ from equation (6.19), with the mean increment of 3.5 MW dashed. The construction is illustrative; the qualitative change is the point.

6.4.3 Seasonal differencing

First differencing removes a trend but not a repeating cycle, because a daily cycle makes $y_t - y_{t-1}$ itself periodic. The transformation that removes a cycle of period s is the seasonal difference,

$$\Delta_s y_t = y_t - y_{t-s}, \qquad \Delta_{24} y_t = y_t - y_{t-24}, \qquad \Delta_{168} y_t = y_t - y_{t-168}, \tag{6.20}$$

written here for hourly data. The quantity $\Delta_{24} y_t$ answers a question an operator asks naturally: how does today's load at this hour differ from yesterday's at the same hour? The quantity $\Delta_{168} y_t$ asks the same question against last week, which also removes the weekday-weekend contrast, since the same hour one week ago falls on the same day of the week.

Seasonal differencing is a strong transformation and should be recognized as the seasonal persistence baseline of Chapter 3 in disguise: the forecast $\hat{y}_t = y_{t-24}$ is exactly the claim that $\Delta_{24} y_t$ has mean zero. Modeling $\Delta_{24} y_t$ is therefore modeling the error of that baseline, which connects this chapter's residual framing to the baselines chapter: a seasonal ARIMA model on a load series and an autoregressive correction on a seasonal-persistence baseline's residual are the same object written two ways.

6.4.4 The integrated part of ARIMA

The acronym ARIMA expands to AutoRegressive Integrated Moving Average, and its three orders are written ARIMA(p, d, q): p autoregressive lags, d differencing operations, q moving-average lags. The "integrated" label refers to the differencing: a series that must be differenced d times to become stationary is said to be integrated of order d, because the original series is recovered by summing, that is integrating, the differenced one.

The orders compose in a fixed way. ARIMA$(1, 0, 0)$ is the AR(1) of equation (6.4); ARIMA$(0, 1, 0)$ is the random walk; ARIMA$(1, 1, 0)$ fits an AR(1) to the first differences; and ARIMA$(2, 1, 1)$ fits two autoregressive lags and one moving-average lag to the first differences. The notation is more intimidating than the content, which is three independent choices: how many past values, how many differences, how many past innovations.

6.4.5 The moving-average part

The moving-average component of ARIMA is not a rolling average of past observations. The rolling mean $(y_t + y_{t-1} + y_{t-2})/3$ of Chapter 4 is a feature; the moving-average component of a time-series model is something else, a regression on past *innovations*. The first-order case, MA(1), is

$$y_t = \mu + \varepsilon_t + \theta\,\varepsilon_{t-1}, \tag{6.21}$$

which says that the current value is the mean, plus today's unpredictable shock, plus a fraction θ of yesterday's shock. The shocks are unobserved, so fitting equation (6.21) requires inferring them, which is why moving-average models are estimated by likelihood methods rather than by least squares on observed columns.

The mechanism it represents is the propagation of a surprise. Suppose a weather front arrives two hours earlier than the forecast said it would, and the fleet load at 15:00 exceeds the prediction by 10 MW, so $\varepsilon_{15:00} = +10$ MW. The front does not vanish at 16:00. Its effect on the following hours is not the effect of the load level but the effect of the original surprise, and the model $y_t = \mu + \varepsilon_t + 0.5\,\varepsilon_{t-1}$ encodes exactly that: half of the 15:00 surprise is carried into 16:00, and nothing of it reaches 17:00. A moving-average term therefore gives a disturbance a finite, sharply bounded memory, where an autoregressive term gives it an infinite geometric one.

Autoregressive and moving-average dependence. The two components answer different questions about what carries information forward. In an autoregressive model the previous *value* of the process affects the current one, $y_t \leftarrow y_{t-1}$; in a moving-average model the previous *unexpected shock* affects the current one, $y_t \leftarrow \varepsilon_{t-1}$. The practical difference is the shape of the memory. An AR(1) disturbance decays forever at rate ϕ by equation (6.5); an MA(1) disturbance lasts exactly one step and then is gone. Persistent physical state is naturally autoregressive; one-off events whose influence fades quickly, such as a mistimed weather front or a metering outage, are naturally moving-average.

6.4.6 ARMA and ARIMA

Combining the two components for a stationary series gives ARMA(p, q),

$$y_t = c + \sum_{i=1}^{p} \phi_i\, y_{t-i} + \varepsilon_t + \sum_{j=1}^{q} \theta_j\, \varepsilon_{t-j}, \tag{6.22}$$

and applying equation (6.22) to a d-times differenced series gives ARIMA(p, d, q). The combination is used because the two memory shapes coexist in real series: the autoregressive terms represent persistent state, the moving-average terms represent the decaying influence of shocks, and a model with both can often match an observed dependence structure with fewer coefficients than either alone would need.

Equation (6.22) can also be extended with the exogenous term of equation (6.9), giving the model statistical software calls ARIMAX or, with seasonal orders, SARIMAX. For energy work that extension is not optional. A model of fleet load without a temperature term is not a candidate, however carefully its p, d and q were chosen.

6.4.7 Seasonal ARIMA

The seasonal extension duplicates the three orders at the seasonal period, giving

$$\text{ARIMA}(p, d, q)(P, D, Q)_s, \tag{6.23}$$

in which (p, d, q) act at consecutive lags and (P, D, Q) act at multiples of the period s. For hourly energy data the two relevant periods are $s = 24$ for the daily cycle and $s = 168$ for

the weekly one; for daily data, $s = 7$. A model written ARIMA$(1,0,1)(1,1,1)_{24}$ therefore has one ordinary autoregressive lag, one ordinary moving-average lag, one seasonal difference at 24 hours, one seasonal autoregressive term at lag 24 and one seasonal moving-average term at lag 24.

The notation does not need to be memorized. The idea that matters is that ARIMA can encode short-term dependence and seasonal dependence explicitly, in separate coefficients, rather than relying on a feature engineer to supply the right lags. That explicitness is the family's genuine advantage, and Section 6.6.1 identifies the cases in which it pays.

6.5 The autocorrelation and partial autocorrelation functions

Two diagnostic functions summarize how a series relates to its own past. This section defines them, reads them for a load series, and distinguishes their classical use in model selection from the use a modern practitioner has for them.

6.5.1 The autocorrelation function

The autocorrelation function reports the correlation of a series with itself at each lag,

$$\mathrm{ACF}(k) = \rho_k = \mathrm{Corr}(y_t,\ y_{t-k}) = \frac{\gamma_k}{\gamma_0}, \tag{6.24}$$

with γ_k the autocovariance of equation (6.17). For an hourly fleet load series the values at the lags that matter are $\rho_1 = 0.95$, $\rho_2 = 0.88$, $\rho_3 = 0.81$, $\rho_{24} = 0.82$, $\rho_{48} = 0.76$ and $\rho_{168} = 0.70$.

Figure 6.5 plots those six numbers as a stem plot. The horizontal axis lists the lags quoted above and is therefore not linear in lag; the vertical axis is the correlation; each stem with its marker is one value, printed above the marker. Three readings follow directly. The values at lags 1 to 3 are the largest, so recent history is the strongest single predictor, which is why persistence is a hard baseline to beat at short horizons. The value at lag 24 is 0.82, higher than the value at lag 3, so yesterday at the same hour carries more information than three hours ago: the daily cycle is real structure and not an artifact of smoothness. And the value at lag 168 is still 0.70, so the weekly cycle survives a full week of intervening variation, which is the weekday-weekend pattern asserting itself.

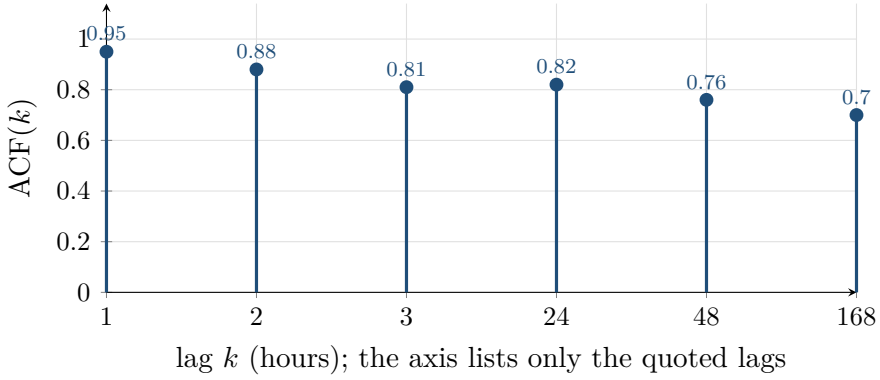


Figure 6.5: The autocorrelation function of an hourly fleet load series at the six lags quoted in Section 6.5.1, computed from equation (6.24). The horizontal axis is categorical, not linear in lag, so that the three regions can be shown together.

6.5.2 The shape of a load series' autocorrelation function

Plotted over all lags rather than the selected six, the function of a load series has a characteristic shape with three components: a high value at lag one that decays smoothly over the first few lags; local peaks at lags 24, 48, 72 and onward, each slightly lower than the last; and a further peak near lag 168. Each component is diagnostic of a different structure. The smooth decay at short lags is persistence, a continuous physical process sampled frequently. The peaks at multiples of 24 are daily seasonality, and their presence is the evidence that a seasonal term, or a y_{t-24} feature, belongs in the model. The peak near 168 is weekly seasonality, carried by the weekly rhythm of occupancy and business activity. This is the same decomposition Chapter 2 obtained from the shape of the load curve itself; the autocorrelation function reaches it from the dependence structure instead.

One caution belongs with the shape. A high autocorrelation at lag 24 does not mean that y_{t-24} causes y_t; it means that both respond to the same daily driver, namely the cycle of occupancy, solar gain and temperature. The function measures association, and the physical reading has to be supplied separately.

6.5.3 The partial autocorrelation function

The autocorrelation function has a known deficiency: its value at lag 2 is inflated by the relation at lag 1. If y_t depends on y_{t-1} and y_{t-1} depends on y_{t-2}, then y_t and y_{t-2} are correlated even when y_{t-2} adds nothing once y_{t-1} is known. Under a pure AR(1) process this is exactly what happens, and the autocorrelation function decays as $\rho_k = \phi^k$ with no lag having direct influence beyond the first.

The partial autocorrelation function removes the intervening dependence. It reports the correlation remaining after conditioning on the shorter lags,

$$\mathrm{PACF}(k) = \mathrm{Corr}\big(y_t,\ y_{t-k} \mid y_{t-1}, \ldots, y_{t-k+1}\big), \qquad \mathrm{PACF}(2) = \mathrm{Corr}\big(y_t,\ y_{t-2} \mid y_{t-1}\big), \quad (6.25)$$

and answers the question a modeler actually has: is lag 2 still useful once lag 1 has been accounted for? The contrast is worth stating plainly. The autocorrelation function measures total association at a lag, direct and indirect together; the partial autocorrelation function measures the direct, incremental association at that lag. For a pure AR(p) process the partial function is nonzero at lags 1 through p and zero beyond, which makes it the natural instrument for reading off an autoregressive order.

6.5.4 From order selection to residual diagnostics

The two functions carry historical weight because of the Box–Jenkins methodology, which built model identification on them: the partial function identifies p, since it cuts off after lag p for a pure autoregressive process; the autocorrelation function identifies q, since it cuts off after lag q for a pure moving-average process; and a slowly decaying autocorrelation function indicates that differencing is required. For decades this was how p, d and q were chosen, by eye, from two plots.

That use is now largely obsolete. Order selection is done by information criteria or by rolling-origin validation over a grid of candidate orders, which is both more reliable and automatable. The functions survive for a different and more valuable purpose: as diagnostics on the *residuals* of a fitted model, whatever family the model belongs to. Computing the autocorrelation function of the out-of-sample residuals of a boosted ensemble, a linear regression or a recurrent network asks one question of all three: has the model left temporal information unexplained? A residual autocorrelation function that is flat near zero says the temporal structure has been captured. A large value at lag one says a short-lived state is missing. A spike at lag 24 says the daily

cycle has not been fully represented, which usually means a missing or badly encoded calendar feature. A spike at lag 168 points at the weekly pattern in the same way.

Principle 6.3. *Use the autocorrelation and partial autocorrelation functions on residuals, not on the raw series. On the raw series they select orders for one model family; on the residuals they audit any model at all.*

6.6 Model comparison, multi-step forecasting and horizon

This section places the autoregressive family against the alternatives, sets out the two ways a multi-step forecast can be constructed, and establishes that the value of autoregressive information depends on the horizon.

6.6.1 ARIMA and gradient boosting

The two families are suited to different problems, and Table 6.1 lists the conditions that favor each along with the reason. Reading down the first column, ARIMA is attractive when the series is regular and well behaved, when temporal structure dominates the variation, when few external predictors are available or needed, when linear dependence is an adequate description, and when the interpretability of the lag structure is itself of value. Reading down the second, boosting is attractive when many structured predictors are available, when the relationships with those predictors are nonlinear, when interactions and thresholds matter, and when the history can be summarized adequately by engineered lag features.

Table 6.1: Conditions favoring an ARIMA model and conditions favoring a gradient-boosted ensemble.

ARIMA is attractive when	Boosting is attractive when	Why
the series is regular and stable	many structured predictors exist	ARIMA has to find structure inside the series; boosting can be told about it
temporal structure dominates the variation	weather relationships are nonlinear	linear lag terms cannot bend; trees can
few external predictors are available	interactions and thresholds matter	ARIMA adds them one linear term at a time; trees split on them
linear dependence is adequate	history is summarized well by lag features	a lag feature in a tree is autoregression with a nonlinear link
the lag structure itself is informative	accuracy outweighs coefficient interpretation	coefficients of a stationary ARIMA have a direct reading

For an aggregator's thermostat fleet the second column dominates. The load depends on temperature nonlinearly and in interaction with hour of day; humidity, solar gain and the number of available devices are all measured; and dispatch events introduce thresholds no linear model represents. Boosting with good features is therefore the stronger starting point, and ARIMA's place in that program is as a benchmark and as the residual model of Section 6.3.

6.6.2 ARIMA, recurrent networks and feature-engineered models

Against a recurrent network the comparison is about how temporal state is represented. ARIMA states the temporal structure explicitly, in a handful of coefficients, with linear relationships among a small number of lags; the structure is written down and can be read. A recurrent network of the kind Chapter 7 develops learns a nonlinear latent state, $\mathbf{h}_t = f(X_t, \mathbf{h}_{t-1})$, which summarizes an arbitrarily long history in a vector whose components have no assigned meaning. The network is the better instrument when the temporal dependence is too complicated to be written as a lag equation, for example when the effect of a dispatch depends on how long the fleet has been in its current regime; it costs more data, more computation and all of the interpretability.

Against a feature-engineered machine-learning model the comparison is about where the information lives. Suppose the true relation is

$$P_{t+3\mathrm{h}} = f\big(P_t,\ P_{t-24},\ \hat{T}_{t+3\mathrm{h}},\ \text{hour},\ \text{weekday}\big), \tag{6.26}$$

which is a realistic shape for a three-hour-ahead fleet forecast. A tree ensemble accommodates equation (6.26) directly: the right-hand side is a feature vector and $\hat{y} = f(X_t)$ is what the ensemble computes. ARIMA accommodates it awkwardly, because most of the information on the right-hand side is not in the output history at all. The forecast temperature $\hat{T}_{t+3\mathrm{h}}$ is a future exogenous input, the calendar terms are deterministic covariates, and only P_t and P_{t-24} are lags of the target. A seasonal ARIMAX model can be made to hold all of it, at the price of writing each component as a linear term. ARIMA is one tool among several and is not the default for an energy forecasting problem.

6.6.3 Recursive multi-step forecasting

A forecast is rarely wanted for one step only. At 14:00 the aggregator needs 15:00, 16:00, 17:00 and 18:00 together, and a one-step model must be extended to produce them. The recursive construction feeds the model's own output back as its input,

$$\hat{y}_{t+1} = f(y_t), \qquad \hat{y}_{t+2} = f(\hat{y}_{t+1}), \qquad \hat{y}_{t+3} = f(\hat{y}_{t+2}), \tag{6.27}$$

so that the second step is computed from a predicted value and the third from a prediction built on a prediction. The appeal is economy: one fitted model serves every horizon.

The cost is that errors propagate. Take the one-step model $y_{t+1} = 0.9\,y_t + 10$ with $y_t = 100$, so $\hat{y}_{t+1} = 0.9 \times 100 + 10 = 100$. Suppose the realized value is 105. The recursive scheme has already issued its next forecast using 100 rather than 105, and that discrepancy enters every subsequent step: with the stale value the two-step forecast is 100, whereas with the true value it would have been $0.9 \times 105 + 10 = 104.5$, a gap of 4.5 MW that was created by the model's own error rather than by any new disturbance. Two properties of equation (6.27) govern how bad this gets. The one-step error variance compounds across steps, so the multi-step intervals are wider than a naive propagation suggests; and the model is being evaluated at inputs it never saw during training, since it was fitted on observed values and is now being fed predicted ones, which is a form of distribution shift.

6.6.4 Direct multi-horizon forecasting

The alternative is to train a separate model for each horizon, each predicting from the information available at issue time,

$$\hat{y}_{t+1} = f_1(X_t), \qquad \hat{y}_{t+2} = f_2(X_t), \qquad \hat{y}_{t+3} = f_3(X_t), \tag{6.28}$$

so that no predicted value is ever fed back. Each f_h is fitted on pairs (X_t, y_{t+h}) drawn from history, which is a well-posed supervised problem for every h, and each is free to weight the features differently: f_1 may lean on y_t, while f_{24} leans on the weather forecast and the calendar.

Figure 6.6 contrasts the two constructions. The upper row is the recursive scheme: a single model box is applied three times along a chain, with each predicted value drawn as the input to the next application, and the red annotation marks where the error enters and is carried forward. The lower row is the direct scheme: the feature vector fans out to three separate model boxes, each emitting one horizon, with no arrows between the outputs. The structural difference visible in the figure is the absence of any path from one output to another in the lower row, which is precisely the absence of error propagation.

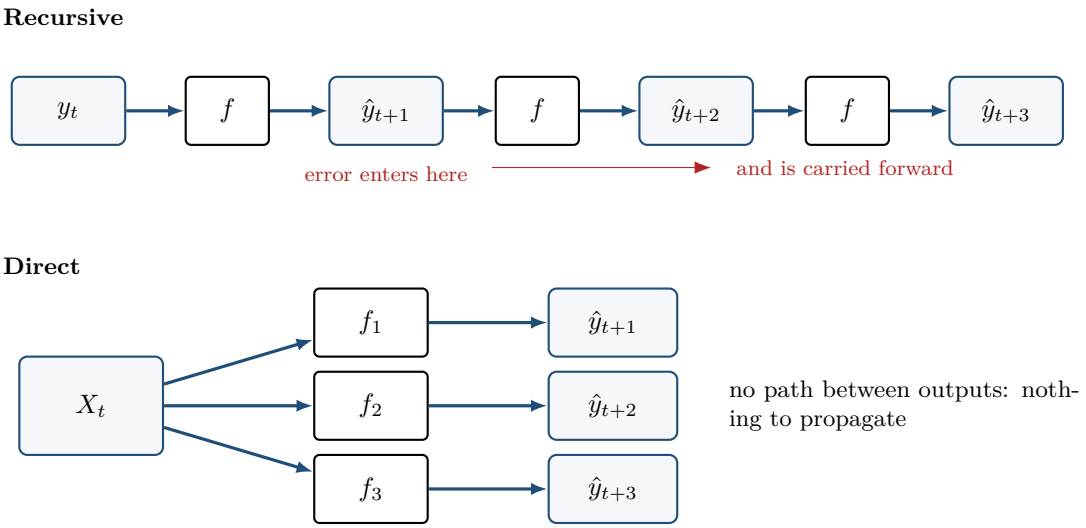


Figure 6.6: Recursive and direct multi-step forecasting, equations (6.27) and (6.28). Upper row: one model applied repeatedly to its own output. Lower row: one model per horizon, each reading only the information available at issue time.

Neither scheme dominates. The direct scheme costs one fitted model per horizon, which for 96 quarter-hourly steps is a real engineering burden, and its forecasts need not be mutually consistent, since nothing forces $\hat{y}_{t+2}$ to be reachable from $\hat{y}_{t+1}$. The recursive scheme is cheap and self-consistent and accumulates error. The common practical compromise is direct models at the horizons that carry operational decisions, together with a recursive fill for the rest.

6.6.5 The value of autoregressive information depends on the horizon

The usefulness of everything in this chapter is a function of the horizon h, for a reason that follows from equation (6.5): a geometrically decaying quantity is informative for a few multiples of its time constant and negligible afterward. Three regimes are worth naming. At $h = 15$ minutes the autoregressive information dominates, because the fleet's physical state cannot change much in a quarter of an hour, and the last observed value plus the last residual are close to sufficient. At $h = 3$ hours the weather forecast and the thermal state matter more, because the initial state has partly decayed while temperature has moved. At $h = 24$ hours the recent residuals say very little, because the state that generated them has turned over completely; what remains is weather, calendar and fleet composition.

Figure 6.7 draws the trade-off. The horizontal axis is the horizon on a logarithmic scale from 15 minutes to 24 hours; the vertical axis is the share of the forecast's useful information contributed by each source; the two curves are the autoregressive contribution and the contribution of weather, calendar and state features; the three dashed verticals mark the horizons

named above. The curves cross between one and two hours ahead, which is the horizon at which a forecaster should stop tuning the residual correction and start improving the weather inputs. The quantitative shape is illustrative, but the ordering of the three regimes is not: it follows from the decay of the autoregressive term.

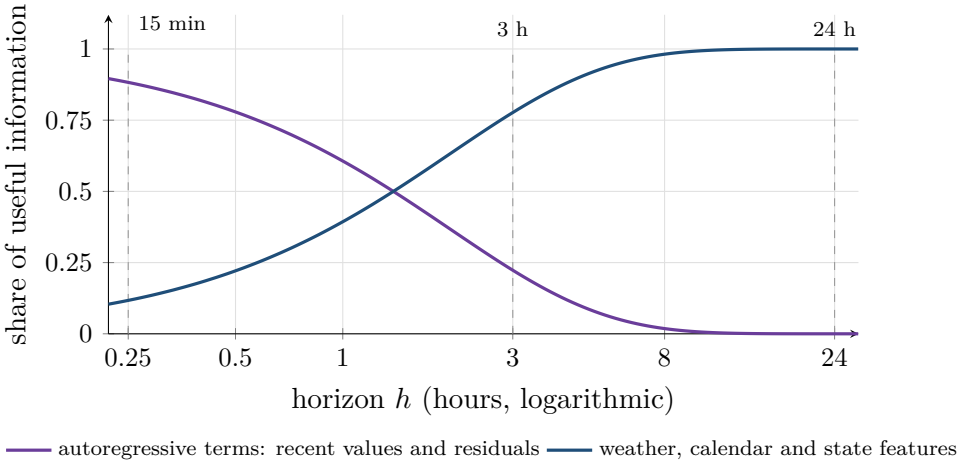


Figure 6.7: The value of autoregressive information against horizon, illustrative. The curves are $v_{\mathrm{AR}}(h) = e^{-h/2}$ and $v_{\mathrm{W}}(h) = 1 - e^{-h/2}$ with h in hours, chosen so that the autoregressive share is 0.88 at 15 minutes, 0.22 at three hours and negligible at 24 hours; the dashed verticals mark the three regimes of Section 6.6.5.

A thermostat fleet at 17:00. The abstraction becomes concrete in the following situation. At 13:45 a model predicts

$$\hat{P}_{17:00} = f(P_{14:00},\ P_{\text{yesterday},17:00},\ \hat{T}_{17:00},\ N_{\text{avail}},\ D) = 110\,\text{MW}, \tag{6.29}$$

using the fleet's recent load, yesterday's load at the target hour, the forecast temperature, the number of available devices and the duty cycle. The model has, however, underpredicted for the previous hour: at quarter-hour resolution the residuals at 13:00, 13:15, 13:30 and 13:45 are $+3$, $+4$, $+5$ and $+5$ MW. The reading is that the current fleet state is hotter or more active than the five features capture, most likely because the buildings' thermal mass is above what the outdoor-temperature history implies. For a short horizon the forecast should be adjusted upward by something like the persistent part of that bias. For the 24-hour-ahead forecast the same residual run is much less relevant, because by then the fleet will have passed through an entire overnight cooling cycle and the state that produced the bias will no longer exist.

6.6.6 Residual analysis in practice

Everything in this chapter reduces operationally to a short list of diagnostics computed on out-of-sample residuals. Table 6.2 gives the list: the quantity to compute, what a departure from the expected value indicates, and the usual remedy. The five rows are ordered from cheapest to most informative, and the last of them, the breakdown by operating regime, is the one most often skipped and most often decisive, because a model can have excellent aggregate statistics and fail systematically on heat waves or on dispatch days, which are exactly the days the forecast is used for a decision.

Table 6.2: Residual diagnostics to compute on out-of-sample forecasts.

Quantity	What a departure indicates	Usual remedy
mean residual	bias: the model is systematically high or low	recalibrate the intercept; check for a level shift in the fleet
residual variance against temperature	heteroscedasticity: error grows in extreme weather	model the variance, or forecast quantiles rather than a mean
residual autocorrelation ACF	temporal information unexplained; spikes at 24 or 168 point at calendar structure	add the missing state feature, else an AR correction
residual by horizon	the model is good at some horizons and not others	fit direct models per horizon; validate the AR correction by horizon
residual by operating regime	failure confined to normal days, heat waves, dispatch events or high duty cycle	regime features or interactions; separate models per regime

Conceptual questions

"Your GBM performs well but the residuals have strong lag-1 autocorrelation. What would you do?"
A strong answer treats the autocorrelation as a diagnostic rather than a defect to be patched. It begins by asking why the structure is there: the two common causes are a missing state variable, such as indoor temperature, duty cycle or control history, and a data-latency problem in which a feature arrives too late and is effectively stale. The first preference is to add the feature, because a feature that represents the state works at every horizon and identifies the mechanism. If the autocorrelation survives the feature work, then model the residuals autoregressively as a correction on top of the gradient-boosting machine, as in equation (6.12). The answer closes on validation: the correction has to be evaluated by horizon, because persistence helps substantially at fifteen minutes and contributes little several hours ahead.

Conceptual questions

"When would you use ARIMA instead of gradient boosting?"
A strong answer names the conditions rather than expressing a preference: when the dominant structure is low-dimensional temporal dependence, when the series is stable, and when there are few nonlinear external drivers to represent. It then applies those conditions to the case at hand. For an aggregator's thermostat fleet load, none of the three holds, because temperature enters nonlinearly and in interaction with hour of day, several exogenous variables are measured, and dispatch introduces regime changes; so ARIMA belongs in the program as a benchmark and as a residual model, not as the first choice.

6.6.7 Three representations of time dependence

Stepping back, this book handles temporal dependence in three ways, and Table 6.3 sets them beside one another: how each represents the past, what it assumes, and what it costs. The first is explicit lag features, y_{t-1}, y_{t-24}, y_{t-168}, supplied to a regression or a tree ensemble, which was the approach of Chapters 4 and 5. The second is autoregressive structure, $y_t = \sum_i \phi_i y_{t-i} + \varepsilon_t$, which is this chapter. The third is a learned latent state, $\mathbf{h}_t = f(X_t, \mathbf{h}_{t-1})$, which is Chapter 7.

The three form a progression from explicit lags, through statistical temporal dynamics,

Table 6.3: Three representations of temporal dependence.

Representation	How the past enters	What it assumes	What it costs
explicit lag features	chosen lags as columns of X_t	the modeler knows which lags matter	nothing beyond feature work; silent about lags not chosen
autoregressive structure	coefficients ϕ_i on past values, θ_j on past shocks	stationarity and linear dependence	differencing decisions; no room for many exogenous drivers
learned latent state	a state vector updated each step, $\mathbf{h}_t = f(X_t, \mathbf{h}_{t-1})$	enough data to identify the state	data, computation and interpretability

to a learned state, and the progression is one of increasing generality bought with decreasing transparency. It is also the order in which a forecaster should try them, because the simplest representation that removes the residual autocorrelation is the one to keep.

6.7 What to retain

Definition 6.1 (Residual dynamics)**.** An autoregressive model asks whether the past of a process contains information about its future; differencing handles a level that moves; and a moving-average term represents persistence in unexpected shocks. The practical consequence is a procedure rather than a model family: after fitting any forecast model, inspect its out-of-sample residuals, and if $\text{Corr}(e_t, e_{t-1}) \neq 0$ the model is leaving temporal information unexplained. The remedy is sometimes an autoregressive correction, sometimes a better feature, and sometimes a hidden state the model does not represent at all.

Table 6.4 collects the chapter's objects with the subsection that defines each.

Table 6.4: The objects of this chapter.

Object	Definition	Section
e_t	residual, $y_t - \hat{y}_t$; the chapter's unit of analysis	6.1.1
$\text{Corr}(e_t, e_{t-k})$	residual autocorrelation; target value zero	6.1.2
AR(1)	$y_t = c + \phi y_{t-1} + \varepsilon_t$; mean $c/(1-\phi)$	6.2.1
ϕ	persistence; a deviation decays as ϕ^k	6.2.2
AR(p)	p lags of the target, possibly non-consecutive	6.2.3
ARX	autoregression plus $\boldsymbol{\beta}^\mathsf{T} X_t$: persistence plus drivers	6.2.6
$\hat{e}_t$	forecast residual added to a model's prediction	6.3.1
stationarity	constant mean, lag-dependent covariance	6.4.1
Δy_t, $\Delta_s y_t$	first and seasonal differences	6.4.2, 6.4.3
ARIMA(p, d, q)	lags, differences, past shocks	6.4.4
MA(q)	$y_t = \mu + \varepsilon_t + \sum_j \theta_j \varepsilon_{t-j}$	6.4.5
$(P, D, Q)_s$	seasonal orders at period s; $s = 24$, 168	6.4.7
ACF(k)	total association at lag k	6.5.1
PACF(k)	direct association at lag k, shorter lags held fixed	6.5.3
recursive, direct	feeding predictions back versus one model per horizon	6.6.3, 6.6.4

6.8 Exercises

Exercise 6.1. For $y_t = 20 + 0.8\,y_{t-1} + \varepsilon_t$, compute the unconditional mean and the one-step forecast from $y_{t-1} = 120$. Then iterate the forecast forward five steps with no further innovations and verify that the deviation from the mean follows equation (6.5). After how many steps is the deviation below 5 MW? Repeat with $\phi = 0.5$ and comment on the difference in half-life.

Exercise 6.2. A fleet's residuals satisfy $e_t = 0.6\,e_{t-1} + \varepsilon_t$ and the most recent residual is $+8$ MW, as in equation (6.13). Compute the residual correction for the next four steps. At which step does the correction fall below 1 MW? Using equation (6.3), state the fraction of residual variance the correction removes at one step ahead, and explain why that fraction is not achieved at four steps ahead.

Exercise 6.3. Take the series $80, 84, 87, 89, 94, 98, 101, 103$ MW. Compute its first difference and its seasonal difference at period $s = 4$. Compute the sample variance of the level, of the first difference and of the seasonal difference. Which transformation would you carry into an autoregressive fit, and what does the comparison of the three variances tell you about which structure dominates this short series?

Exercise 6.4. The one-step model is $y_{t+1} = 0.9\,y_t + 10$ and $y_t = 100$, so $\hat{y}_{t+1} = 100$; the realized value is 105. Compute the two-step and three-step recursive forecasts of equation (6.27) both from the predicted value and from the realized value, and state the gap at each step. Then describe the direct models f_1, f_2, f_3 of equation (6.28) that would avoid the gap, and say what additional cost they carry.

Chapter 7

Recurrent networks: learned hidden state, LSTM and GRU

7.1 From explicit lags to a learned state

Every model so far has been handed a feature vector. The regression and tree models of Chapter 5 received the vector X_t that Chapter 4 built, and the autoregressive models of Chapter 6 received a fixed list of lags of the series itself. In both cases a person decided which parts of the past the model would see: y_{t-1}, y_{t-24}, y_{t-168}, a rolling mean over six hours, a degree variable. The question of this chapter is

What if the useful information in the past cannot be summarized well by any fixed set of lags, and the model should learn the summary instead?

For a fleet of thermostats that possibility is plausible rather than academic. The quantity that determines how much load a building can give up at 17:00 is its thermal condition, and that condition is the result of several hours of outdoor temperature, of how the indoor temperature has evolved, of how heavily the compressor has been cycling, and of whether the thermostat was dispatched recently. None of those is a single lagged number. This chapter develops the model class that constructs its own summary of history: the recurrent network, and the two gated variants, the long short-term memory cell and the gated recurrent unit, that make it usable over long sequences.

Jargon: recurrent network, hidden state, gate, element-wise product, backpropagation through time, lookback window

A *recurrent network* processes a sequence one step at a time, carrying a vector from each step to the next. That vector is the *hidden state* $\mathbf{h}_t \in \mathbb{R}^d$: a fixed-dimension summary of everything the network has read so far, written in bold because the plain symbol h is the forecast horizon in this book. A *gate* is a vector of numbers in $(0, 1)$, produced by the network itself, that multiplies another vector component by component and so decides how much of it passes; the component-by-component multiplication is the *element-wise product*, written $\odot$. *Backpropagation through time* is ordinary gradient backpropagation applied to the sequence of recurrent steps after they have been written out as a chain. The *lookback window* L is the number of past steps fed to the network at each training example; the network sees nothing earlier.

Key idea

A feature-engineered model requires a person to choose how history is summarized. A recurrent network learns the summary: a state of fixed dimension, updated by the same transformation at every time step, whose contents are whatever helps the forecast.

This opening section sets up the problem physically, states the recurrent update, and gives the example that shows why a summary of the trajectory can carry information that the current measurement does not.

7.1.1 A building as a partially observed state-space system

The physical starting point is the state-space description already used in the notation chapter. A building evolves as

$$x_{t+1} = g(x_t,\ u_t,\ w_t), \qquad P_t = h_{\mathrm{m}}(x_t,\ u_t), \tag{7.1}$$

with x_t the internal state, u_t the control applied to the thermostat, w_t the weather and other disturbances, and h_{m} the map from state to metered power. If x_t were known, forecasting would largely reduce to integrating equation (7.1) forward under forecast weather and a planned control, which is what Chapter 11 eventually does at fleet scale.

The difficulty is that x_t is not observed. A reasonable state for a single house is

$$x_t = [\,T_{\mathrm{in}},\ T_{\mathrm{wall}},\ T_{\mathrm{mass}},\ s_t,\ \mathrm{occ}_t\,]^{\mathsf{T}}, \tag{7.2}$$

the indoor air temperature, the wall temperature, the temperature of the furniture and structure that store heat, the compressor mode s_t, and occupancy. Telemetry typically delivers three numbers: the indoor temperature T_{in}, the metered power P_t, and the outdoor temperature T_t from a nearby weather station. The wall and mass temperatures are not measured, occupancy is not measured, and the compressor mode is only visible indirectly through the power.

The consequence is that the state must be inferred from the history of what *is* measured. A sequence of indoor temperatures and powers over the last several hours contains information about the unmeasured mass temperature, because a building whose air temperature has been held down by a hard-running compressor has a different stored heat content from one whose compressor has barely run. Inferring a hidden state from a record of observations is exactly what a recurrent network attempts, with one important difference developed in Section 7.4.2: the network is never told what the state means.

7.1.2 The recurrent update

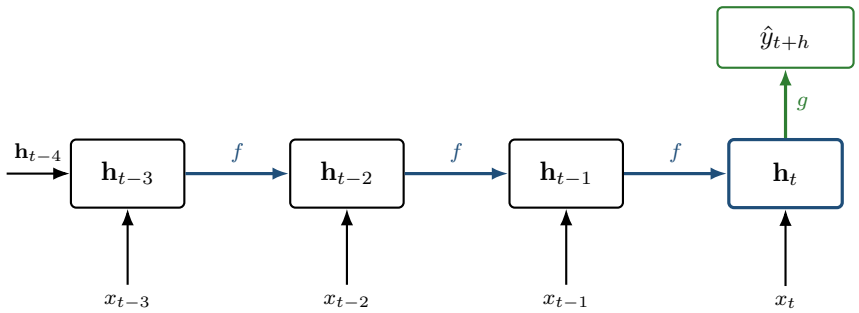
A recurrent network is defined by two maps. A transition builds the new state from the previous state and the current inputs, and a readout turns the state into a forecast,

$$\mathbf{h}_t = f(x_t,\ \mathbf{h}_{t-1}), \qquad \hat{y}_{t+1} = g(\mathbf{h}_t). \tag{7.3}$$

Here x_t is the vector of inputs observed at step t and $\mathbf{h}_t$ is the hidden state. The pair has the same shape as equation (7.1), with one map propagating a state and another reading an output off it, and that resemblance is the reason the class is attractive for physical systems.

Processing a sequence means applying f repeatedly. Starting from an initial state, the first input produces $\mathbf{h}_1$, which together with the second input produces $\mathbf{h}_2$, and so on, so that the state at the end of the sequence has been influenced by every input in it. Figure 7.1 draws this. The boxes along the middle row are the hidden states at successive steps, the arrows between them are applications of the transition f, the arrows entering from below are the inputs at each step, and the arrow leaving the last box is the readout g. The feature to notice is that the arrows between the boxes all carry the same label: there is one transition, with one set

of weights, reused at every step. A network that processes a 24-step sequence has no more parameters than one that processes a 6-step sequence, because the parameters belong to f and not to the positions.



Every arrow labeled f is the *same* transition with the same weights W_x, W_h, b_h; the number of parameters does not grow with the length of the sequence.

Inputs x_s enter one step at a time, in order.

Figure 7.1: A recurrent network written out over four time steps. The middle row holds the hidden states, the horizontal arrows are applications of the transition f of equation (7.3), the vertical arrows from below are the per-step inputs, and the arrow at the upper right is the readout g producing the forecast. The same weights are used at every step.

7.1.3 Two buildings with identical current weather

The reason a state is worth carrying is that the present measurement can be the same in physically different situations. Consider two buildings at the same moment, both reporting an outdoor temperature of 35°C. Over the previous four hours building A saw 28, 30, 32, 34, 35°C, a steady warming; building B saw 39, 38, 37, 36, 35°C, a cooling afternoon. The current weather is identical and the thermal condition is not: building A has been absorbing heat for hours from a cooler starting point, while building B has been shedding heat after a hot morning, so its structure is much warmer.

Figure 7.2 shows both trajectories. The horizontal axis is time in hours before the forecast is issued and the vertical axis is temperature. The two solid curves are the outdoor temperatures of the two buildings, which meet at 35°C at the right-hand edge; the two dashed curves are the temperatures of the buildings' thermal mass, obtained by driving a first-order lag with each outdoor trajectory. The solid curves converge and the dashed curves do not: at the issue time the two buildings' stored heat differs by 6.6°C even though their measured outdoor temperature is the same. A model whose input is the current temperature alone cannot tell these two buildings apart, and will make the same forecast for both. A model whose input is the sequence can distinguish them, because the state it carries depends on the whole trajectory and the trajectories differ.

> **Principle 7.1.** *Two systems with identical current measurements can be in different states. Whenever that is true, a model that reads only the current measurement is underspecified, and the remedy is to carry a summary of the trajectory.*

7.2 The recurrent network

This section writes out the simplest recurrent cell, says what its hidden state holds, contrasts it with an explicit lag list, and explains why simply lengthening the lag list is not an equivalent solution.

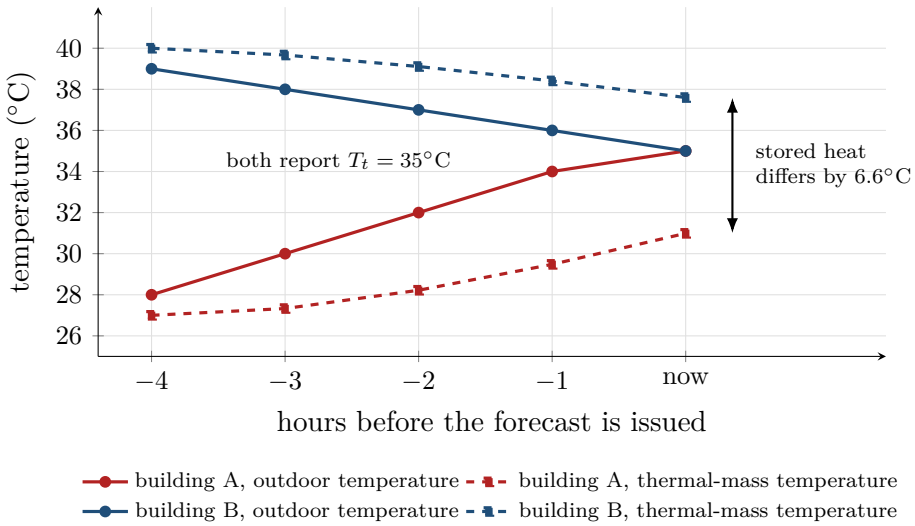


Figure 7.2: Two buildings whose current outdoor temperature is identical and whose thermal condition is not. Solid curves are the outdoor temperature sequences of Section 7.1.3. Dashed curves are the thermal-mass temperature from the first-order recursion $T_{\mathrm{mass}}(k+1) = T_{\mathrm{mass}}(k) + [\,T(k) - T_{\mathrm{mass}}(k)\,]/\tau$ with $\tau = 3$ hours, started at 27°C for building A and 40°C for building B. The initial values and the time constant are illustrative; the divergence at the right-hand edge is the point.

7.2.1 The recurrent cell equations

The basic recurrent cell instantiates f and g of equation (7.3) as an affine map followed by a nonlinearity,

$$\mathbf{h}_t = \tanh(W_x\, x_t + W_h\, \mathbf{h}_{t-1} + b_h), \tag{7.4}$$

$$\hat{y}_t = W_y\, \mathbf{h}_t + b_y. \tag{7.5}$$

Each symbol has a role. The vector x_t holds the inputs observed at step t; for a thermostat fleet a natural choice is $x_t = [\,P_t,\ T_t,\ T_{\mathrm{in}},\ u_t\,]^{\mathsf{T}}$, the metered power, the outdoor temperature, the indoor temperature and the control signal. The vector $\mathbf{h}_{t-1}$ is the memory inherited from the previous step. The matrix $W_x \in \mathbb{R}^{d\times m}$ says how the current inputs write into that memory, the matrix $W_h \in \mathbb{R}^{d\times d}$ says how the previous memory propagates into the new one, and b_h is a bias. The hyperbolic tangent squashes each component into $(-1, 1)$, which both supplies the nonlinearity that makes the model more than a linear filter and keeps the state bounded. The readout equation (7.5) is an ordinary linear layer; with a multi-output head it produces a whole forecast vector, as Section 7.5.1 describes.

The parameter count is modest. With input dimension m and state dimension d the cell has $d(m + d + 1)$ parameters, so $m = 4$ inputs and $d = 32$ state components give $32 \times 37 = 1184$ weights, independent of how long the sequences are.

The equation is not the content. What matters about equation (7.4) is not its algebraic form but what $\mathbf{h}_t$ ends up holding. It is a compressed, learned representation of the history the network has read. For a thermostat fleet it may amount to something like a thermal condition, a behavioral condition and a recent operating condition, mixed together in d numbers. Nothing in the training procedure specifies those quantities; they appear, in whatever entangled form, because representing them reduces the forecast error.

7.2.2 Chosen summaries and learned summaries

The contrast with the previous two chapters is sharp. A gradient-boosted model is handed a vector such as

$$X_t = [\, P_{t-1},\ P_{t-2},\ P_{t-24},\ T_t,\ T_{t-1},\ T_{t-2}\,]^{\mathsf{T}}, \tag{7.6}$$

in which a person has already decided that the first, second and twenty-fourth lags of power and the first three temperatures are the relevant history, and that nothing else about the past matters. The model then learns a function of that vector. A recurrent network is handed the sequence

$$x_{t-L+1},\ x_{t-L+2},\ \ldots,\ x_t \tag{7.7}$$

and constructs $\mathbf{h}_t$ from it. The division of labor has moved: in the feature-engineered case a person chooses how history is summarized and the model learns the mapping from the summary to the target; in the recurrent case the model learns both.

Neither side of that trade is automatically better. A chosen summary encodes domain knowledge directly and cheaply, and the lag list of equation (7.6) is a good summary precisely because the autocorrelation and partial autocorrelation functions of Chapter 2 show which lags carry information. A learned summary is more flexible and is paid for with more parameters, more data and less interpretability. The remainder of the chapter is largely about when the flexibility is worth the price.

7.2.3 A thermostat sequence

Make the sequence concrete. Take the last six hourly records for a thermostat cohort, each a four-component vector of outdoor temperature, indoor temperature, metered power and control signal,

$$x_{t-5},\ x_{t-4},\ \ldots,\ x_t, \qquad x_s = [\, T_s,\ T_{\text{in}}(s),\ P_s,\ u_s\,]^{\mathsf{T}}, \tag{7.8}$$

and run them through equation (7.4) to obtain $\mathbf{h}_{t-5}, \ldots, \mathbf{h}_t$. The final state is the only thing the readout sees.

What might that final state contain, if the network has been trained to predict the cohort's power or its flexibility one hour ahead? Plausible contents include: the building stock has been warming over the window; the compressors have run heavily and the fleet is near the top of its duty cycle; the thermostats have recently come out of a dispatch and are recovering; the current metered load is therefore temporarily suppressed; and the indoor temperatures sit close to the comfort limit, so little further curtailment is available. Each of those is a statement about the window as a whole rather than about any one record in it. None of them is specified anywhere in the model. They emerge in the state only to the extent that they help predict the target, which is both the appeal of the approach and the reason the resulting state resists interpretation.

7.2.4 The cost of a very long lag list

A fair objection is that history can be handed to a tree ensemble simply by listing more lags. Feeding $P_t, \ldots, P_{t-100}$ and $T_t, \ldots, T_{t-100}$ into a gradient-boosted model is legitimate, and it sometimes works well. Four problems limit it.

The first is dimension. Two hundred lag features for two signals, before any calendar or state variables are added, is a large and highly collinear design matrix; neighboring lags are nearly duplicates of each other, so the ensemble spends its splits distinguishing variables that carry almost the same information. The second is that order is not represented. A tree sees two hundred unordered columns; nothing in the model says that P_{t-7} comes between P_{t-8} and P_{t-6}, so a pattern defined by the *shape* of a trajectory, such as a steady rise followed by a plateau, has to be rediscovered as a conjunction of threshold tests on individual columns. The third is

that the useful memory length is not fixed: on a mild day the last two hours determine almost everything, while during a heat wave with an earlier curtailment the relevant window is much longer. A fixed lag list must be long enough for the worst case and then carries dead weight in every other case. The fourth is interaction. Effects such as "the response depends on how long ago the last dispatch was *and* on how hot it has been since" are interactions across time, which a tree can represent only by deep conjunctions and many splits.

A recurrent cell addresses all four at once by replacing the list with a state of fixed dimension. Whether the window is 10 steps or 100 steps, $\mathbf{h}_t \in \mathbb{R}^d$ with the same d and the same parameters; the order is respected because the steps are processed in order; and the length of the effective memory is itself something the weights can control, a capability made explicit by the gates of Section 7.3.

7.3 Vanishing gradients and the long short-term memory cell

The cell of equation (7.4) carries information forward in principle and fails to carry it far in practice. This section derives why, then builds the long short-term memory cell that repairs it, gate by gate.

7.3.1 Backpropagation through time and vanishing gradients

Training a recurrent network means computing the derivative of a loss at the end of a sequence with respect to weights that acted at every step. Suppose the loss depends on the final state, $\mathcal{L} = \mathcal{L}(\mathbf{h}_T)$. The sensitivity of the loss to a state k steps earlier is a product of per-step Jacobians,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}_{t-k}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}_t} \cdot \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} \cdot \frac{\partial \mathbf{h}_{t-1}}{\partial \mathbf{h}_{t-2}} \cdots \frac{\partial \mathbf{h}_{t-k+1}}{\partial \mathbf{h}_{t-k}}, \tag{7.9}$$

which is ordinary backpropagation applied to the chain of Figure 7.1 and is called backpropagation through time.

For the cell of equation (7.4) each factor is

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} = \operatorname{diag}\big(1 - \tanh^2(a_t)\big)\, W_h, \qquad a_t = W_x x_t + W_h \mathbf{h}_{t-1} + b_h, \tag{7.10}$$

a fixed matrix W_h premultiplied by a diagonal whose entries lie in $(0, 1]$, since $1 - \tanh^2$ is at most one and is much smaller than one wherever the state is saturated. The product of k such factors therefore behaves like γ^k for some effective per-step gain γ, and $\gamma < 1$ is the normal case: the squashing derivative pulls it down, and W_h must be kept small for training to be stable in the first place.

The arithmetic is unforgiving. With $\gamma = 0.5$, ten steps give $0.5^{10} \approx 0.001$ and fifty steps give $0.5^{50} \approx 9 \times 10^{-16}$, which is numerically zero. Almost no gradient signal reaches the weights to tell them that something fifty steps ago mattered, so the fitted cell emphasizes recent history whether or not the distant history is relevant. This is the vanishing gradient problem. The same product can also diverge: if $\gamma > 1$ the gradient explodes, which is the easier failure to notice and the easier one to fix, by clipping the gradient norm. Figure 7.3 plots the decay for several values of γ on a logarithmic vertical axis, including the $\gamma = 1$ case that the gated cell of Section 7.3.6 is designed to approximate.

The memory horizon that thermal storage requires. The lost range is exactly the range a thermostat problem needs. How much flexibility a building can offer at 17:00 depends on how much heat it took on through the afternoon, and a curtailment at 10:00 can still influence the 17:00 response through the thermal storage of the structure, because the building spent the

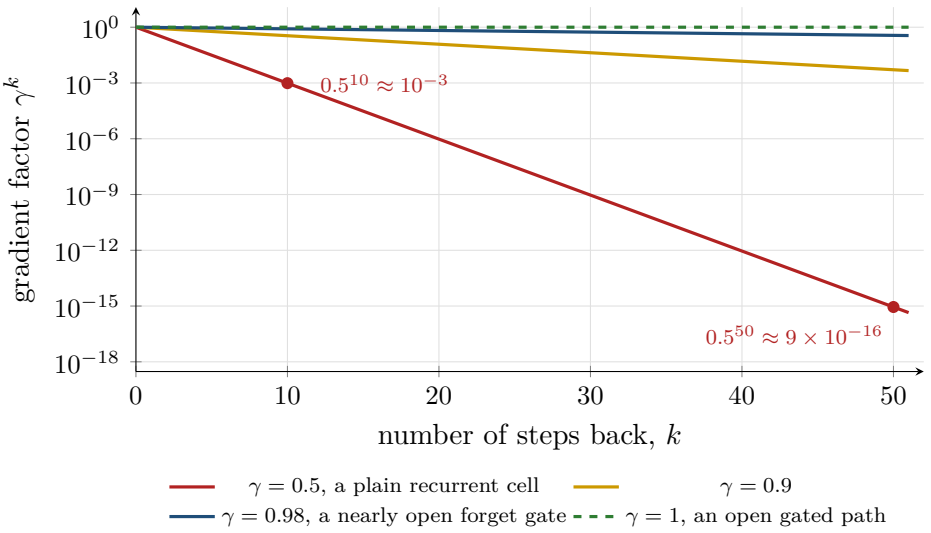


Figure 7.3: Decay of the gradient factor in equation (7.9). Curves are γ^k for $\gamma \in \{0.5, 0.9, 0.98, 1\}$ on a logarithmic vertical axis. The two marked values on the red curve are the ones quoted in Section 7.3.1; the corresponding value on the blue curve is $0.98^{50} \approx 0.36$. A plain recurrent cell operates near the red curve; the gated cell-state path of Section 7.3.6 operates near the dashed line.

intervening hours recovering from it. On an hourly series that is a seven-step dependency, and on a fifteen-minute series it is twenty-eight steps. With $\gamma = 0.5$ the gradient at twenty-eight steps is 4×10^{-9}: the training signal that would connect the morning event to the evening forecast does not arrive. The long short-term memory cell was designed for precisely this situation.

7.3.2 A dedicated memory state

The long short-term memory cell, abbreviated LSTM, splits the single hidden state into two. The state $\mathbf{h}_t$ remains the exposed representation, the thing the readout and the next step's gates see. A second state $\mathbf{c}_t$, the cell state, is an internal store that is longer-lived and is not squashed on every step. Its update is the one equation worth memorizing,

$$\mathbf{c}_t = f_t \odot \mathbf{c}_{t-1} + i_t \odot \tilde{\mathbf{c}}_t, \tag{7.11}$$

in which f_t is the forget gate, i_t the input gate and $\tilde{\mathbf{c}}_t$ a candidate memory, all defined below.

The symbol $\odot$ is the element-wise product: corresponding components are multiplied, so

$$[2,\ 3] \odot [4,\ 5] = [8,\ 15]. \tag{7.12}$$

The significance for equation (7.11) is that each component of the memory receives its own gate value. The gates are vectors, not scalars, so one component of $\mathbf{c}_t$ can be held almost unchanged while another is overwritten in the same step.

Read in words, equation (7.11) says

$$\text{new memory} \ = \ \text{retained old memory} \ + \ \text{accepted new information}, \tag{7.13}$$

and that sentence is the whole design. Everything else in the cell is machinery for producing the two gate vectors and the candidate.

7.3.3 The forget gate

The retention factor is produced by the network itself from the current inputs and the previous exposed state,

$$f_t = \sigma\big(W_f x_t + U_f \mathbf{h}_{t-1} + b_f\big), \qquad \sigma(z) = \frac{1}{1+e^{-z}}, \tag{7.14}$$

where the logistic function σ maps every component into $(0, 1)$. That range is what makes f_t a retention factor rather than a general weight: it can attenuate a memory component but not amplify or invert it. A value such as

$$f_t = [\,0.95,\ 0.1,\ 0.8\,] \tag{7.15}$$

means that the first component of the memory keeps 95 % of its value this step, the second keeps 10 %, and the third keeps 80 %. The question the gate answers is which parts of the old memory should still matter now, and because f_t depends on x_t the answer can change from step to step.

Holding a gate value fixed turns it into a time constant. Repeated application of a constant retention f decays a memory component as f^k, so its effective memory time constant in steps is

$$\tau_{\text{eff}} = -\frac{1}{\ln f}. \tag{7.16}$$

On an hourly series the three components of equation (7.15) have time constants of 19.5, 0.43 and 4.5 hours. The interpretation is physical. A component that represents the thermal loading of the building stock should decay slowly, because thermal mass changes over many hours, and the gate learns $f_t \approx 1$ for it. A component that represents the compressor state a few minutes ago is useless an hour later, and the gate learns a small f_t for it. One cell can therefore maintain several memories operating on different timescales at once, which is what the heterogeneous vector in equation (7.15) expresses.

7.3.4 Candidate memory and the input gate

The information offered for writing is the candidate memory

$$\tilde{\mathbf{c}}_t = \tanh\big(W_c x_t + U_c \mathbf{h}_{t-1} + b_c\big), \tag{7.17}$$

a tanh rather than a logistic because it is a signed quantity to be added, not a fraction. Its content is whatever about the current step is worth recording: that the outdoor temperature rose sharply, that the compressor switched on, that a control event began, that the indoor temperature went above setpoint.

How much of that candidate is actually written is decided by a second gate of the same form as the forget gate,

$$i_t = \sigma\big(W_i x_t + U_i \mathbf{h}_{t-1} + b_i\big), \tag{7.18}$$

so the quantity added to the memory is $i_t \odot \tilde{\mathbf{c}}_t$. The separation matters: the candidate says what could be written and the input gate says whether now is the moment to write it. A step that carries no new information can be allowed to pass with $i_t \approx 0$, leaving the memory untouched.

Example 7.1 (The memory update in numbers)**.** Take one component of the cell state and follow it through one step. Suppose the stored value is $\mathbf{c}_{t-1} = 10$ and the forget gate returns $f_t = 0.9$, so the retained part is $0.9 \times 10 = 9$. Suppose the candidate is $\tilde{\mathbf{c}}_t = 4$ and the input gate returns $i_t = 0.5$, so the accepted part is $0.5 \times 4 = 2$. Then equation (7.11) gives

$$\mathbf{c}_t = 9 + 2 = 11.$$

The arithmetic is deliberately trivial, and every component of a real cell state does exactly this at every step, with its own four numbers.

7.3.5 The output gate

What the rest of the network sees is not the memory itself but a gated, squashed view of it,

$$o_t = \sigma(W_o x_t + U_o \mathbf{h}_{t-1} + b_o), \qquad \mathbf{h}_t = o_t \odot \tanh(\mathbf{c}_t). \tag{7.19}$$

The output gate answers which parts of the memory are relevant to the present prediction. The consequence is the one that makes the two-state design worth its complexity: the cell state can hold information that is not currently being used. A component recording that a curtailment occurred three hours ago can sit in $\mathbf{c}_t$ with $o_t \approx 0$, invisible to the readout, and then be exposed later when the forecast needs it. A single-state cell has no way to retain something it is not currently using, because everything it retains passes through the readout.

Figure 7.4 assembles the four equations. The horizontal path across the top is the cell state, entering as $\mathbf{c}_{t-1}$ on the left and leaving as $\mathbf{c}_t$ on the right; the circle marked $\times$ on that path applies the forget gate and the circle marked $+$ adds the gated candidate. The four boxes along the bottom are the gates and the candidate, all of them functions of the same pair $[\,x_t,\ \mathbf{h}_{t-1}\,]$ carried on the gray input line. The lower right branch is equation (7.19): the cell state is passed through a tanh, multiplied by the output gate, and emitted as $\mathbf{h}_t$. The feature of the diagram that carries the argument of Section 7.3.6 is the top path. From $\mathbf{c}_{t-1}$ to $\mathbf{c}_t$ there is one multiplication and one addition, and no matrix multiplication and no squashing function at all.

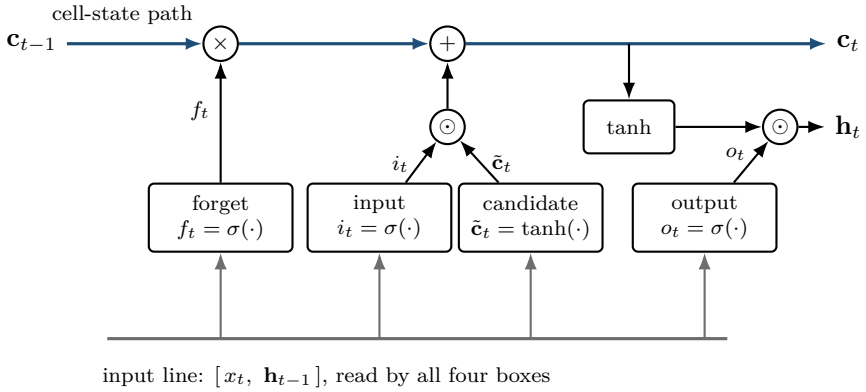


Figure 7.4: The long short-term memory cell of equations (7.11) to (7.19). The top path carries the cell state from $\mathbf{c}_{t-1}$ to $\mathbf{c}_t$ through one element-wise multiplication by the forget gate and one addition of the gated candidate. The four boxes are the gates and the candidate, each a function of the current inputs and the previous hidden state. The hidden state leaves at the right as $\mathbf{h}_t = o_t \odot \tanh(\mathbf{c}_t)$.

7.3.6 The additive path and long memory

The reason the gated cell helps with long dependencies is visible in equation (7.11). Differentiating the cell state with respect to its predecessor, and holding the gates fixed, gives

$$\frac{\partial \mathbf{c}_t}{\partial \mathbf{c}_{t-1}} = \operatorname{diag}(f_t), \tag{7.20}$$

so the product over k steps that appears in equation (7.9) becomes $\prod_s f_s$ component by component rather than a product of matrices. Two things follow. When a forget gate stays near one, the cell state satisfies $\mathbf{c}_t \approx \mathbf{c}_{t-1} + i_t \odot \tilde{\mathbf{c}}_t$: information passes through many steps as a running sum, without being repeatedly multiplied by a weight matrix and pushed through a saturating nonlinearity. And the gradient that flows backward along that path is multiplied by numbers near one instead of by $\gamma < 1$. With $f = 0.98$ the factor after fifty steps is $0.98^{50} \approx 0.36$,

against 9×10^{-16} for $\gamma = 0.5$; these are the blue and red curves of Figure 7.3. This is the fundamental reason gated cells train on long sequences where plain recurrent cells do not.

A physical analogy makes the structure familiar. Stored thermal energy in a building evolves as

$$E_{t+1} = \alpha\, E_t + Q_t, \tag{7.21}$$

with $\alpha = e^{-\Delta t/RC}$ the fraction retained over one step and Q_t the heat added during it; for $RC = 20$ hours and hourly steps $\alpha = 0.951$. Equation (7.11) has the same form, with f_t in the role of α and $i_t \odot \tilde{\mathbf{c}}_t$ in the role of Q_t, except that the retention factor is produced by the network at each step instead of being fixed by a physical time constant. The claim is not that a trained cell discovers thermal energy. It is that the two systems share a structure, a state that persists with a multiplicative decay and is updated additively by new input, and that this structure is the right one for a quantity with memory.

> **Principle 7.2.** *A gated cell state replaces a product of weight matrices with a product of retention factors. Information and gradients survive long sequences because the path between distant steps is additive rather than repeatedly transformed.*

7.3.7 A thermostat forecasting example

Put the cell to work on the running problem. The input at each hourly step is $x_t = [\,T_t,\ T_{\text{in}},\ P_t,\ u_t\,]^{\mathsf{T}}$ and the sequence covers an afternoon:

- 12:00 — hot outside, HVAC running at a moderate duty cycle.
- 13:00 — hotter, HVAC running heavily.
- 14:00 — hotter still, HVAC still running heavily.
- 15:00 — hot, and a dispatch begins: u_t becomes nonzero.
- 16:00 — hot, load curtailed, metered power well below normal.
- 17:00 — the quantity to forecast is the flexibility $F_{17:00}$.

A trained cell may carry four kinds of information through that window: an accumulated thermal loading, built up by the additive path over 12:00 to 14:00; a record that the compressors were near saturation before the event; the fact of the recent curtailment; and, as a consequence of the first three, a rebound potential, since buildings that were curtailed while heavily loaded will draw hard when released. The state $\mathbf{h}_{16:00}$ is then handed to the readout, which produces $\hat{F}_{17:00}$. The point of the example is that none of the four quantities is an input. They are combinations of the window that the weights found useful, and the curtailment at 15:00 reaches the 17:00 forecast through the cell state rather than through an engineered "hours since last dispatch" feature.

7.4 The gated recurrent unit and a comparison of recurrent cells

Before comparing cells, this section states the argument that makes sequence models attractive for controllable resources specifically, and the price paid for a learned state. It then introduces the lighter gated cell and sets the three side by side.

7.4.1 Control history as the strongest argument

The case for a sequence model is strongest when the history that matters is a history of the operator's own actions.

Example 7.2 (Two fleets that look identical)**.** Two thermostat fleets report the same three numbers at the same moment: the same metered power P_t, the same outdoor temperature T_t, and the same average indoor temperature T_{in}. Fleet A has not been dispatched for several days. Fleet B came out of a two-hour curtailment thirty minutes ago, so its buildings are warm, its compressors are recovering, and it is primed for a rebound. Asked for flexibility over the next hour, the correct answers differ substantially: fleet A can be curtailed normally, while fleet B has already spent much of its thermal slack and will respond less and for a shorter time.

A model whose input is a snapshot cannot separate those two fleets unless somebody has engineered a control-history feature, such as hours since the last event, the duration of the last event, or the cumulative curtailed energy over the last day. Those features can be built, and Chapter 4 built features of exactly that kind. The difference is that a recurrent network can encode $u_{t-1}, u_{t-2}, \dots$ into its state without being told which summary of the control history matters, and the relevant summary is genuinely hard to guess: it depends on event duration, on weather during and after the event, and on how saturated the fleet was when the event began. In forecasting for controllable distributed resources, this is the most compelling argument for sequence models.

7.4.2 A learned state and a physical state

Set the two state equations next to each other. The physical model is

$$x_{t+1} = g(x_t,\ u_t,\ w_t), \tag{7.22}$$

in which the components of x_t were chosen by a person: this one is an indoor temperature in degrees, that one a state of charge in kilowatt-hours. The recurrent model is

$$\mathbf{h}_{t+1} = f_\theta(\mathbf{h}_t,\ x_t), \tag{7.23}$$

in which the components of $\mathbf{h}_t$ were chosen by the optimizer. In the first case the state semantics are specified and the dynamics are fitted or derived; in the second the dynamics are fitted and the semantics are whatever happens to be convenient.

The price is interpretability. A trained hidden state is a vector such as

$$\mathbf{h}_t = [\,0.21,\ -0.73,\ 0.44,\ \dots]^{\mathsf{T}}, \tag{7.24}$$

and the honest answer to what component 17 represents is that nobody knows. It is almost certainly a mixture: some thermal condition, some weather history, some schedule information, some behavioral regularity, some device-response characteristic, in proportions that the training data happened to favor. The vector is useful for prediction and is not a physical description. That distinction is the one the conceptual box of Section 7.6.6 asks about, and it has operational consequences taken up in Section 7.6.4.

7.4.3 The gated recurrent unit

The gated recurrent unit, abbreviated GRU, keeps the idea of gating and discards the separate cell state. It has two gates instead of three and a single state. Its update is a convex combination of the old state and a candidate,

$$\mathbf{h}_t = (1 - z_t) \odot \mathbf{h}_{t-1} + z_t \odot \tilde{\mathbf{h}}_t, \tag{7.25}$$

which reads as new state equals retained old state plus accepted candidate state, the same sentence as equation (7.13) with the two coefficients tied so that they sum to one.

The update gate z_t plays both roles that f_t and i_t play in the LSTM. When $z_t \approx 0$ the state is carried forward essentially unchanged, $\mathbf{h}_t \approx \mathbf{h}_{t-1}$, and memory is preserved; when $z_t \approx 1$ the state is replaced by the candidate and the past is largely discarded. Because retention and acceptance are forced to be complements, a GRU cannot write new information without displacing some old information, which is the main expressive difference from the LSTM.

The second gate controls how much of the past is used when the candidate itself is formed,

$$z_t = \sigma\big(W_z x_t + U_z \mathbf{h}_{t-1} + b_z\big), \qquad r_t = \sigma\big(W_r x_t + U_r \mathbf{h}_{t-1} + b_r\big), \tag{7.26}$$

$$\tilde{\mathbf{h}}_t = \tanh\big(W_h x_t + U_h (r_t \odot \mathbf{h}_{t-1}) + b_h\big). \tag{7.27}$$

With $r_t \approx 0$ the candidate is built from the current inputs alone, which is the right behavior at a regime change: the past has stopped being relevant, as when a dispatch event begins and the preceding steady operation no longer describes the fleet. With $r_t \approx 1$ the candidate uses the full previous context. Figure 7.5 shows the arrangement: the previous state enters at the upper left and splits, one branch running straight across the top and the other passing through the reset gate into the candidate box; the two branches meet in the blue combination box, where the update gate weights them by $1 - z_t$ and z_t; and the result leaves at the right as the new state. The diagram has one state line rather than two, which is the visual summary of the difference from Figure 7.4.

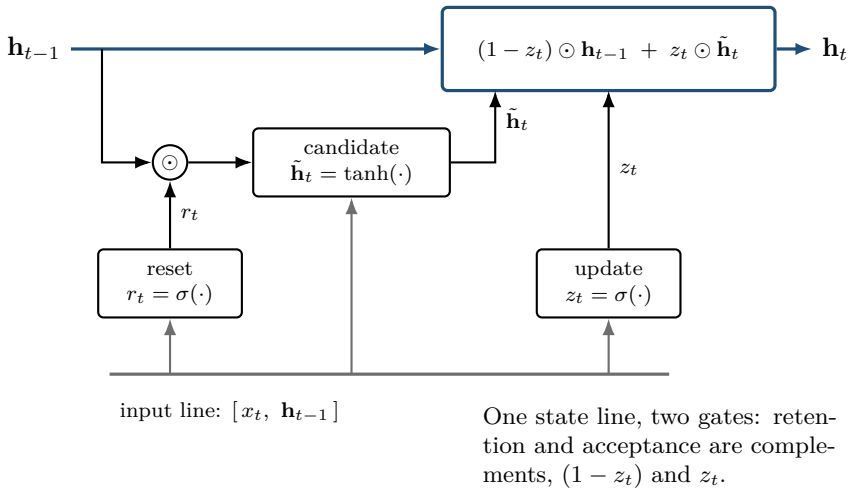


Figure 7.5: The gated recurrent unit of equations (7.25) to (7.27). The previous state is both carried forward along the top line and, after passing the reset gate, used to form the candidate. The blue box is the convex combination of equation (7.25), in which the update gate divides the new state between the retained old state and the candidate.

7.4.4 Comparison of the three cells

Table 7.1 places the three cells side by side: what each carries between steps, how many gates it has, how many parameters a unit costs, and what the cell is good for. The parameter column uses $d(m + d + 1)$ as the cost of one affine recurrent map with input dimension m and state dimension d, evaluated at $m = 5$ and $d = 32$; the plain cell needs one such map, the GRU three and the LSTM four.

The ordering in the table is one of complexity, not of quality. Neither gated cell is universally superior. A GRU has fewer parameters and often reaches a given validation error in less training time, which matters when the model is retrained nightly. An LSTM offers finer control over

Table 7.1: The three recurrent cells of this chapter. Parameter counts are for input dimension $m = 5$ and state dimension $d = 32$, excluding the readout layer.

Cell	Carried between steps	Gates	Parameters	Character
Plain	hidden state $\mathbf{h}_t$	none	1 216	lowest complexity; emphasizes recent history; fails on long dependencies
GRU	hidden state $\mathbf{h}_t$	update z_t, reset r_t	3 648	medium complexity; gating with fewer parameters; often trains faster
LSTM	hidden state $\mathbf{h}_t$ and cell state $\mathbf{c}_t$	forget f_t, input i_t, output o_t	4 864	highest complexity; separate store allows information to be held without being exposed

memory, because retention and acceptance are independent and because the output gate lets the cell store something it is not currently using. Which wins on a particular fleet is an empirical question, settled by the rolling-origin comparison of Chapter 9 and not by argument. A claim that the LSTM is intrinsically the better model is not supportable.

7.5 Architectures for forecasting

A recurrent cell is a component, not a forecaster. This section covers the ways the cell is wired into a model that answers an operational question, the inputs that are known in advance, and the two choices that most often decide whether the result is any good: the loss and the lookback window.

7.5.1 Sequence-to-one and sequence-to-sequence

The simplest wiring is sequence-to-one. The previous 24 hours $x_{t-23}, \ldots, x_t$ are read in order, the final hidden state $\mathbf{h}_t$ is taken as a summary of the window, and a dense layer maps it to a single number, for example the flexibility three hours ahead, $\hat{F}_{t+3}$. Everything in the window that matters must survive into the final state, since nothing else is passed on.

The second wiring is sequence-to-sequence. The same encoder reads the past window, but the readout produces a whole trajectory, $\hat{P}_{t+1}, \ldots, \hat{P}_{t+12}$, either from one multi-output dense layer or from a second recurrent network that is stepped forward over the forecast horizon. This is the form an optimizer needs. A dispatch problem does not consume a single number; it consumes a trajectory over the hours it is scheduling, and a model that emits the trajectory jointly can represent the fact that consecutive hours are related.

Figure 7.6 contrasts the two. Both panels read the same input sequence on the left, drawn as three boxes with an ellipsis, into a recurrent encoder. In the upper panel the final state passes through one dense layer to a single output. In the lower panel the state initializes a decoder that emits twelve values, and a second input box above the decoder supplies the future information discussed in Section 7.5.3. The structural difference is the width of the output and the presence of that second input.

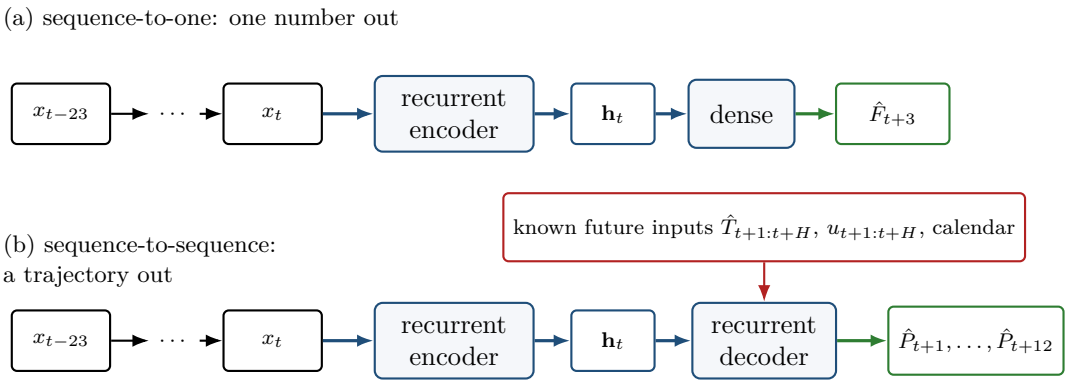


Figure 7.6: Two forecasting architectures built on the same recurrent encoder. Panel (a) is sequence-to-one: the final hidden state passes through a dense layer to a single target. Panel (b) is sequence-to-sequence: the state initializes a decoder that emits a trajectory, and the red box supplies the future inputs of Section 7.5.3, which are known or forecast at issue time.

7.5.2 Multi-horizon forecasts are coupled

Consider forecasting fleet flexibility for the four hours 16:00 to 20:00, as the vector

$$[\,\hat{F}_{16},\ \hat{F}_{17},\ \hat{F}_{18},\ \hat{F}_{19},\ \hat{F}_{20}\,]. \tag{7.28}$$

These five numbers are not five separate forecasting problems. The flexibility at 18:00 depends on whether the fleet was dispatched at 16:00 and 17:00, because a dispatched fleet arrives at 18:00 with its thermal slack partly spent. Five independently fitted hourly models cannot represent that dependence; they will each predict a plausible marginal value, and the vector they jointly produce can correspond to no physically reachable trajectory.

Stated properly, the object being forecast is a trajectory conditional on a control sequence, which is the structure Chapter 1 introduced when it put the planned action u inside the information set, and which Chapter 11 returns to at fleet scale. A sequence-to-sequence model is a natural fit because it emits the whole trajectory from one state and can be given the planned control over the horizon as an input. The underlying problem is a dynamic control problem, and treating it as a row of independent hourly regressions discards its most important structure.

7.5.3 Known future inputs

Some information about the target period is available at issue time. At 14:00, forecasting 17:00, the calendar position of 17:00 is known exactly, the aggregator's own planned control $u_{17:00}$ is known because the aggregator decides it, and a weather forecast $\hat{T}_{17:00}$ is available. These are not lagged variables and they are not unknown; a model that ignores them is discarding information that is in the information set $\mathcal{I}_t$.

A serviceable architecture therefore has two input paths, the historical sequence and the known future,

$$\underbrace{x_{t-L+1},\ \ldots,\ x_t}_{\text{observed past}} \qquad \text{and} \qquad \underbrace{\hat{T}_{t+1:t+H},\ u_{t+1:t+H},\ z_{t+1:t+H}}_{\text{known or forecast future}}, \tag{7.29}$$

with z the calendar vector. Written out, the model is

$$\mathbf{h}_t = f_\theta(x_t,\ \mathbf{h}_{t-1}), \qquad x_t = [\,P_t,\ T_t,\ T_{\text{in}},\ u_t,\ N_{\text{avail}}\,]^{\mathsf{T}}, \tag{7.30}$$

$$\hat{y}_{t+h} = g_\theta(\mathbf{h}_t,\ \hat{T}_{t+h},\ u_{t+h},\ z_{t+h}). \tag{7.31}$$

The encoder summarizes what has happened, including how many devices were available and what was commanded; the readout combines that summary with what is known about the target hour. Equations (7.30) and (7.31) are close to the shape of a serious forecasting model for a controllable fleet, and the asymmetry in them is worth noting: the past enters through a learned state, the future enters directly, because the future values are known rather than inferred.

7.5.4 Training the network

Fitting is the familiar procedure. Over N training examples the squared error loss is

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{7.32}$$

with the absolute error $\frac{1}{N}\sum_i |y_i - \hat{y}_i|$ the usual alternative when the target has outliers that should not dominate the fit; Chapter 9 discusses which error measure to optimize and which to report, and Chapter 10 replaces both with a loss whose minimizer is a quantile. Gradient descent then adjusts every weight in the model, including those inside the gates, the state transformations and the readout, to solve

$$\theta^* = \arg\min_{\theta} \; \mathcal{L}(\theta). \tag{7.33}$$

Nothing here is special to recurrent models. The gradient is obtained by backpropagation through time as in equation (7.9), which is why the gating of Section 7.3 affects not only what the model can represent but whether the optimizer can find it.

7.5.5 The lookback window

The lookback window L is a modeling decision and is routinely overlooked. If each training example consists of the previous $L = 24$ hours, the model sees 24 hours and nothing more. Weekly structure lives at a lag of 168 hours, so a 24-hour window cannot represent it: the information is not in the input, and no amount of training will recover it. A sequence model has no memory beyond the window it is shown. The hidden state is persistent within an example, not across examples, unless the implementation explicitly carries state from one window to the next.

The remedy is not simply a longer window. Setting $L = 1000$ hours introduces four costs. Training becomes harder, because the gradient must traverse a thousand steps and the argument of Section 7.3.1 applies even to gated cells once the sequence is long enough. Most of the window is irrelevant, and irrelevant inputs add variance rather than information. Computation grows linearly in L for every forward and backward pass. And the ratio of parameters to effective information worsens, so the overfitting risk of Section 7.6.5 grows. The window should reflect the plausible memory of the physical system: for thermostats, hours to a day or two, since that is the range of building thermal time constants and of dispatch after-effects. Weekly and seasonal structure is better supplied as a calendar feature than extracted from a seven-day sequence, which is the subject of the next subsection.

7.5.6 Sequence inputs and explicit context

The productive arrangement combines both kinds of input. The sequence path carries the quantities whose trajectory matters, P_t, T_t, T_{in}, u_t. A separate context path carries the variables that are better stated than inferred: the hour of day and day of week in the cyclic encoding of Chapter 4, a holiday indicator, the number of available devices N_{avail}, the equipment class

or cohort identifier, and the tariff in force. The context vector is concatenated with the final hidden state before the readout.

The reason is economy. Asking a recurrent network to infer the day of the week from a 168-step sequence spends capacity and data on something known exactly from the timestamp. Using a sequence model does not mean abandoning the domain knowledge of the previous chapters; it means spending the learned capacity on the part of the problem where the summary is genuinely unknown, which is the trajectory, and stating the rest.

7.6 When to use a recurrent model

The model class is now defined; what remains is the judgment about whether to use it. This section gives the conditions on both sides, places recurrent models in a hierarchy of ways to represent system memory, states what a learned state does not provide, and closes with a worked design and the benchmarks it must beat.

7.6.1 Conditions that favor a recurrent model, and conditions that do not

Table 7.2 lists the five situations in which a recurrent model may genuinely help. The left column states the condition, the right column says why an engineered feature set struggles with it. The common element across the rows is that the useful summary of history is either unknown or not fixed, so choosing it in advance either fails or requires a feature list long enough to cover every case.

Table 7.2: Conditions under which a recurrent model may improve on a feature-engineered one.

Condition	Why engineered features struggle
The thermal trajectory $T_{\text{in}}(t-5{:}t)$ carries information absent from the current state	The informative quantity is the shape of the path, not any one lag; summarizing it requires guessing which functional of the path matters
Control history matters: a curtailment changes both the response and the rebound	The relevant summary depends on event duration, on weather since, and on how saturated the fleet was at the start
The useful memory length varies with the situation	A fixed lag list must be sized for the worst case and carries dead weight otherwise
Several signals co-evolve: weather, indoor temperature, power, control	Cross-signal interactions across time become deep conjunctions of threshold tests
The target is a multi-step trajectory, not one number	Independent per-horizon models produce vectors that no reachable trajectory matches

The negative case deserves equal weight. Suppose a gradient-boosted model on the feature vector

$$X_t = [\, P_{t-1},\ P_{t-24},\ P_{t-168},\ \hat{T}_{t+h},\ \text{hour},\ D_t \,]^{\mathsf{T}} \tag{7.34}$$

already forecasts well. Then a recurrent model buys a longer and less stable training procedure, a larger hyperparameter search over state dimension, number of layers, window length, learning rate and batch size, a harder debugging problem when a forecast goes wrong in production, and the loss of the feature-importance diagnostics of Chapter 5. If the improvement is negligible, the correct decision is not to use it. Operational simplicity is a real benefit, and it is paid for in incidents avoided rather than in validation error.

7.6.2 Three ways of representing history

The three model classes of this book so far differ in exactly one respect. A feature-engineered model with a tree ensemble builds X_t by hand and learns

$$\hat{y} = f(X_t); \tag{7.35}$$

an autoregressive model fixes the functional form and learns the coefficients,

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \varepsilon_t; \tag{7.36}$$

a recurrent model learns a state and reads the forecast off it,

$$\mathbf{h}_t = f(x_t,\ \mathbf{h}_{t-1}), \qquad \hat{y}_t = g(\mathbf{h}_t). \tag{7.37}$$

All three are supervised models fitted by minimizing a loss on historical data. The distinction is how history is represented: chosen and summarized by a person, assumed to act through a fixed linear memory, or compressed into a learned state.

7.6.3 A hierarchy of representations of system memory

Arranged by how much of the memory structure is specified rather than learned, the four options form the hierarchy of Table 7.3. The table's columns are the representation, what a person supplies, and what the data supplies. Read downward, the amount specified increases and the amount learned decreases; read upward, flexibility increases and interpretability falls. The highest level is the physical state model of equation (7.22), in which a person specifies what the state means and the data only calibrates parameters.

Table 7.3: Four representations of system memory, ordered by how much structure is specified in advance.

Level	Representation	Supplied by the modeler	Supplied by the data
1	explicit lags, P_{t-1}, P_{t-24}	which lags matter	the response to each lag
2	autoregressive model, Chapter 6	the order and the linear form	the coefficients ϕ_k
3	recurrent state $\mathbf{h}_t$	the state dimension and the window L	the state's contents and its dynamics
4	physical state, $x_{t+1} = F(x_t, u_t, w_t)$	what the state means and the dynamics	the parameters, such as RC and efficiencies

The levels combine. A standard construction for engineered systems keeps a physical model for the part that is understood and a learned recurrent model for the part that is not. Take a first-order thermal model for the indoor temperature,

$$C\,\frac{\mathrm{d}T_{\mathrm{in}}}{\mathrm{d}t} = \frac{T_t - T_{\mathrm{in}}}{R} - Q_{\mathrm{HVAC}}, \tag{7.38}$$

with R a thermal resistance, C a capacitance and Q_{HVAC} the cooling delivered. Equation (7.38) predicts the physics and is wrong about everything behavioral: occupancy, manual setpoint changes, window opening, opt-outs. The hybrid forecast is

$$\hat{P}_{t+h} = \underbrace{P^{\mathrm{phys}}_{t+h}}_{\text{from equation (7.38)}} + \underbrace{g_\theta(\mathbf{h}_t)}_{\text{learned correction}}, \tag{7.39}$$

with the recurrent model fitted to the residual of the physical model rather than to the load itself. The physical term supplies the extrapolation behavior and the interpretability; the learned term absorbs what the physics omits. In engineered systems with a credible physical model this is usually the more attractive construction, and it is the residual-modeling idea of Chapter 6 with a learned nonlinear memory in place of a linear one.

7.6.4 A learned state is not a state estimate

A recurrent network constructs $\mathbf{h}_t$ from a record of observations, and it is tempting to call that a state estimate. The resemblance is real but loose, and the difference matters. A probabilistic state estimator, such as a Kalman filter, returns a distribution

$$p\left(x_t \mid y_{1:t}\right), \tag{7.40}$$

over a state whose components have units and meanings, together with a covariance that says how well each component is pinned down by the measurements. A standard recurrent network returns a single deterministic vector with no units, no stated meaning and no covariance. It is a learned representation, not an inference.

The operational consequence is immediate. Suppose the model reports $\hat{F} = 50\,\mathrm{MW}$ of flexibility for the evening peak. What does it say about the uncertainty of that number? By itself, nothing. The 50 MW could be a confident interpolation inside well-populated training conditions, or an extrapolation into weather the model has never seen, or the output of a network half of whose telemetry inputs were missing and silently imputed. The architecture does not distinguish these cases, and the forecast looks the same in all three. Sequence modeling and probabilistic forecasting are separate problems: the first chooses how history is represented, the second attaches an honest distribution to the answer, and Chapter 10 takes up the second. A recurrent network can be made probabilistic, by predicting quantiles or distribution parameters instead of a point, but that is an additional decision and not a property it has by default.

7.6.5 Overfitting and extreme events

A gated recurrent model has many parameters, and the data situation in distributed energy resource forecasting is often poor in the way that punishes parameters most. Telemetry is sparse and intermittent. Fleet composition changes as customers enroll and leave, so the system being modeled is not stationary. Above all, the events of interest are rare: a history can contain a million normal timesteps and only two hundred dispatch events worth learning from. The nominal sample size is large and the effective sample size for the question being asked is small, and a flexible network will happily fit the quirks of two hundred events.

The consequence lands precisely where it hurts. Average validation error over a year can look excellent while the model has seen only a handful of days above 40°C. The hottest day of the next summer arrives, the utility's need is at its maximum, the forecast matters more than on any other day of the year, and the model is operating in the region where it is least supported. Two conclusions follow for practice. Validation must be stratified by regime rather than averaged, which is the rolling-origin and regime-conditional evaluation of Chapter 9; and the forecast needs an honest uncertainty statement that widens where support is thin, which is Chapter 10, together with the explicit treatment of conditions the training data does not cover that a later chapter gives.

7.6.6 Conceptual questions

Three questions recur, one on model choice, one on interpretation and one on mechanism.

Conceptual questions

Why would you use an LSTM instead of gradient boosting for a DER aggregator's flexibility forecasting?
A strong answer makes the case conditional on evidence. One would use a recurrent model if validation shows that the trajectory of recent measurements carries predictive information that engineered lags and current-state features do not capture: thermal accumulation over the afternoon, HVAC cycling patterns, recent curtailment and the rebound it sets up. The answer should then concede the other branch explicitly: if a gradient-boosted model with careful features performs comparably, the simpler model is preferable operationally, because it trains faster, is easier to debug and supports feature-importance diagnostics. The decision belongs to the validation table, not to the model class.

Conceptual questions

What does the LSTM cell state represent physically?
The answer to avoid is "the building's thermal state". The cell state is a learned latent state. It may well encode thermal condition, HVAC operation, occupancy patterns and control history, because encoding them reduces the forecast error, but its components have no predefined physical meaning and no units, and nothing guarantees that any one component corresponds to any one physical quantity. Claiming physical meaning for it requires that the architecture or the training objective impose that meaning, for example by constraining a component to reconstruct a measured temperature.

Conceptual questions

Why are LSTMs better than basic RNNs for long sequences?
Training a basic recurrent network multiplies gradients through the same recurrent transition at every step, as in equation (7.9), so over many steps they vanish or explode and the network cannot learn dependencies that span the sequence. An LSTM adds a gated cell-state path along which information is retained with almost no transformation, equation (7.11), so the backward path is a product of retention factors near one instead of a product of weight matrices. The mechanism, not the parameter count, is the answer.

7.6.7 A design example and its benchmarks

Make the whole chapter concrete with one design. The task: at 14:00, forecast fleet flexibility for the four hours 17:00 to 20:00. Figure 7.7 lays out the stages. The first box is the input sequence, the seven hourly records from 08:00 to 14:00, each carrying metered power, outdoor temperature, average indoor temperature, duty cycle and control signal. The second box is the recurrent encoder, which maps that sequence to the state $\mathbf{h}_{14:00}$ in the third box. The fourth box is the known future information for 17:00 to 20:00: the weather forecast, the planned control and the calendar. The fifth box is the readout that combines state and future information, and the sixth is the output vector of four flexibility forecasts. The lower row of the figure is the benchmark set of the following paragraph, feeding a comparison box.

A design is not a result. Before the complexity is accepted, the model must be compared against four benchmarks. The first is a seasonal baseline: the flexibility delivered in similar past events at the same hour, in the sense of Chapter 3. The second is a linear model $F = X_t\boldsymbol{\beta}$ on the engineered feature vector. The third is gradient boosting on the same features, which is the strongest of the cheap competitors. The fourth is any of the first three with an autoregressive correction applied to its residuals, the construction of Chapter 6, since much of the apparent gain from a complex model is sometimes just residual autocorrelation that a cheap correction also captures. The recurrent model earns its place only if it delivers a meaningful out-of-sample

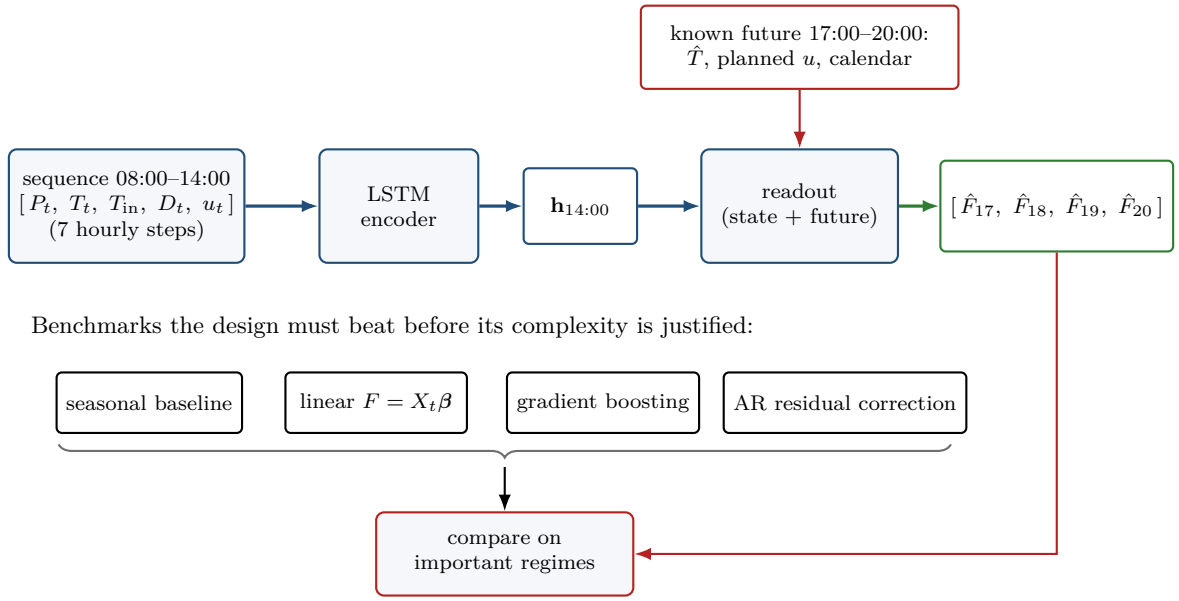


Figure 7.7: The design of Section 7.6.7. The upper row is the forecasting model: a seven-step input sequence, a recurrent encoder, the resulting state, a readout that also consumes the known future inputs, and a four-hour flexibility trajectory. The lower row is the benchmark set against which the design is evaluated.

improvement, and in particular an improvement on the regimes that matter: hot days, event days, and the hours immediately after a dispatch. An improvement in the annual average with no improvement on those regimes is not a reason to deploy it.

7.7 What to retain

Definition 7.1 (Recurrent forecasting model)**.** A supervised model with engineered features learns $\hat{y}_t = f(X_t)$, where a person has chosen how history is summarized. A recurrent model learns the summary itself, carrying a hidden state $\mathbf{h}_t = f(x_t, \mathbf{h}_{t-1})$ and reading the forecast off it. Plain recurrent cells cannot learn long dependencies because backpropagation through time multiplies gradients by a factor below one at every step; the long short-term memory cell repairs this with a gated additive memory, $\mathbf{c}_t = f_t \odot \mathbf{c}_{t-1} + i_t \odot \tilde{\mathbf{c}}_t$, which is retained old memory plus accepted new information, and the gated recurrent unit achieves much of the same with two gates and one state. Sequence models become interesting when thermal condition, control history, rebound and trajectory-dependent effects cannot be summarized by a few lags, and the model to use is the simplest one that captures the predictive structure actually needed.

Table 7.4 collects the chapter's objects with the subsection that defines each.

7.8 Exercises

Exercise 7.1. A plain recurrent cell has an effective per-step gradient gain of $\gamma = 0.6$. Using the product form of equation (7.9), find the number of steps after which the gradient factor falls below 10^{-4}. Repeat for a gated cell path whose forget gate sits at $f = 0.97$, using equation (7.20). Convert both answers to hours for a 15-minute series and state which one covers a dependency between a 10:00 curtailment and a 17:00 forecast.

Exercise 7.2. A three-component cell state holds $\mathbf{c}_{t-1} = [\,10,\ 4,\ -2\,]$. One step produces $f_t = [\,0.95,\ 0.1,\ 0.8\,]$, $i_t = [\,0.3,\ 0.9,\ 0.5\,]$ and $\tilde{\mathbf{c}}_t = [\,2,\ -6,\ 4\,]$. Compute $\mathbf{c}_t$ from equation (7.11).

Table 7.4: The objects of a recurrent forecasting model.

Object	Definition	Subsection
$\mathbf{h}_t$	hidden state, $\mathbf{h}_t = \tanh(W_x x_t + W_h \mathbf{h}_{t-1} + b_h)$	7.2.1
W_x, W_h, W_y	input, recurrent and readout weights, shared across steps	7.2.1
L	lookback window: the number of past steps each example contains	7.5.5
BPTT	backpropagation through time, equation (7.9)	7.3.1
γ^k	per-step gradient gain raised to the number of steps back	7.3.1
$\mathbf{c}_t$	cell state, $\mathbf{c}_t = f_t \odot \mathbf{c}_{t-1} + i_t \odot \tilde{\mathbf{c}}_t$	7.3.2
$\odot$	element-wise product; each memory component has its own gate	7.3.2
f_t	forget gate; constant f gives memory time constant $-1/\ln f$	7.3.3
i_t, $\tilde{\mathbf{c}}_t$	input gate and candidate memory; the written part is $i_t \odot \tilde{\mathbf{c}}_t$	7.3.4
o_t	output gate; $\mathbf{h}_t = o_t \odot \tanh(\mathbf{c}_t)$ exposes part of the memory	7.3.5
z_t, r_t	gated recurrent unit update and reset gates	7.4.3
g_θ	readout from the state and the known future inputs, equation (7.31)	7.5.3

Then use equation (7.16) to give each component's effective memory time constant in hours for an hourly series, and say which component could plausibly be tracking building thermal loading and which could be tracking compressor state.

Exercise 7.3. A model has input dimension $m = 6$ and state dimension $d = 48$, and a dense readout to four flexibility horizons. Using $d(m + d + 1)$ per recurrent affine map, count the parameters of a plain cell, a gated recurrent unit and a long short-term memory cell, add the readout, and compare the totals with the number of parameters in the gradient-boosted alternative of equation (7.34). Given that the training history contains 200 dispatch events, comment on Section 7.6.5.

Exercise 7.4. A fleet's load has clear weekly structure. One proposal is a recurrent model with $L = 168$ hourly steps; another is $L = 24$ steps plus the cyclic calendar features of Chapter 4 on the context path of Section 7.5.6. State the ratio of forward-pass cost between the two, the argument from Section 7.5.5 for the second, and one circumstance in which the first would nonetheless be the better choice. Then list the four benchmarks of Section 7.6.7 that either proposal must beat.

Chapter 8

Attention, transformers and the temporal fusion transformer

8.1 The bottleneck of a single recurrent state

The recurrent networks of Chapter 7 read a sequence one step at a time and carry everything they have learned about the past in a single vector. The forecast is then a function of that vector alone. The question of this chapter is

Which parts of the history matter for this particular forecast, and can the model be allowed to decide that for itself?

The chapter answers it with one mechanism. Instead of compressing the history first and predicting from the compressed summary, the model compares the current forecasting need against each past observation and forms a weighted combination of them. The weights are computed anew for every forecast, so the notion of relevant history becomes a learned, context-dependent quantity rather than a fixed architectural choice. That mechanism is attention; the architecture built around it is the transformer; and the version adapted to forecasting is the temporal fusion transformer.

Jargon: attention, query, key, value, transformer, multi-horizon

Attention is a layer that, for each prediction, computes a set of nonnegative weights summing to one over a collection of candidate inputs, and returns the weighted average of those inputs. The weights come from comparing a *query* (a vector describing what the prediction needs) against a *key* for each candidate input (a vector describing what that input offers); the thing averaged is the *value* of each input (a vector describing what it contributes). *Self-attention* is the case in which the queries, keys and values are all derived from the same sequence. A *transformer* is a sequence model built from stacked self-attention and feed-forward layers, with no recurrence. *Multi-horizon* forecasting means producing $\hat{y}_{t+1}, \ldots, \hat{y}_{t+H}$ in one pass rather than one horizon at a time.

Key idea

A recurrent network must compress all useful history into $\mathbf{h}_T$ before predicting. Attention does not compress first: it compares the forecasting need directly against the historical observations and weights them by relevance. Relevance itself becomes learned and context dependent.

This section states the limitation that motivates the mechanism and makes it concrete on the thermostat-fleet forecast used throughout the book.

8.1.1 The recurrent bottleneck

The recurrent computation of Chapter 7 is the chain

$$x_1 \longmapsto \mathbf{h}_1 \longmapsto \mathbf{h}_2 \longmapsto \cdots \longmapsto \mathbf{h}_T, \qquad \mathbf{h}_t = f(\mathbf{h}_{t-1}, x_t), \tag{8.1}$$

and the prediction is made from $\mathbf{h}_T$. The state $\mathbf{h}_t$ is a vector of fixed width, typically tens to a few hundred numbers, and it is the only route by which anything observed at step 1 can reach the forecast made at step T. Every feature of the history that the forecast needs must therefore survive T successive applications of f without being overwritten. Gating, the mechanism that makes an LSTM or a GRU work, improves the odds of survival but does not change the structure: the information still has to pass through one vector of fixed width, and it still has to pass through every intermediate step.

Attention starts from a different question. If the forecast needs the load observed one week ago, there is no evident reason to compress that observation into a state, propagate the state through 168 updates, and rely on the observation still being recoverable. The alternative is to compare the current forecasting need against the historical observations themselves and retrieve the ones that match.

Figure 8.1 contrasts the two. Panel (a) is the recurrent chain of equation (8.1): the inputs $x_1, \ldots, x_T$ enter from below, the hidden states form a left-to-right chain, and the single arrow out of $\mathbf{h}_T$ is the only path from any input to the forecast. Panel (b) is attention: the same inputs lie in a row, a query derived from the forecasting need enters from the right, and every input is connected to the output by one arrow whose weight the query sets. The structural difference visible in the figure is path length. In panel (a) the path from x_1 to the forecast has length T; in panel (b) it has length one, for every input equally.

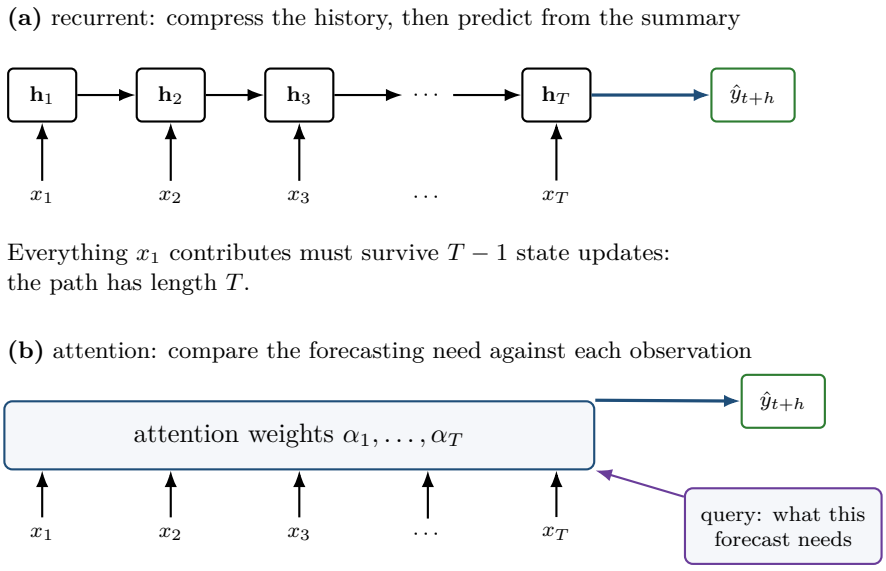


Figure 8.1: The compression bottleneck of a recurrent network against the direct retrieval of attention. In (a) the inputs $x_1, \ldots, x_T$ enter the hidden-state chain of equation (8.1) and only $\mathbf{h}_T$ is read. In (b) the inputs are read directly, weighted by $\alpha_1, \ldots, \alpha_T$, which a query derived from the forecasting need determines. The number of inputs is reduced to five for legibility; in practice T is the length of the history window.

8.1.2 A weighted history for a thermostat forecast

Consider the aggregator's standing problem: at 14:00 the operator needs a forecast of the thermostat fleet's flexibility at 17:00, and the model is given the previous 48 hours of telemetry. Not every one of those 48 hourly observations carries the same amount of information about the target. Five kinds of observation stand out.

- 13:00 today, because it carries the fleet's current thermal state: indoor temperatures and compressor status an hour ago are close to what they will be when the event starts.
- 12:00 today, because the recent duty cycle D_t says how hard the fleet is already working.
- 17:00 yesterday, because it is the same target hour under broadly similar conditions, which is the seasonal-persistence information of Chapter 3.
- Yesterday's dispatch event, because a fleet curtailed recently arrives at the next event with a depleted thermal buffer.
- The most recent earlier period with a similar outdoor temperature, because the weather sensitivity of the response is what sets its magnitude.

Against those, 03:00 yesterday carries almost nothing: the fleet was idle, the buildings were cool, and nothing that happened then survives into this afternoon.

Figure 8.2 shows one plausible set of weights over seven of the 48 candidate timesteps. The horizontal axis lists the timesteps and the vertical axis is the weight each receives; the weights are nonnegative and sum to one, so the figure is a distribution over the history. Three features of the figure are the point of the chapter. Mass concentrates on the last few hours before the issue time, where the state information lives. A secondary peak sits on 17:00 yesterday, the same hour as the target, with almost no mass on the hours around it. And the early hours of yesterday receive weights near zero. The forecast is then built from the weighted combination

$$\text{retrieved information} = \sum_j \alpha_j \, v_j, \qquad \alpha_j \geq 0, \qquad \sum_j \alpha_j = 1, \tag{8.2}$$

with v_j the information carried by timestep j. No recurrent network produces equation (8.2); a recurrent network produces one state in which all seven contributions have already been merged, and the merge was performed before the model knew which forecast it was being asked for.

8.2 The attention mechanism

This section defines the mechanism that produces weights such as those of Figure 8.2: the three vectors it compares, the equation that turns a comparison into a weighted average, the scaling that keeps the comparison numerically stable, and what happens when a sequence attends to itself.

8.2.1 Query, key and value

Attention is organized around three objects, each a learned linear function of the inputs.

Query
: What is being looked for. One query per prediction, or per position in self-attention: a vector encoding the forecasting need.

Key What a given historical observation contains. One key per candidate input: a vector encoding what that observation offers to a prospective query.

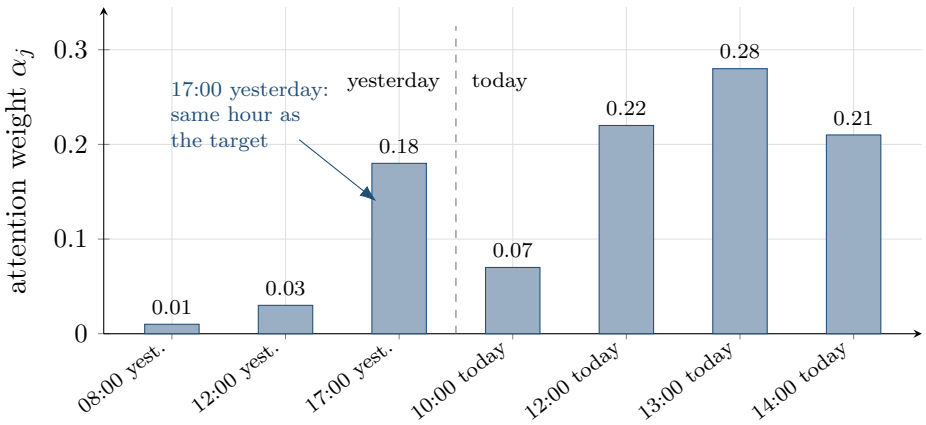


Figure 8.2: Illustrative attention weights over seven of the 48 hours of history available at 14:00, for a forecast of fleet flexibility at 17:00. The weights are those of Section 8.1.2; they are nonnegative and sum to one, so the bars form a distribution over the history. Mass concentrates on the hours just before the issue time and on the same hour one day earlier.

Value
: What is retrieved if the observation turns out to be relevant. One value per candidate input: the content actually passed on.

The mechanism is a soft lookup. The query is compared against every key to produce a relevance score; the scores are normalized into weights; the weights are applied to the values. The separation of key from value is what makes this more than a similarity search: the field used for matching need not be the field that is retrieved, exactly as a database index is not the record it points to.

For the thermostat forecast of Section 8.1.2, the query at 14:00 encodes something like *what history is useful for predicting 17:00 load on a very hot weekday?* A timestep whose key encodes *17:00, hot, weekday* matches it strongly. A timestep whose key encodes *03:00, mild, weekend* matches it weakly. The first contributes most of the retrieved information and the second almost none. The model is not remembering the whole history equally well; it is locating the historically relevant parts of it.

Figure 8.3 draws that comparison for the two timesteps. The query is the box on the left. The two rows in the middle are the two candidate timesteps, each with its key on the left and its value on the right. The arrows from the query to the keys carry relevance scores, and the arrows from the values to the output box carry the weights those scores produce. The figure shows the asymmetry that matters: the two values are comparable objects, and the difference in their contribution to the output comes entirely from the match between query and key.

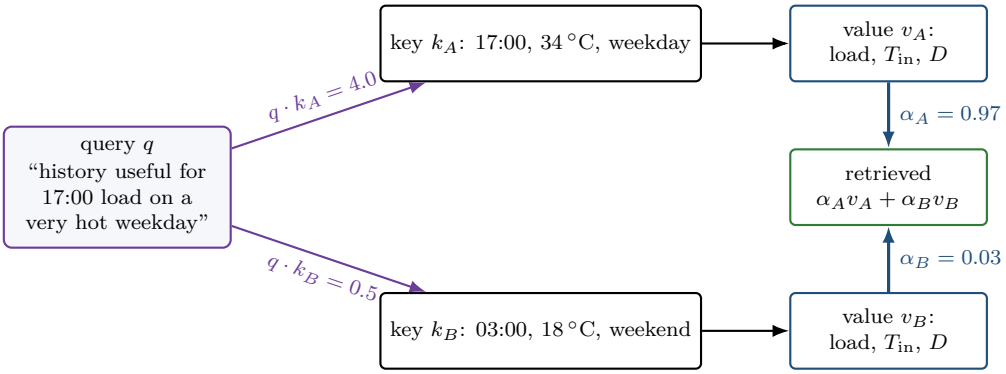
8.2.2 The attention equation

Collect the queries, keys and values of a layer as the rows of three matrices $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$, with d_k the width of a query or key vector. Scaled dot-product attention is then

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\mathsf{T}}{\sqrt{d_k}}\right)\mathbf{V}. \tag{8.3}$$

Equation (8.3) has three stages, and reading them in order is the whole mechanism.

Similarity. The product $\mathbf{Q}\mathbf{K}^\mathsf{T}$ is a matrix of inner products: entry (i, j) is the dot product of query i with key j, which is large when the two vectors point in the same direction. One row of this matrix is the set of relevance scores for one prediction.



The field used for matching (the key) is not the field retrieved (the value).

Figure 8.3: Query, key and value for two candidate timesteps of the thermostat history. The query describes the forecasting need; each timestep's key describes what it offers; each timestep's value is what it contributes. The weights are the softmax of the two relevance scores, $\alpha_A = e^{4.0}/(e^{4.0} + e^{0.5}) = 0.97$ and $\alpha_B = 0.03$. Scores are illustrative.

Normalization. The softmax is applied along each row,

$$\alpha_{ij} = \frac{\exp(s_{ij})}{\sum_{j'} \exp(s_{ij'})}, \qquad s_{ij} = \frac{q_i \cdot k_j}{\sqrt{d_k}}, \tag{8.4}$$

which turns real-valued scores into nonnegative weights summing to one. This is where the distribution of Figure 8.2 comes from.

Retrieval. Multiplying the weight matrix by $\mathbf{V}$ forms, for each prediction, the weighted average $\sum_j \alpha_{ij} v_j$ of equation (8.2). Nothing else happens in the layer: attention is a data-dependent weighted average, and its only nonlinearity is the softmax.

8.2.3 The softmax of three relevance scores

A three-element example fixes the arithmetic. Suppose a query produces the relevance scores $s = [1, 2, 4]$ for three historical observations. Exponentiating gives $e^1 = 2.718$, $e^2 = 7.389$ and $e^4 = 54.598$, whose sum is 64.705, so the weights are

$$\alpha = \left[\frac{2.718}{64.705}, \frac{7.389}{64.705}, \frac{54.598}{64.705}\right] = [0.042,\ 0.114,\ 0.844], \tag{8.5}$$

and the retrieved information is $0.04\, v_1 + 0.11\, v_2 + 0.85\, v_3$. The third observation dominates.

Figure 8.4 shows the transformation. The left panel plots the three scores and the right panel the three weights of equation (8.5); the vertical scales differ because the left panel is in arbitrary score units and the right is a probability. The comparison shows what the exponential does: a score gap of two, from 2 to 4, becomes a weight ratio of 7.4, so modest differences in similarity become large differences in contribution. The softmax is a soft selection, and how soft or hard it is depends entirely on the scale of the scores it is given.

8.2.4 Scaling by the square root of the key dimension

The division by $\sqrt{d_k}$ in equation (8.3) addresses exactly the sensitivity just described. A dot product of two d_k-dimensional vectors is a sum of d_k terms. If the components are roughly independent with unit variance, the sum has variance d_k and therefore standard deviation $\sqrt{d_k}$:

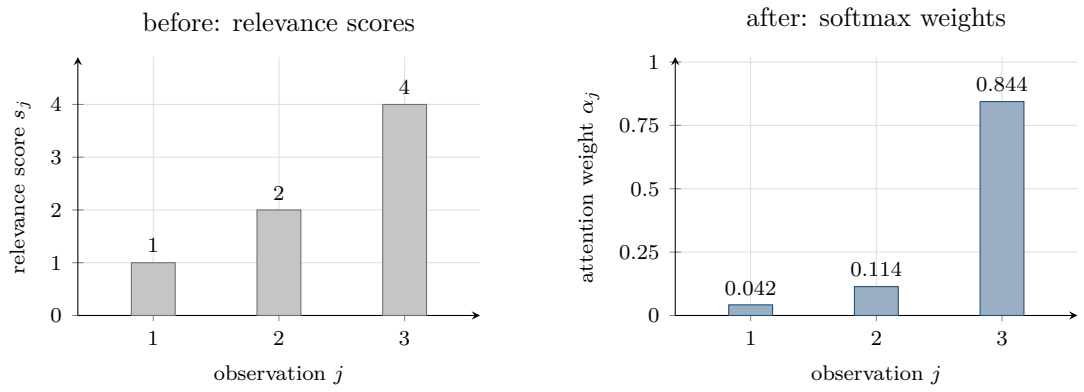


Figure 8.4: The softmax of the relevance scores $[1, 2, 4]$. The left panel is the scores, the right panel the weights of equation (8.5), computed from equation (8.4) with the scaling already absorbed into the scores. A score gap of two becomes a weight ratio of $e^2 = 7.39$.

raw scores grow with the width of the key vectors. At $d_k = 64$ the typical score is eight times larger than at $d_k = 1$, and a softmax over scores of that size is nearly a hard selection, placing almost all weight on a single observation. Two problems follow. The layer loses the ability to blend several moderately relevant observations, and the gradient of the softmax at a saturated point is almost zero, so training stalls.

Dividing by $\sqrt{d_k}$ restores unit scale regardless of d_k, which is why the term appears. It is a variance-stabilizing constant, not a tunable hyperparameter, and it is the reason the mechanism of equation (8.3) is called *scaled* dot-product attention.

8.2.5 Self-attention and direct long-range paths

In the form used inside a transformer, the queries, keys and values are all linear projections of the same input sequence $x_1, \ldots, x_T$. Each position asks which other positions of the same sequence are relevant to it, and receives a weighted combination of them. This is self-attention, and its consequence for a time series is the one anticipated in Figure 8.1: position t interacts with positions $t-1$, $t-24$ and $t-168$ directly, through a single weighted connection, without any of the information passing through the intervening hidden states.

The contrast with a recurrent network is a statement about path length. In a recurrent network the dependence of the prediction on x_1 is mediated by $T-1$ compositions of f, so representing a long-range dependence requires the network to preserve a signal across all of them. In self-attention the dependence is one multiplication by one weight. A long-range dependence is therefore no harder to represent than a short-range one; whether it is easier to *learn* is a separate question that depends on how much data is available, and Section 8.6.1 returns to it.

Why this structure suits energy series. The load at 17:00 depends strongly on P_{t-24}, the same hour yesterday, and on P_{t-168}, the same hour last week. Chapter 2 established those two lags as the dominant features of the autocorrelation of an energy series, and Chapter 6 used them as seasonal autoregressive terms. A recurrent network has to carry that information through 24 or 168 updates; attention connects to it directly. Three properties of fleet data make the mechanism more attractive still: strong daily and weekly repetition, a history of irregular events such as dispatches, outages and extreme weather whose relevance depends on what is being predicted, and a long control-response memory, since a curtailment two days ago can still be affecting the thermal state today.

8.2.6 Positional encoding

Self-attention as defined has no notion of order. The computation in equation (8.3) treats the rows of $\mathbf{K}$ and $\mathbf{V}$ as an unordered set: permuting the history permutes the weights identically, and the retrieved value is unchanged. For a time series that is a serious omission, because the model cannot distinguish x_1 preceding x_2 from the reverse, and cannot by itself know that one key is 24 hours old and another one week old.

Transformers therefore add a positional encoding to each input before the first attention layer: a vector that depends on the position rather than on the observation. Three kinds of position information matter for a forecasting model. Sequence position says where in the window an observation sits. Relative timing says how far it is from the issue time, which is what makes a lag of 24 distinguishable from a lag of 168. Calendar context says what hour, weekday and season the observation belongs to, which is the cyclic encoding of Chapter 4 supplied to a sequence model instead of to a tree. Once positions are encoded a key can describe *when* as well as *what*, which is precisely what the query of Figure 8.3 needs to match against.

8.3 Transformers

A transformer is an architecture built around repeated self-attention instead of recurrence. This section gives the block it is built from, the training property that makes it attractive at scale, the multiple-head extension, and the cost that limits sequence length.

8.3.1 The encoder block

The unit of a transformer is an encoder block that maps a sequence representation to another sequence representation of the same shape. It contains a self-attention layer, a position-wise feed-forward network, and around each of them a residual connection and a normalization step. Blocks are stacked, and a forecast head is placed on the final representation.

Figure 8.5 draws one block. Reading upward: the input representation of the sequence, carrying the positional encoding of Section 8.2.6, enters a multi-head self-attention layer; its output is added to its input and normalized; the result passes through a feed-forward network applied independently at each position; that output is again added and normalized. The dashed boundary marks what is stacked L times, and the two arrows on the left are the residual paths that let a block pass its input through unchanged. The absence of any arrow from one time position to the next is the structural point: there is no $\mathbf{h}_t = f(\mathbf{h}_{t-1}, x_t)$ anywhere in the diagram. All interaction between positions happens inside the attention layer, and the feed-forward network sees each position in isolation.

8.3.2 Parallel training

A recurrent network is sequential by construction: $\mathbf{h}_2$ cannot be computed before $\mathbf{h}_1$, so a training step over a window of length T costs T dependent operations no matter how much computing hardware is available. A transformer has no such dependence. During training, when the whole input window is known, the attention scores for all positions are one matrix product, and the feed-forward layer applies to all positions independently. The computation at every position therefore happens at once.

This is an engineering property rather than a statistical one, but it has statistical consequences, because it decides what size of model and dataset is practical. It is the main reason transformers displaced recurrent networks wherever data is plentiful: with large datasets, long sequences and parallel hardware, the same wall-clock budget buys a much larger model. For a dataset of a few device-years it buys nothing, because training cost was not the binding constraint in the first place.

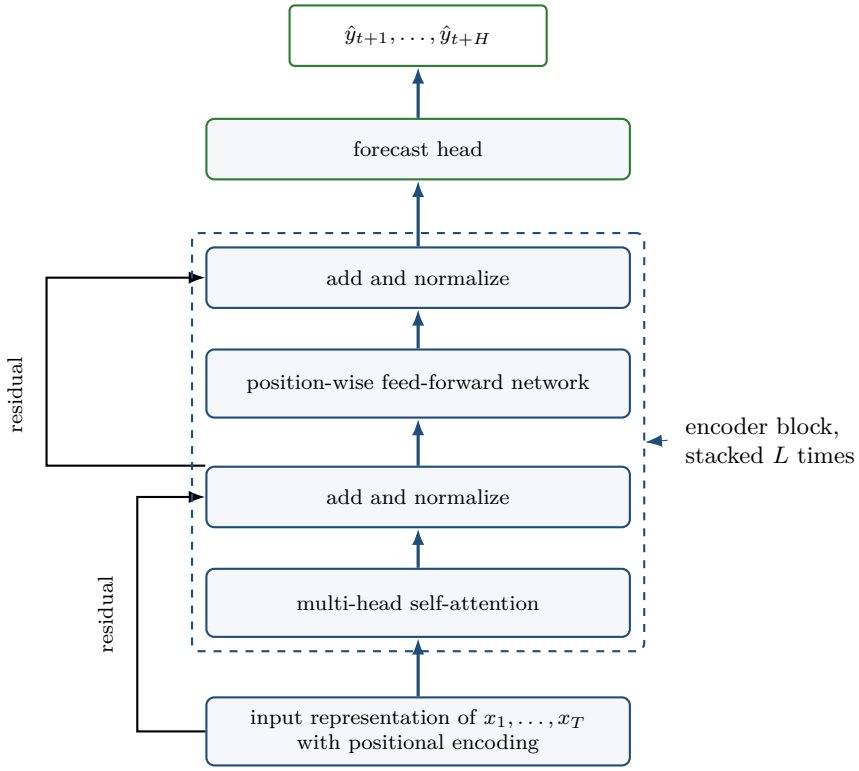


Figure 8.5: One transformer encoder block and the forecast head above it. The boxes are layers; the upward arrows are the forward path; the two left-hand arrows are residual connections, which add a layer's input to its output before normalization. The dashed boundary is what is stacked L times.

8.3.3 Multi-head attention

A single attention layer produces one set of weights per position, which forces one notion of relevance. Multi-head attention runs several attention mechanisms in parallel on the same input, each with its own learned projections to queries, keys and values, and concatenates their outputs before the feed-forward layer. Different heads can then specialize in different kinds of relevance, and the next layer combines their retrievals.

For the thermostat fleet, one might hope the heads learn to look at the recent duty cycle, at the same hour on similar hot days, at recent dispatch events, and at the current gap between indoor temperature and setpoint. Those four are plausible and interpretable decompositions of the forecasting need, and the architecture allows them. It does not produce them. Heads are initialized at random and trained only to reduce forecast error, so what they settle on is whatever reduces error, which in general does not align with any human description of the problem. The heads in a trained model are discovered patterns, not assigned roles, and Section 8.5.1 returns to how much weight may be put on reading them.

8.3.4 Sequence length and quadratic cost

The matrix $\mathbf{QK}^\mathsf{T}$ in equation (8.3) has one entry for every ordered pair of positions, so it has T^2 entries and the layer costs on the order of T^2 operations and T^2 memory. That scaling is the architecture's main practical limit. A window of $T = 100$ gives 10^4 entries and is trivial. One of $T = 8760$, a year of hourly data, gives 7.7×10^7 entries per head per layer and is heavy but feasible. One of $T = 100\,000$ gives 10^{10} and is not practical at all. A recurrent network, by contrast, costs on the order of T operations, though it must spend them sequentially.

The quadratic cost is one of the reasons specialized time-series architectures exist rather than plain transformers: they restrict attention to a subset of pairs, or attend to summaries of blocks of time, or combine a short attention window with explicit seasonal lags. The temporal fusion transformer of the next section takes a different route to the same end. It keeps the window short and spends its capacity on the structure of the inputs instead.

8.4 The temporal fusion transformer

Attention is worth its cost only under particular conditions. This section states those conditions, then describes the architecture that packages attention for forecasting: the three kinds of input it distinguishes, the variable selection that runs alongside attention, the multi-horizon output that makes it useful to an operator, and the response model that output turns out to be.

8.4.1 Conditions under which attention is worthwhile

For the ordinary problem of forecasting hourly load, a transformer is usually not needed. Given P_{t-1}, P_{t-24}, P_{t-168}, a weather forecast $\hat{T}_{t+h|t}$, the hour of the day and the day of the week, the gradient boosted ensemble of Chapter 5 typically performs very well, and a transformer fitted to the same information has nothing extra to find. The useful question is not which model is most modern. It is: *what structure exists in this problem that the simpler model cannot capture?* If the answer is none, the simpler model wins on training cost, on diagnosis, on robustness and on the time it takes to put into production.

Six conditions make attention more interesting, and they tend to appear together.

- Long history matters, so that a window of a few lags discards information.
- The important lags vary case by case, so no fixed lag set is right for every forecast.
- Several related signals must be read jointly, rather than one series with exogenous columns.

- The forecast is multi-horizon, so the whole trajectory is needed at once.
- Training data is large.
- Many related series are modeled jointly, so one model sees many sequences.

Those six describe a fleet. An aggregator with 100 000 enrolled thermostats, each with hourly telemetry, a dispatch history and a weather forecast attached, satisfies every one of them, which is why this chapter sits where it does in the book rather than as an appendix to Chapter 5.

8.4.2 Many related series

The sixth condition deserves separate statement because it changes the data situation qualitatively. Write $y_{i,t}$ for the series of device i at time t. Instead of one model per device, a single global sequence model is fitted to all of them, learning the weather sensitivity, the daily usage pattern and the control-response dynamics that the devices share, with device-level or cohort-level information supplied as an input so the model can express how they differ. The device count then multiplies the effective sample size: 100 000 devices with two years of hourly history supply on the order of 1.8×10^9 observations rather than 1.8×10^4. Multi-series and global models are the subject of Chapter 11; here they matter only as the reason a data-hungry architecture can be contemplated at all.

8.4.3 Three kinds of input

The temporal fusion transformer is an architecture designed for multi-horizon forecasting with heterogeneous inputs. Its organizing idea is a distinction this book has already drawn in another form: an input is classified by *when it is known*, and the classification determines where in the model it may enter. Chapter 1 made the same distinction when it defined the information set $\mathcal{I}_t$ and ruled out using an observed future temperature in place of the forecast available at issue time. The architecture builds that rule into its wiring.

Table 8.1 lists the three kinds with their thermostat examples. The classification is not an implementation detail: a feature placed in the wrong class is either wasted, if a known future input is treated as unknown, or leakage, if an unknown input is treated as known.

Table 8.1: The three kinds of input a temporal fusion transformer distinguishes, with examples from the thermostat fleet.

Kind	When it is known	Thermostat-fleet examples
Static	Fixed over the horizon; attached to the series, not to time	equipment type, climate zone, building type, customer cohort
Observed past	Known up to the issue time t only; must itself be forecast if needed later	load P_t, indoor temperature T_{in}, duty cycle D_t, realized control u_t
Known future	Known at issue time for the whole horizon	hour and weekday at $t+h$, planned dispatch u_{t+h}, weather forecast $\hat{T}_{t+h\mid t}$

The aggregator's 14:00 forecast makes the three concrete. A forecast is issued at 14:00 for the window 17:00 to 20:00. The static inputs are the thermostat model, the climate region and the customer cohort. The observed history is P, T_{in}, the duty cycle and the realized control over 08:00 to 14:00. The known future inputs are the weather forecast, the calendar variables and the planned dispatch over 15:00 to 20:00. Figure 8.6 lays these out on a timeline: the

horizontal axis is clock time from 08:00 to 20:00, the vertical dashed line is the issue time, and the three bars are the three input kinds, each drawn over the span of time for which it is available. The static bar spans the whole axis, because a climate zone does not change within a day. The observed bar stops at the dashed line, because nothing after it has been measured. The known-future bar starts at the dashed line, because a weather forecast and a dispatch plan for the evening exist at 14:00. The shaded region on the right is the target window.

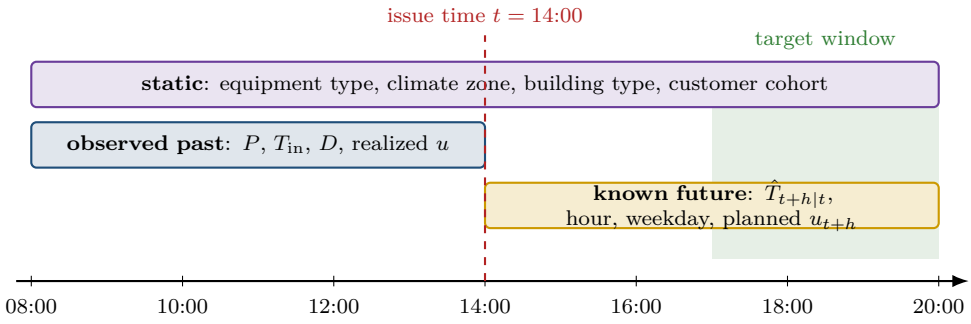


Figure 8.6: The three kinds of input on a timeline, for a forecast issued at 14:00 for 17:00 to 20:00. Each bar spans the time over which that kind of input is available at issue time: static inputs span everything, observed inputs end at the issue time, known-future inputs begin there. The shaded region is the target window. Classification by availability is the rule that prevents the leakage of Chapter 1.

8.4.4 The architecture in outline

The model is assembled in the order the inputs become available. Static inputs are encoded once and used to condition everything downstream. The observed history and the known future inputs are passed through a local temporal layer that captures short-range dynamics. Attention is then applied over the resulting sequence, giving the long-range, context-dependent retrieval of Section 8.2.5. A multi-horizon head produces the whole forecast trajectory. Four capabilities are combined in that chain: local temporal dynamics from the recurrent layer, long-range relationships from attention, known future information from the third input class, and feature selection from the mechanism of the next subsection.

8.4.5 Variable selection alongside attention

The architecture also learns, for each forecast, how much weight to give each *input variable*, not only each timestep. The candidate list for the thermostat fleet is the feature set of Chapter 4: outdoor temperature T, relative humidity RH, load P, hour of day, duty cycle D, the available device count N_{avail}, the humidity forecast and the control history. The weights are context dependent. On a 36 °C afternoon the outdoor temperature dominates; thirty minutes after a curtailment ends the recent duty cycle and the control history dominate, because what limits the response is the thermal buffer the previous event consumed. A single global feature importance, of the kind a boosted ensemble reports, cannot express that: importance varies with the operating condition.

The two selection mechanisms answer different questions and should not be conflated. Variable selection answers *which features matter*, for example that temperature matters far more than humidity for this forecast. Attention answers *which times matter*, for example that 13:00 matters far more than 08:00. Both are recomputed for every forecast, which is what makes the model's own account of itself condition-specific rather than global.

8.4.6 Multi-horizon output

The model emits $\hat{y}_{t+1}, \ldots, \hat{y}_{t+H}$ in one pass. That matches what an operator needs. A dispatch decision for the evening peak is not supported by a single number for 17:00; it needs the trajectory

$$F(16{:}00),\ F(16{:}15),\ F(16{:}30),\ \ldots,\ F(20{:}00), \tag{8.6}$$

because the commitment is over an interval and the resource has to hold for its duration. The whole profile is what constrains the decision, not one point of it.

The horizons in equation (8.6) are not separate problems that happen to be solved together. They are dynamically related, and the relation runs through the physical state. Curtailing the fleet at 17:00 draws down the thermal buffer, which changes the state that determines the flexibility available at 18:00; continued curtailment reduces it further; and when the curtailment is released the recovery load P_{rebound} rises above the baseline while buildings return to setpoint. A model that predicts each horizon independently has no way to respect that coupling, and will report flexibility profiles that are physically impossible, for example a sustained maximum response over four consecutive hours. A multi-horizon model can learn the shape of the whole profile, including the decay and the rebound.

> **Principle 8.1.** *The horizons of a flexibility forecast are coupled through the physical state: what is delivered early changes what remains later. A forecaster that produces horizons independently cannot represent that coupling, and its trajectory may be infeasible even when each point is individually plausible.*

8.4.7 The overlap with control

Once planned dispatch is a known future input and the output is a trajectory, the object being fitted is

$$P_{t+h} = f(\text{state at } t,\ u_{t+1:t+h},\ w_{t+1:t+h}), \tag{8.7}$$

with u the control sequence and w the disturbances. That is not a forecasting equation in the narrow sense; it is a response model, which maps a current state, a future control sequence and future disturbances to a future response. The same expression would be written down by someone building a controller.

The practical reading is that the line between forecasting and system dynamics thins at this point. A model of the form of equation (8.7) can be queried as a forecaster, by fixing u at the planned dispatch, or as a simulator, by sweeping u over candidate dispatches and comparing the resulting trajectories. The second use is what a dispatch optimizer needs, and it is available only because the control sequence was made an explicit input rather than being absorbed into the history.

8.5 Interpretation and model choice

Attention weights are easy to plot, which makes it tempting to read more from them than they contain. This section states what they are evidence of, places the architecture against its alternatives, and gives the hierarchy the three families of model form.

8.5.1 The evidential status of attention weights

Suppose a trained model places high attention on T_{t-24}, yesterday's temperature at the target hour. The correct reading is that the model found that input useful for reducing its loss. It is not evidence that yesterday's temperature caused today's outcome. The distinction matters

most where control actions are involved, because the aggregator's own dispatches are in the history: a model may attend strongly to a past dispatch because dispatches are scheduled on hot days, not because the dispatch itself changed the quantity being forecast. Reading an intervention effect off an attention weight is the same error as reading a treatment effect off a regression coefficient in observational data.

Attention is also an incomplete account of the model's own computation. Several things stand between the weights and the output. There are multiple heads, each with its own weights, and their outputs are concatenated and mixed. There are nonlinear feed-forward layers after the attention layer. There are interactions the weights do not display, and residual paths that carry information around the attention layer entirely. A plot of attention weights is therefore a diagnostic, useful for noticing that a model ignores the known-future weather input or that it attends to a sensor that was offline, and not a derivation of why the model produced the number it produced.

Principle 8.2. *An attention weight is evidence that the model used an input, not that the input caused the outcome, and not a complete explanation of the output. Treat attention maps as diagnostics for debugging, never as an estimate of a causal effect.*

8.5.2 Transformer, recurrent network and boosted trees

Table 8.2 places the three model families of Chapters 5, 7 and this one side by side, with how each represents the history, what it is good at, where it is weak, and the situation that calls for it. The rows are ordered by how much of the temporal representation the model is left to discover.

Table 8.2: Three families of model for a temporal forecasting problem.

Family	History is represented by	Strengths	Weaknesses	Calls for it
Boosted trees	engineered vector X_t of lags, weather, calendar, state	accurate on tabular features, fast, easy to diagnose	cannot use what X_t omits; no notion of sequence	the signal is well summarized by a fixed feature set
LSTM or GRU	a recurrent state, $\mathbf{h}_t = f(\mathbf{h}_{t-1}, x_t)$	natural state and memory reading; works on moderate data; moderate cost	sequential training; long range still hard; history compressed into one vector	moderate sequence length and data; a state reading is wanted
Transformer	direct weighted interaction between any x_i and x_j	long-range interaction; parallel training; flexible input handling	data hungry; quadratic in T; frequently unnecessary	long or context-dependent dependence; many series; multi-horizon output

The comparison against boosted trees is the one that decides most real projects, and it turns on a single question about the data rather than on the models. If the information relevant to

the target is well represented by a vector such as $X_t = [P_{t-1}, P_{t-24}, P_{t-168}, T_t, \text{hour}, D_t]$, then a boosted ensemble will usually be excellent, because the problem has already been reduced to the tabular form trees are best at. If the signal instead lives in the raw window $x_{t-200:t}$ with relationships across time that no short feature list captures, a sequence model becomes interesting and a transformer becomes the sequence model of choice. Model choice follows the structure of the information, not the publication date of the method.

8.5.3 A hierarchy of temporal representations

The three families form a progression in who decides which part of the history matters. With regression or gradient boosting the analyst decides: the feature vector X_t is constructed by hand and the model sees only what it contains. With a recurrent network the model decides, but once: it learns a summary $\mathbf{h}_t$ of the history, and the same summary serves every prediction. With attention the model decides afresh for each prediction, because relevance is recomputed from the current forecasting need.

Figure 8.7 arranges the three as rows. Each row gives the object that carries the history, who constructs it, and whether it varies with the forecast; the arrow on the left is increasing amount of the representation left to be learned, and the annotations on the right are the data requirement that comes with it. The figure is the chapter's organizing claim: engineered summary, then learned state, then dynamic retrieval from history. Each step buys flexibility and costs data.

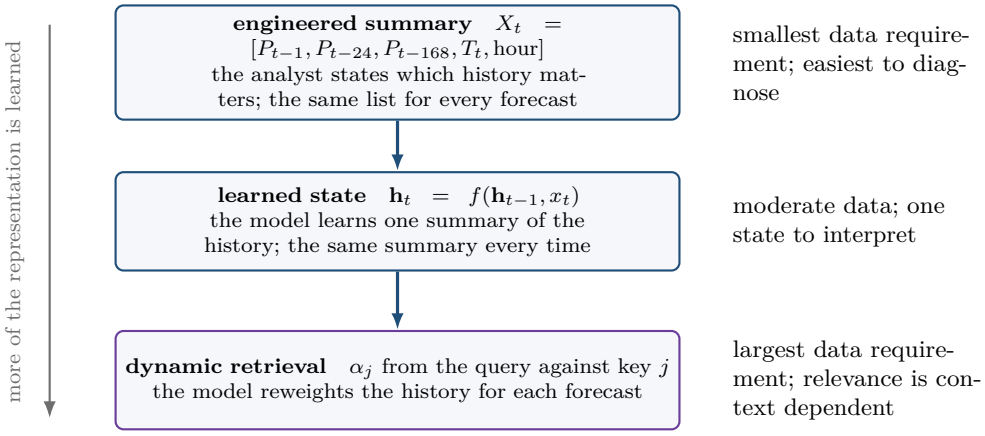


Figure 8.7: The hierarchy of Section 8.5.3. Each row names the object that carries the history, who constructs it and whether it varies between forecasts. Moving down the rows transfers more of the representation from the analyst to the model and raises the amount of data required.

8.5.4 Similar historical events and the feature-engineering alternative

A concrete case shows what dynamic retrieval buys. The fleet faces an unusual 39 °C day. The most informative history for this afternoon's response is probably not P_{t-24}, yesterday's mild afternoon, nor P_{t-168}, last week's mild afternoon. It is the previous extremely hot day, which may be $t - 912$ hours back: a point in the history that no fixed lag set contains and that a recurrent state has long since overwritten. Attention can in principle learn that an observation 38 days old is unusually relevant today, because relevance is computed by matching keys against a query, and the query encodes *39°C weekday afternoon*.

The same effect can be obtained without attention, and often should be. Construct the retrieval explicitly as a feature: search the history for the five days most similar to today in outdoor temperature, humidity, day type and fleet composition, and supply their observed

responses, or summary statistics of them, as columns of X_t. A boosted ensemble then has the similar-day information available, and the similarity metric is one an engineer can inspect, argue about and correct. This is the similar-day baseline of Chapter 3 promoted from a benchmark to a feature. Read this way, attention automates one class of representation learning that careful feature engineering can sometimes accomplish directly, and the choice between them is a question of which is cheaper and more reliable on the data at hand.

8.6 Data requirements and the fleet case

Whether these architectures can be used at all is a question about data, and whether they should be is a question a reviewer is likely to ask. This section covers both, and assembles the chapter's pieces into one fleet architecture along the way.

8.6.1 Data requirements and global models

Complex sequence models need large training sets, and the energy sector has a particular form of data abundance that can mislead. Ordinary consumption data is enormous: a fleet of 100 000 thermostats reporting every fifteen minutes generates roughly 3.5×10^9 observations a year. The events a flexibility forecast must get right are not ordinary. Extreme heat waves occur a few times per summer, long-duration dispatch events a handful of times per season, grid emergencies rarely. Billions of telemetry observations can therefore coexist with a few dozen examples of the scenario the model is being built for, and a model fitted on the billions will be fitted almost entirely to ordinary operation.

Global models of the kind described in Section 8.4.2 strengthen the data argument without dissolving that caveat. One model $f_\theta(x_{i,t}, s_i)$, fitted across all devices with a device-specific static input s_i to absorb their differences, learns from millions of sequences instead of one, and that is the regime in which deep sequence models become plausible at all. Chapter 11 treats the construction. The rare-event problem survives it, because pooling across devices multiplies the number of sequences but not the number of distinct extreme days: every device in the fleet experiences the same heat wave.

8.6.2 An architecture for fleet flexibility forecasting

Figure 8.8 assembles the pieces into the system this chapter has been building toward. Historical fleet telemetry, namely load, indoor temperature, duty cycle and control history, enters a sequence encoder on the left. Static device information enters from below. The attention layer in the middle receives the encoded history together with the known future inputs drawn above it: the weather forecast, the future calendar and the planned dispatch. The multi-horizon head on the right produces a future response trajectory. The output is labeled with three quantiles rather than one number, the median P_{50} with the P_{10} and P_{90} bounds around it, because the honest form of the answer is a distribution,

$$p\left(F_{t+1:t+H} \mid \mathcal{I}_t,\ u_{t+1:t+H}\right), \tag{8.8}$$

not a point trajectory $\hat{F}$. Equation (8.8) is sequence modeling and probabilistic forecasting combined: the sequence model supplies the conditional mean structure over the whole horizon, and the machinery of Chapter 10 supplies the spread. Both conditionings in it are load bearing. The information set $\mathcal{I}_t$ is the leakage constraint of Chapter 1; the control sequence $u_{t+1:t+H}$ is the dependence on the operator's own plan of equation (8.7).

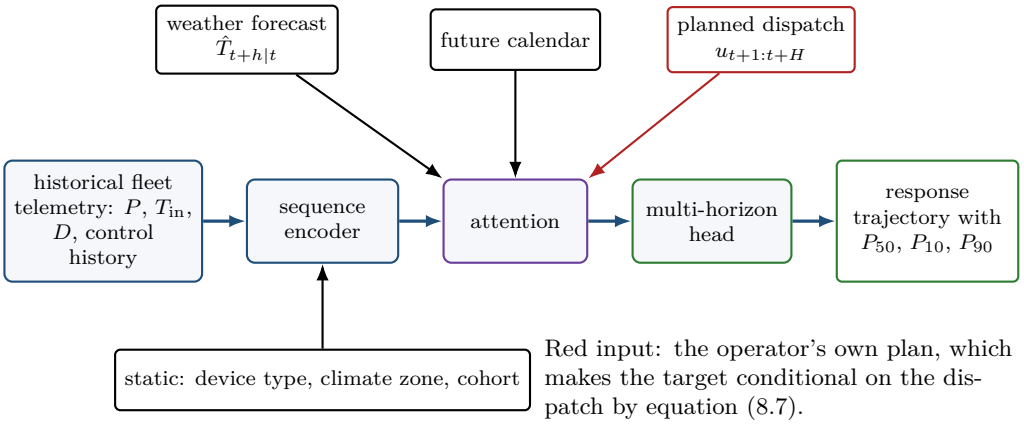


Figure 8.8: A fleet flexibility forecaster built on attention. The left-to-right path is the forecast computation; the boxes above the attention layer are the known future inputs of Table 8.1; the box below the encoder is the static input. The output is the predictive distribution of equation (8.8) rather than a point trajectory.

8.6.3 Three conceptual questions and the deeper idea

A candidate asked about forecasting architecture should not open with transformers. The defensible order is the order of this book: define the target and the issue time, establish a baseline, construct features from the information set, fit a strong structured-data model, inspect the residuals for the temporal structure the model missed, and add sequence modeling only if that inspection justifies it. The sentence that belongs at the end of that sequence is that if longer histories or multi-horizon trajectories turn out to contain significant additional signal, an LSTM or an attention model such as a temporal fusion transformer is worth testing. Leading with the architecture inverts the reasoning and invites the question of what evidence supports it.

Conceptual questions

"Why would attention help in energy forecasting?"
Because the relevance of a historical observation depends on the forecast context rather than only on how old it is. For a hot weekday afternoon the most informative history may be another extreme period weeks back rather than the immediately preceding hours, and for a forecast issued half an hour after a curtailment ends it may be the dispatch record. Attention weights the history dynamically for each forecast instead of forcing it through a fixed recurrent state or a fixed lag list. A strong answer adds that the mechanism also connects to P_{t-24} and P_{t-168} directly, which is where most of an energy series' autocorrelation sits.

Conceptual questions

"Why not use a transformer for everything?"
Because the information structure often does not call for it. If engineered lag, weather, calendar and state features already summarize what the target depends on, a tree ensemble is likely to be as accurate, easier to train, easier to diagnose and more robust with limited data. The case for a transformer has to be made on validation evidence that longer or context-dependent sequence relationships carry signal the feature set does not, and that case is most likely to succeed at fleet scale or on multi-horizon trajectories. Quadratic cost in sequence length and a large appetite for data are the prices paid for the flexibility.

Conceptual questions

"What is the difference between a transformer and an LSTM?"
An LSTM propagates information forward through a learned recurrent state and reads the final state; a transformer relates positions directly through attention weights. The consequences are that long-range dependence is easier to represent in a transformer, because the path between any two positions has length one, and that training parallelizes, because there is no sequential state to compute. The costs are quadratic scaling in sequence length and a larger data requirement. An LSTM remains an effective and simpler choice for moderate sequence lengths and moderate datasets, and for the aggregator's purposes it also offers a state that can be compared against a physical one.

The fleet setting lines up with these architectures on every axis. The histories are long and layered, combining recent hours, daily and weekly cycles, and a record of irregular events. The devices are heterogeneous, which is what static inputs are for. Substantial future information is genuinely known, namely the weather forecast and the dispatch plan. The output needed is a multi-horizon trajectory. And the datasets are very large. None of that proves a transformer will beat a well-built gradient boosted ensemble on the aggregator's data. The question is empirical, it is settled by the validation protocol of Chapter 9, and the honest prior is that the simpler model is hard to beat.

> **Principle 8.3.** *The progression of this book is a progression in who decides what matters. Feature engineering asserts which history matters; a recurrent network learns one summary of it; attention makes relevance itself learned and context dependent, recomputed for each forecast.*

8.7 What to retain

Definition 8.1 (Attention)**.** Attention computes, for each prediction, a set of nonnegative weights summing to one over the available information, by comparing a query against one key per item, and returns the weighted average of the items' values,

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}.$$

A transformer builds a sequence model on that mechanism instead of on recurrence, and a temporal fusion transformer adapts it to forecasting by separating static inputs, observed history and known future inputs and by emitting a whole horizon at once. Attention is chosen when the temporal structure of the problem justifies it, not because it is the most recent method available.

Table 8.3 collects the chapter's objects with the subsection that defines each.

8.8 Exercises

Exercise 8.1. A query produces the relevance scores $s = [0.5,\ 1.5,\ 3.0]$ for three historical observations. Compute the attention weights from equation (8.4). Now suppose the $\sqrt{d_k}$ scaling of Section 8.2.4 had been omitted with $d_k = 64$, so that the raw scores are eight times as large: recompute the weights and state how much of the total weight the third observation then carries. Explain, in terms of the gradient of the softmax, why the second case is a problem during training.

Table 8.3: The objects of this chapter.

Object	Meaning	Section
$\mathbf{h}_T$	the recurrent bottleneck: the only path from x_1 to the forecast	8.1.1
α_j	attention weight on item j; nonnegative, summing to one	8.1.2
$\mathbf{Q}, \mathbf{K}, \mathbf{V}$	query, key and value matrices: the query matches keys, the values are retrieved	8.2.1
attention layer	scaled dot-product attention, $\text{softmax}(\mathbf{Q}\mathbf{K}^\mathsf{T}/\sqrt{d_k})\,\mathbf{V}$	8.2.2
$\sqrt{d_k}$	variance-stabilizing divisor; keeps the softmax from saturating	8.2.4
self-attention	$\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ from one sequence; unit-length path between any two positions	8.2.5
positional encoding	the input that gives attention a notion of order and calendar time	8.2.6
encoder block	self-attention, feed-forward, residuals and normalization, stacked L times	8.3.1
multi-head attention	several attention mechanisms in parallel; roles are discovered, not assigned	8.3.3
T^2 cost	size of the attention matrix; the limit on window length	8.3.4
static, observed, known future	the three input kinds, classified by when they are known	8.4.3
variable selection	context-dependent weights over features, as against over times	8.4.5
$\hat{y}_{t+1}, \ldots, \hat{y}_{t+H}$	multi-horizon output; horizons coupled through the state	8.4.6
$p(F_{t+1:t+H} \mid \mathcal{I}_t, u)$	the target object: a predictive distribution over a trajectory	8.6.2

Exercise 8.2. Take the seven weights of Figure 8.2 and suppose the corresponding fleet loads are [21, 48, 76, 39, 71, 74, 72] MW in the order shown. Compute the attention-weighted retrieval of equation (8.2). Compare it with the unweighted mean of the seven values, and with seasonal persistence, which uses 17:00 yesterday alone. Which of the three is closest to the last observed value, and what does that tell you about where the mechanism places its mass?

Exercise 8.3. Count the entries of the attention matrix of Section 8.3.4 for a window of one week of hourly data, one year of hourly data, and one year of five-minute data. For each, compare the count with the T operations a recurrent network would use. At what window length does the attention matrix first exceed 10^8 entries, and what does that imply for a model intended to read five-minute telemetry over a whole cooling season?

Exercise 8.4. Classify each of the following into the three input kinds of Table 8.1, for a forecast issued at 14:00 for 17:00 to 20:00: the building's floor area; the outdoor temperature measured at 13:00; the outdoor temperature that will be measured at 18:00; the day-ahead weather forecast for 18:00 issued this morning; the number of available devices N_{avail} at 18:00; the dispatch the operator has scheduled for 17:30. For each item you place in the observed-past class, say what would go wrong if it were wired into the model as a known future input instead.

Chapter 9

Forecast validation: time-ordered backtesting, metrics and regimes

9.1 Time order and the validation split

Chapters 5 to 8 built models of increasing capacity: regularized regression, gradient-boosted trees, recurrent networks, attention. Each of them can be made to fit a historical load record closely. None of that establishes that any of them would have been useful to an operator, and the gap between the two statements is the subject of this chapter. Its question is

How do we know that a forecast is genuinely better, and not merely better on the data we happened to test it on?

A model can look excellent and be operationally useless. The usual cause is not a bug in the model but a validation procedure that gave the model information it will never have in production, or that summarized its performance in a number nobody acts on. For time series the procedure differs from the random train/test split used on cross-sectional data, and the difference is not a technicality: it changes the reported error by factors, not percentages.

Jargon: backtest, out-of-sample, rolling origin, expanding and rolling window, point-in-time data, leakage

A *backtest* re-runs a forecasting system over history as if it were operating then, producing at each historical issue time the forecast it would have produced. A forecast is *out-of-sample* when the target was not used in fitting; it is *out-of-time* when the target lies after every observation used in fitting, which is the stronger requirement this chapter imposes. *Rolling-origin* evaluation repeats the exercise with the origin (the issue time) moved forward, giving many out-of-time evaluations instead of one. The origin can be advanced with an *expanding window*, in which the training set keeps all history, or a *rolling window*, in which it keeps only the most recent fixed span. *Point-in-time* data is data stored with the time at which each value *became available*, not only the time it refers to. *Leakage* is the use, in training or evaluation, of anything that would not have existed at issue time.

Key idea

Validation must recreate the actual information flow: past, then issue time, then an unseen future. A metric computed in any other order measures a task the system will never be asked to perform.

This section establishes why a random split is invalid here, builds the time-ordered alternative, distinguishes the expanding from the rolling window, and states the reconstruction requirement that makes a backtest honest.

9.1.1 Why a random row split fails

Take the aggregator's record: hourly fleet load and weather from 2019 through 2024, about 52 000 rows. The reflex borrowed from cross-sectional machine learning is to shuffle the rows and hold out 20 % of them. The result is a model trained on rows from 2024 and scored on rows from 2021. In operation that never happens: at the moment a forecast is issued, every observation after it is absent, and no amount of care in the fitting procedure recovers it.

Two mechanisms turn that ordering violation into an optimistic number. The first is direct: a model that has seen 2024 has seen the fleet's 2024 composition, the 2024 firmware, the 2024 enrollment and the 2024 weather regime, and 2021 is then partly an interpolation rather than an extrapolation. The second is indirect and usually larger. The features of Chapter 4 are built from lags and rolling statistics, so a row at 14:00 and a row at 15:00 share most of their inputs; splitting at random puts near-duplicates of the test rows into the training set, and the model is scored on neighbors of what it memorized. Hourly energy series are strongly autocorrelated, which is exactly what makes this failure severe here and mild in a problem with independent rows.

Figure 9.1 contrasts the two arrangements. Each row of cells is the 2019–2024 record divided into calendar quarters, read left to right; blue cells are used for fitting, red cells for evaluation, and gray cells are held back for later. The upper row is a random 80/20 split, in which red cells are scattered among blue ones; the lower row is a time-ordered split, in which every red cell lies after every blue one. Only the lower arrangement answers a question an operator can act on.

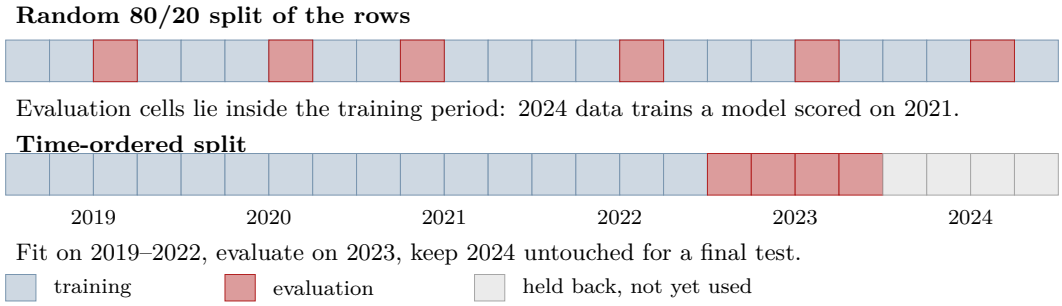


Figure 9.1: Two ways of dividing the same 2019–2024 hourly record, drawn as 24 calendar quarters. In the upper arrangement the evaluation quarters are drawn at random and sit among the training quarters; in the lower arrangement they follow them. Only the lower arrangement reproduces the order in which information arrives in operation. The positions of the red cells in the upper row are illustrative.

9.1.2 A single time-ordered split, and why one is not enough

The simplest valid procedure trains on 2019–2022 and tests on 2023. The resulting error answers a well-posed counterfactual: if this model had been fitted at the end of 2022 and run through 2023, how would it have performed? That question is the one an operator asks, and the number is interpretable without further qualification.

One such split is nevertheless weak evidence, because it confounds the model with the test year. The year 2023 may have been unusually mild, with few of the hot afternoons during which the forecast matters; or it may have contained an extreme heat event that no model could have anticipated. In either case the single number is dominated by the accident of which twelve months were held out.

The remedy is to repeat the exercise with the split point moved forward, producing several

out-of-time evaluations:

$$\begin{aligned} &\text{2019–2021} \implies \text{test 2022},\\ &\text{2019–2022} \implies \text{test 2023},\\ &\text{2019–2023} \implies \text{test 2024}, \end{aligned} \tag{9.1}$$

where in each line the test period is genuinely future relative to the training period that produced the forecast. This is rolling-origin backtesting, and it is the basic unit of evidence in this chapter. Three test years also permit a statement about consistency, which Section 9.5.4 shows is often more informative than the average itself.

9.1.3 Expanding and rolling windows

Equation (9.1) advances the origin while keeping all history. Writing the training sets as

$$D_1 = [\,\text{2019–2021}\,], \qquad D_2 = [\,\text{2019–2022}\,], \qquad D_3 = [\,\text{2019–2023}\,], \tag{9.2}$$

each set contains the one before it. This is the expanding window, and it is the right choice when old data remains relevant: the physical relationship between outdoor temperature and air-conditioning load does not change from year to year, so a 2019 heat wave still carries information about a 2024 one.

The alternative keeps the window length fixed and discards the oldest data as the origin advances,

$$\begin{aligned} &\text{2020–2021} \implies \text{test 2022},\\ &\text{2021–2022} \implies \text{test 2023},\\ &\text{2022–2023} \implies \text{test 2024}. \end{aligned} \tag{9.3}$$

Discarding information looks wasteful and is often correct, because the system being forecast is not stationary. Between 2019 and 2024 the fleet changed composition as thermostats, electric vehicles and batteries enrolled in different proportions; device firmware was updated and altered how a setpoint change is executed; the customer population turned over; and the aggregator's own control strategy changed, so the historical relationship between a dispatch command and the delivered response is not the current one. Under that kind of concept drift a two-year window describes the present fleet better than a six-year window describes it on average.

Figure 9.2 draws both schemes on the same time axis. Each horizontal bar is one fold: the blue part is the training span, the red part is the evaluation span, and the horizontal axis is the calendar year. The upper group is the expanding window of equation (9.2), whose blue bars all start at 2019 and grow; the lower group is the rolling window of equation (9.3), whose blue bars have constant length and slide. The two groups share their evaluation periods, which is what makes their errors comparable, and the choice between them is an empirical question: fit both and compare on the same test years.

9.1.4 Reproducing the issue-time information state

Splitting on time is necessary and not sufficient. Chapter 1 defined the information set $\mathcal{I}_t$ as everything genuinely known at issue time t; a backtest is honest only if every historical example is built from $\mathcal{I}_t$ and nothing else. Consider the operational task of forecasting the fleet at 17:00 from a 14:00 issue time, a three-hour horizon. Each historical example for that task must be reconstructed as it stood at 14:00 on its own day, which means five specific substitutions.

- The temperature input must be the weather forecast that existed at 14:00, $\hat{T}_{17:00|14:00}$, not the temperature $T_{17:00}$ that was later observed.
- Device participation must be the enrollment and availability roster as of 14:00, not the final roster known after the event.

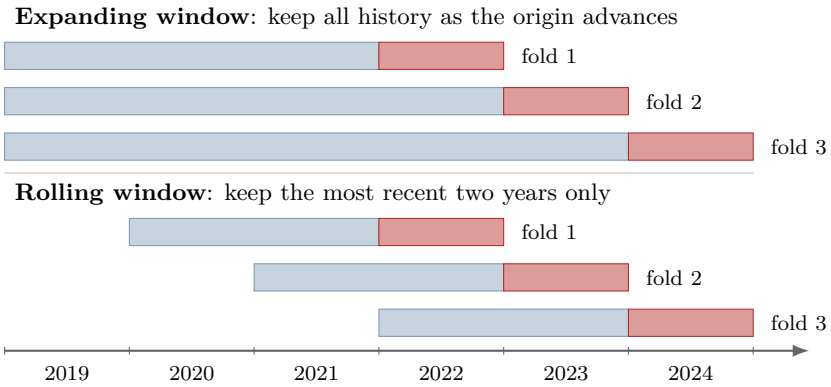


Figure 9.2: Expanding and rolling windows as a timeline. Blue is the training span of a fold, red its evaluation span, and the axis is the calendar year. The expanding folds are equation (9.2); the rolling folds are equation (9.3). Both groups evaluate on 2022, 2023 and 2024, so their errors are comparable fold by fold.

- The event duration must be the duration planned at 14:00, not the duration the event actually ran if it was extended or cut short.
- No telemetry recorded after 14:00 may enter any feature, including the post-event measurements used for verification.
- Measurements must be the values as they stood at 14:00, not values later corrected by a data-quality pipeline.

Each violation makes the reported error smaller than the achievable error, and the size of the effect is substantial: Section 9.5.1 gives a case in which replacing a forecast temperature by the observed one moves the root-mean-square error from 7 to 4 MW. The requirement is demanding in practice because it forces the historical store to be point-in-time, keeping for each value both the time it refers to and the time it arrived.

Principle 9.1. *A backtest is a simulation of the production system against history. Every input to a historical forecast must be the version of that input which existed at the issue time, including weather forecasts, rosters, plans and uncorrected measurements.*

9.1.5 Training error is almost irrelevant

One consequence deserves to be stated on its own, because it is the most common misreading of a model report. Suppose model A attains RMSE = 1.5 MW on its training data and model B attains 4.0 MW. That ranking carries no information about which model to deploy. If on the following year model A attains 10 MW and model B attains 5 MW, model A was fitting noise: a flexible learner with enough capacity can drive training error toward zero by memorizing the idiosyncrasies of the sample, and the gradient-boosted ensembles and recurrent networks of Chapters 5 and 7 both have that capacity.

The quantity of interest is out-of-sample performance on genuinely future data, and the only reliable estimate of it comes from the procedure of Sections 9.1.2 to 9.1.4. Training error retains one narrow diagnostic use: a model whose training error is also large has not learned the structure that exists, which points at features or model class rather than at overfitting. In every other respect the training number should be ignored.

9.2 Error metrics

Once a valid set of out-of-time forecasts exists, it must be reduced to numbers. This section defines the four metrics that carry most of the operational meaning, works each on the same small example, and states the circumstances under which each is the right summary.

9.2.1 Mean absolute error

The mean absolute error averages the magnitude of the error without regard to sign,

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|, \tag{9.4}$$

where y_i is the realized value and $\hat{y}_i$ the forecast issued for it. Take four forecasts whose errors $e_i = y_i - \hat{y}_i$ are $[-2, +5, -3, +4]$ MW. The absolute values are $[2, 5, 3, 4]$, their sum is 14, and

$$\mathrm{MAE} = \tfrac{14}{4} = 3.5\,\mathrm{MW}. \tag{9.5}$$

The statement to make from that number is that the forecast misses by about 3.5 MW on a typical occasion. The metric has the units of the target, which is why it communicates well to people who operate the system: 3.5 MW on a fleet whose response is tens of megawatts is immediately interpretable, where a dimensionless score is not.

9.2.2 Root-mean-square error

The root-mean-square error averages the *squared* error and then takes the square root to restore the units,

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}. \tag{9.6}$$

On the same errors the squares are $[4, 25, 9, 16]$, their mean is 13.5, and $\mathrm{RMSE} = \sqrt{13.5} = 3.67$ MW, slightly above the 3.5 MW of equation (9.5). The squaring is the whole of the difference between the two metrics, and its effect is clearest in a constructed pair of models.

Let model A produce errors of $[4, 4, 4, 4]$ MW and model B produce errors of $[0, 0, 0, 16]$ MW. Both have $\mathrm{MAE} = 4$ MW: by equation (9.4) the two error patterns are indistinguishable. Their root-mean-square errors are not. For model A the squares are all 16, so $\mathrm{RMSE} = 4$ MW; for model B the squares are $[0, 0, 0, 256]$ with mean 64, so

$$\mathrm{RMSE}_{\mathrm{B}} = \sqrt{\tfrac{256}{4}} = 8\,\mathrm{MW}, \tag{9.7}$$

twice model A's. Table 9.1 collects all of these figures: the columns are the three error vectors of this section and the rows are the metrics, with the ratio RMSE/MAE included because it is a direct read-out of how concentrated the errors are. The ratio is exactly 1 when every error has the same magnitude and grows as the distribution develops a tail; model B's value of 2 says that its error budget is spent almost entirely on one occasion.

9.2.3 Choosing between the two

Neither metric is universally correct, and the choice is a statement about consequences rather than about statistics. The mean absolute error reports the typical error magnitude and treats a single 16 MW miss as equivalent to four 4 MW misses. The root-mean-square error declares that it is not equivalent, and the operational question is whether that declaration is true for the decision at hand.

Table 9.1: The metrics of Section 9.2 on three error vectors, in MW. The running example is $e = [-2, +5, -3, +4]$; the other two columns are the equal-MAE pair of Section 9.2.2, for which only the magnitude metrics are meaningful, so the bias row is left blank.

Metric	Definition	Running example	A $[4, 4, 4, 4]$	B $[0, 0, 0, 16]$
MAE	$\frac{1}{N}\sum \lvert e_i \rvert$	3.5	4.0	4.0
RMSE	$\sqrt{\frac{1}{N}\sum e_i^2}$	3.67	4.0	8.0
RMSE/MAE	concentration of the errors; 1 when all are equal	1.05	1.00	2.00
Bias	$\frac{1}{N}\sum(\hat{y}_i - y_i)$, equal to $-\bar{e}$	-1.0	–	–

For grid operations it usually is. Four forecasts that are each 4 MW optimistic produce four small procurement adjustments; one forecast that is 16 MW optimistic can produce a commitment the fleet cannot meet, a reliability event for the utility and a penalty. The consequence is convex in the error, so a convex penalty is the honest summary and root-mean-square error is the appropriate headline. Where the consequence is close to linear in the error, for instance imbalance settled at a fixed price per megawatt-hour, mean absolute error is the better match. The metric should reflect the decision consequences, not convention.

9.2.4 Bias

Bias is the mean *signed* error, with the sign taken so that a positive value means overprediction,

$$\text{Bias} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i) = -\bar{e}. \tag{9.8}$$

On the running example $\bar{e} = (-2 + 5 - 3 + 4)/4 = +1.0$ MW, so Bias $= -1.0$ MW: the forecast underpredicts by one megawatt on average, a small systematic component inside a 3.5 MW typical error. A bias of $+5$ MW would mean the opposite, that the forecast overstates the target by five megawatts on average.

The distinction matters because bias and magnitude have different operational meanings, and a tolerable magnitude can conceal an intolerable bias. Suppose the aggregator's flexibility forecast is consistently 5 MW high. The dispatch optimizer then overcommits on every occasion, in the same direction, and the shortfalls accumulate into a pattern of underdelivery rather than cancelling out. Random errors of the same size are absorbed by reserves; a persistent offset is not, because reserves are sized for a zero-mean disturbance. A forecast report that gives MAE without Bias is therefore incomplete, and the two should always be read together.

9.2.5 Percentage and normalized errors

Two scaled metrics are in common use and both require care. The mean absolute percentage error expresses each error as a fraction of the realized value,

$$\text{MAPE} = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \quad [\%], \tag{9.9}$$

which is intuitive in a report and unstable in exactly the regimes this book cares about. When y_i approaches zero the summand diverges. That happens routinely: the response of a demand-response fleet is near zero outside events, and photovoltaic output is near zero at dawn and

dusk, so a handful of timestamps with $y_i \approx 0$ can dominate the average and make the metric say more about those timestamps than about the forecast. Percentage error is usable on a quantity bounded away from zero, such as total system load, and should be avoided on flexibility and on solar output.

The safer scaling divides by a constant rather than by the realized value,

$$\mathrm{NRMSE} = \frac{\mathrm{RMSE}}{\bar{y}} \qquad \text{or} \qquad \mathrm{NRMSE} = \frac{\mathrm{RMSE}}{C_{\text{fleet}}}, \tag{9.10}$$

with $\bar{y}$ the mean of the target over the evaluation period and C_{fleet} the fleet's installed capacity. The purpose is comparability: $\mathrm{RMSE} = 5\,\mathrm{MW}$ is an excellent forecast of a 1000 MW fleet and a poor forecast of a 50 MW one, at 0.5 % and 10 % of capacity respectively. Dividing by a constant keeps the ranking of models unchanged while making the number portable across fleets, which is what a percentage is wanted for.

9.3 Slicing by regime and horizon

A single number averaged over a test year hides the behavior that determines whether the forecast is useful. This section shows how aggregate metrics conceal regime failures, lists the slices worth computing, treats the horizon as the most important of them, and introduces skill relative to a baseline.

9.3.1 Aggregate metrics hide regime failures

Consider two models evaluated over a full year: $\mathrm{RMSE}_{\mathrm{A}} = 4.2\,\mathrm{MW}$ and $\mathrm{RMSE}_{\mathrm{B}} = 4.5\,\mathrm{MW}$. On the annual number, model A wins. Now restrict the evaluation to the afternoons of extreme heat, when the system peaks and the aggregator's response is called:

$$\mathrm{RMSE}_{\mathrm{A}}^{\text{heat}} = 15\,\mathrm{MW}, \qquad \mathrm{RMSE}_{\mathrm{B}}^{\text{heat}} = 7\,\mathrm{MW}. \tag{9.11}$$

Model A is better on average and worse by a factor of two in the condition that creates nearly all of the value. The annual metric is dominated by the thousands of mild hours during which any reasonable model is adequate and no decision of consequence is taken.

The resolution is not to invent a cleverer single number but to state the operating regime with every metric. A forecast exists to support specific decisions in specific conditions, and performance must be reported in those conditions.

> **Principle 9.2.** *Report metrics by operating regime. An average over all hours measures mostly the hours in which the forecast does not matter.*

9.3.2 The slices that matter

Which slices to compute follows from the physics and the operations, not from the data. Table 9.2 lists the seven used throughout this book: the first column names the slicing variable, the second gives the levels, and the third states what a degradation within that slice would indicate. Three features of the table are worth noting before it is read. The slices are not independent, so a model that fails on extreme heat will usually also fail on the late-afternoon hour slice; the dispatch-status slice is the one that separates baseline load forecasting from flexibility forecasting, since the counterfactual baseline of Chapter 1 is only tested during events; and the prior-control slice exists because the fleet has memory, so the same weather yields a different response from a fresh fleet than from one recovering from a dispatch an hour earlier.

Table 9.2: Slices of the evaluation set, and what a failure within each one indicates.

Slice	Levels	What a degradation indicates
Temperature	$T < 25$°C; 25–35°C; $T \geq 35$°C	missing saturation, sparse training data at the extreme, growing weather uncertainty
Hour of day	morning ramp; afternoon peak; evening; overnight	wrong calendar or solar features, mis-specified daily shape
Day type	weekday; weekend; holiday	calendar features too coarse
Fleet size	by number of enrolled or available devices	aggregation effects, small-fleet variance not represented
Dispatch status	normal operation; active event; post-event rebound	the controlled response, not the baseline, is mis-modeled
Horizon	15 min; 1 h; 3 h; 12 h; 24 h	feature timing, weather-forecast quality, error growth
Prior control	fresh fleet; recently dispatched	device state and recovery dynamics absent from the features

9.3.3 Error growth with the horizon

Of those slices the horizon is the one that changes the number most. Table 9.2 lists five horizons; evaluating the same model at each of them gives

$$\mathrm{MAE}_h: \quad 1.5,\ 3.0,\ 6.0,\ 10.5,\ 14.0\,\mathrm{MW} \quad \text{at } h = 15\,\mathrm{min},\ 1\,\mathrm{h},\ 3\,\mathrm{h},\ 12\,\mathrm{h},\ 24\,\mathrm{h}. \tag{9.12}$$

Figure 9.3 plots those five values: the horizontal axis is the horizon in hours, the vertical axis is the mean absolute error in megawatts, and each marked point is one entry of equation (9.12). The curve rises steeply over the first three hours and then flattens, which is the signature of two different information regimes. At fifteen minutes the current state of the fleet almost determines the answer, so the error is small. By twenty-four hours the initial state has been forgotten and the error is set by the quality of the weather forecast and of the behavior model, which improve only slowly with effort.

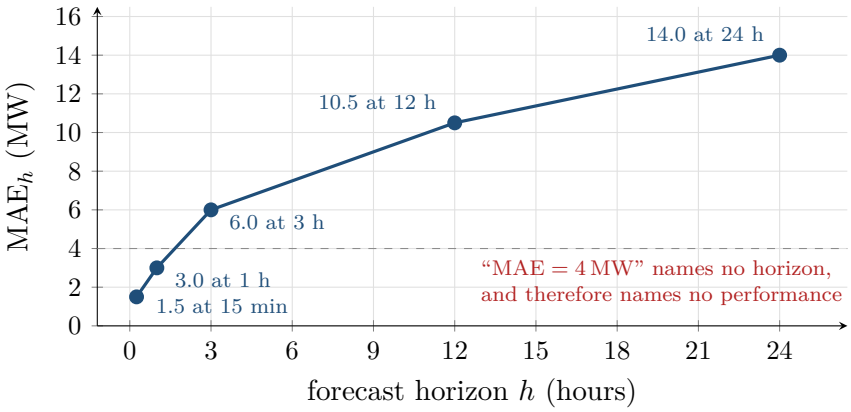


Figure 9.3: Mean absolute error against horizon, from the five values of equation (9.12). The dashed line at 4 MW is a headline figure quoted without a horizon; it coincides with the curve at roughly 1.5 hours and is wrong by a factor of 3.5 at 24 hours. Values are illustrative.

Two practices follow. First, every reported metric carries its horizon: the claim "the model has MAE = 4 MW" is incomplete, and the correct response to it is to ask at what horizon.

Second, horizons should not be averaged together without a reason. A model-predictive dispatch loop that re-optimizes every fifteen minutes depends far more on $h = 1$ to 4 than on $h = 20$ to 24, because the later horizons will be revised several times before they are acted on. Computing MAE_h separately for each h and reporting the error-growth curve of Figure 9.3 gives the operator the quantity that is actually used; a single average over horizons is a weighted combination with weights nobody chose.

9.3.4 Skill relative to a baseline

An absolute error is uninterpretable without a reference, because its size depends on how hard the series is. Chapter 3 supplied the references: persistence, seasonal persistence, similar-day averages and weather regression. The skill score expresses the error reduction relative to one of them,

$$\mathrm{Skill} = 1 - \frac{\mathrm{RMSE}_{\mathrm{model}}}{\mathrm{RMSE}_{\mathrm{base}}}. \tag{9.13}$$

Suppose the seasonal baseline attains $\mathrm{RMSE} = 10\,\mathrm{MW}$ on the test period and a gradient-boosted model attains 7 MW. Then $\mathrm{Skill} = 1 - 7/10 = 0.30$: the model removes 30 % of the baseline's error. That is a defensible headline, because the comparison is against a forecast anyone could have produced.

Now add a recurrent network with $\mathrm{RMSE} = 6.8\,\mathrm{MW}$. Against the same baseline its skill is $1 - 6.8/10 = 0.32$, and the natural sentence is that the network improves on the baseline by 32 %. Against the model it would actually replace, the improvement is

$$\frac{7.0 - 6.8}{7.0} = 0.029, \tag{9.14}$$

that is 2.9 %. Both numbers are arithmetically correct and they support opposite decisions. The 32 % figure compares the network with something nobody would deploy; the 2.9 % figure compares it with the incumbent, and the incumbent is the only relevant reference for a replacement decision. Skill must always name its baseline, and the baseline that matters is the system currently in production.

9.3.5 Statistical and operational significance

Whether 2.9 % justifies deployment is not answered by the 2.9 %. The questions that decide it are: whether the improvement is consistent across backtest periods or confined to one lucky year; whether it appears in the regimes of Table 9.2 that carry the value, particularly extreme heat and active events; how much operational complexity the new model adds, in retraining, monitoring and inference cost; and whether the system is harder to operate and debug when it misbehaves at three in the morning. A small average improvement bought with a much more complex production system is often a net loss.

Statistical significance is a separate matter and is easily confused with importance. An hourly record over several years and a large fleet can supply tens of thousands of evaluation points, enough that $\mathrm{RMSE}_{\mathrm{A}} = 5.02\,\mathrm{MW}$ and $\mathrm{RMSE}_{\mathrm{B}} = 5.00\,\mathrm{MW}$ differ significantly by any standard test. No operator changes any decision over 0.02 MW. The converse also occurs: an improvement too small to reach significance over a full year can be worth a great deal if it is concentrated in the few hours of a system emergency, when the cost of a megawatt of error is orders of magnitude above its average. Significance is a statement about sampling noise; importance is a statement about consequences, and only the second decides deployment.

9.4 Event-level and decision-aware validation

Timestep metrics evaluate the forecast as a sequence of independent numbers, which is not how it is used. This section defines the event-level quantities an operator commits against, shows a forecast that passes the timestep test and fails the event test, and replaces the symmetric metric with the asymmetric cost the decision actually incurs.

9.4.1 Event-level metrics

A demand-response event is a block of time, say 16:00 to 20:00, during which the fleet is asked to hold a reduction. What the utility buys and the aggregator is paid for is a property of the whole block, not of its timesteps. Five quantities describe it, with F_t the delivered reduction of Chapter 1:

$$\begin{aligned} \text{peak reduction} &= \max_{t \in \text{event}} F_t, & \text{average reduction} &= \bar{F}_{\text{event}}, \\ \text{curtailed energy} &= \sum_{t \in \text{event}} F_t \, \Delta t, & \text{minimum sustained} &= \min_{t \in \text{event}} F_t, \end{aligned} \tag{9.15}$$

together with the rebound, the excess load drawn after the event as the buildings recover. Each supports a different contract term: capacity products pay for the minimum sustained level, energy products for the curtailed energy, and the rebound determines whether the event created a second peak an hour later.

A forecast is therefore validated twice, once on the timesteps and once on the event aggregates of equation (9.15). The two tests are not equivalent, and a model can pass the first while failing the second.

9.4.2 A forecast that passes on timesteps and fails on the event

Take a four-hour event with a 50 MW target. The forecast is a flat $[55, 55, 55, 55]$ MW and the delivered response is $[55, 50, 42, 35]$ MW. The timestep errors are $[0, 5, 13, 20]$ MW, so MAE = 9.5 MW against an average response of about 45 MW, roughly a 21 % typical error: unimpressive but not alarming in a summary table.

Figure 9.4 plots the two trajectories. The horizontal axis is the hour of the event, the vertical axis is the fleet's load reduction in megawatts, the blue step line is the forecast, the red step line is what was delivered, and the dashed horizontal line is the 50 MW target. The forecast is flat; the delivery decays. Read through equation (9.15) the event-level picture is different from the timestep picture: the forecast implies a minimum sustained reduction of 55 MW and the fleet achieved 35 MW, so the quantity the capacity product is settled on is overstated by 57 %; the forecast implies 220 MWh of curtailed energy and 182 MWh was delivered; and the target was met for one hour out of four rather than for the whole block. The model missed the response decay, which is the single feature of the trajectory the dispatch optimizer needed, because a thermostat fleet whose buildings are warming cannot hold a reduction indefinitely. Validation that reports only MAE = 9.5 MW does not reveal any of this.

9.4.3 Errors conditional on the decision

Validation can be narrowed further, to the region of the forecast that the decision depends on. Suppose the only use of the flexibility forecast $\hat{F}$ is a yes-or-no commitment: can 75 MW be scheduled for tomorrow evening? Then errors far from 75 MW are of no consequence and errors near it are decisive.

Two cases make the asymmetry concrete. If $\hat{F} = 30$ MW and $F = 25$ MW, the error is 5 MW and no event was scheduled either way, so nothing happened; the error is recorded in the metric and cost nothing. If $\hat{F} = 80$ MW and $F = 65$ MW, the error is 15 MW, the commitment

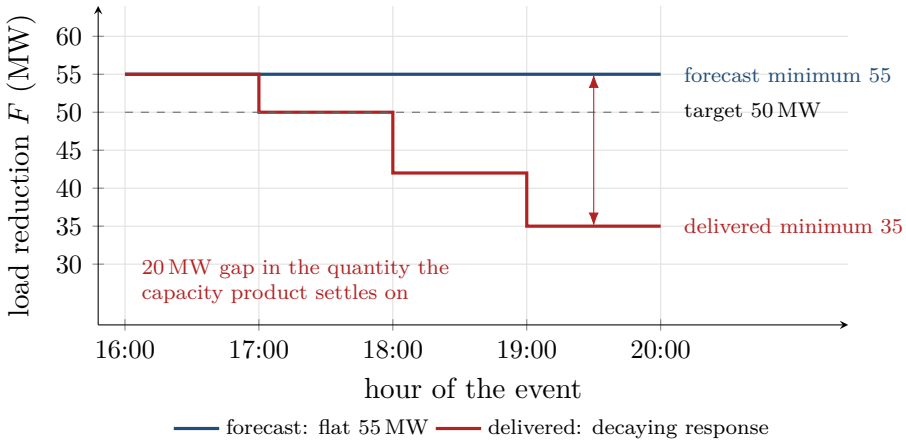


Figure 9.4: A forecast with an acceptable timestep error and an unusable event description. The step lines are the forecast and delivered reductions of Section 9.4.2, hour by hour; the dashed line is the 50 MW target. The timestep mean absolute error is 9.5 MW; the minimum sustained reduction, the quantity a capacity product pays for, is overstated by 20 MW. Values are illustrative.

was made on the strength of the forecast and the fleet failed to deliver 10 MW of it: a penalty, a reliability event and a damaged relationship with the utility. A metric that averages both occasions weights them by nothing but their magnitude, and the magnitudes are in the wrong ratio by a wide margin. The practical remedy is to compute the metrics of Section 9.2 on the subset of occasions near the decision threshold, and to count operational failures $\hat{F} \gg F$ separately as their own statistic.

9.4.4 Asymmetric loss

Generalizing that observation gives the cost function the validation should use. Overprediction, $\hat{F} > F$, leads to a commitment the fleet cannot meet: underdelivery, penalties, and lost standing. Underprediction, $\hat{F} < F$, leaves usable flexibility unsold: an opportunity cost, usually smaller. Writing $e = \hat{F} - F$,

$$L(e) = \begin{cases} c_{\text{over}}\,|e|, & \hat{F} > F \quad \text{(underdelivery)}, \\ c_{\text{under}}\,|e|, & \hat{F} < F \quad \text{(unused flexibility)}, \end{cases} \qquad c_{\text{over}} > c_{\text{under}}. \tag{9.16}$$

Figure 9.5 draws it. The horizontal axis is the signed error e, the vertical axis is the operational cost in relative units, and the function is two straight lines meeting at the origin, with the right-hand branch, overprediction, four times as steep as the left. Mean absolute error is the special case $c_{\text{over}} = c_{\text{under}}$, drawn as the symmetric dashed pair; it is the only member of the family that assumes the two consequences are equally bad, and it is also the one almost always reported.

Equation (9.16) has two consequences that run through the rest of the book. Minimizing its expectation does not give the mean of the forecast distribution but a quantile of it, below the median when $c_{\text{over}} > c_{\text{under}}$, so the best point forecast under an asymmetric cost is a deliberately conservative number; and computing that quantile requires the whole distribution $p(y_{t+h} \mid \mathcal{I}_t)$ rather than a point prediction, which is the subject of Chapter 10. Validation, in other words, points directly at probabilistic forecasting and at the dispatch optimization that consumes it.

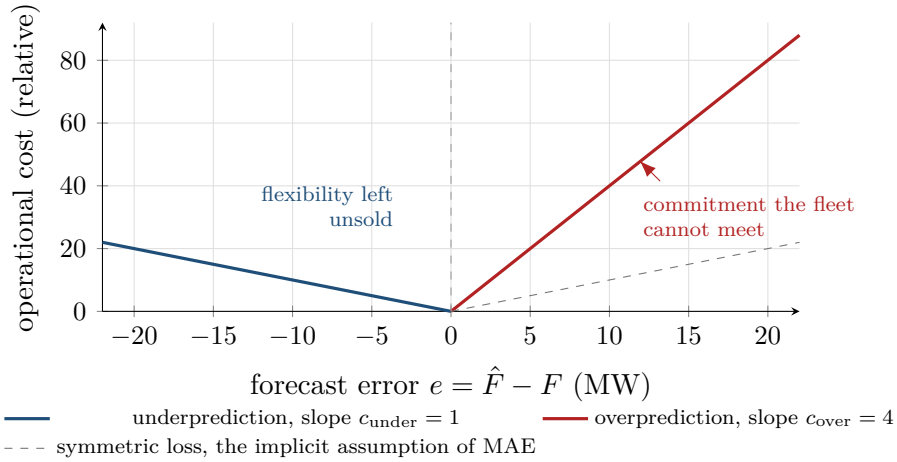


Figure 9.5: The asymmetric loss of equation (9.16) with $c_{\text{under}} = 1$ and $c_{\text{over}} = 4$ in relative units. The dashed line is the symmetric branch that mean absolute error assumes. The ratio $c_{\text{over}}/c_{\text{under}}$ is illustrative and is set in practice by the penalty structure of the contract.

9.5 Leakage, revisions and drift

Three failure modes account for most of the gap between a validated number and a production number: information the backtest should not have had, information arriving later than its timestamp suggests, and a relationship that changes after the model was fitted. This section treats each, then turns to the residual structure that diagnoses what the model has not learned.

9.5.1 Information leakage

Section 9.1.4 stated the reconstruction requirement. The quantitative consequence of breaking it is large. Build a three-hour-ahead model whose weather features are the temperatures that were actually observed at the target time, and it may report RMSE = 4 MW. Rebuild the same model with the weather *forecasts* that were available at each issue time, and the error rises to RMSE = 7 MW. The first model is not a worse-documented version of the second; it is a model of a different task, namely predicting load given tomorrow's weather, and that task is not available to anyone at issue time. Deploying it yields the 7 MW error while the report promises 4.

A subtler version of the same problem comes from data cleaning. Historical telemetry is usually retrospectively corrected: gaps are filled, meter errors are fixed, duplicate records are reconciled and outliers are removed, all with the benefit of hindsight. The archive is therefore cleaner than the real-time stream the production model consumes, and a model trained and validated on the archive meets messier inputs in operation. This offline-online mismatch produces a model that degrades on deployment for reasons invisible in any backtest. A production-quality backtest reproduces the information state that existed, in both senses: the right values and the right quality.

9.5.2 Latency, not timestamp

Point-in-time correctness requires two times per measurement, and confusing them is the most common form of leakage in a system that has already been told to respect time order. A device measurement carrying the timestamp 13:59 may have arrived in the database at 14:04, because of a communications gateway retry, a batching interval, or a bulk-upload schedule. It was not available to a forecast issued at 14:00. A backtest that selects features by the condition

"timestamp < issue time" includes it anyway and so uses the future.

The correct filter is on availability time, not measurement time, which means the historical store must record both and the feature pipeline must honor the distinction. The effect is concentrated exactly where it does the most harm: at short horizons, where the most recent measurements dominate the forecast and where the latency of the last few minutes of telemetry is a large fraction of the horizon itself.

9.5.3 Forecast revisions

A forecasting system does not issue one forecast for a target time; it issues a sequence of them as the target approaches and information accumulates. Table 9.3 gives the sequence for a 17:00 target: the first column is the issue time, the second the horizon, the third the forecast, and the fourth the error against the realized 106 MW. Each row is a separate forecasting task with its own information set, and the table should be read as four evaluations rather than one.

Table 9.3: The revision sequence for a single 17:00 target. Each row is an independent forecast with its own horizon and information set; the realized value was 106 MW.

Issue time	Horizon h	Forecast (MW)	Error $y - \hat{y}$ (MW)
12:00	5 h	92	+14
13:00	4 h	96	+10
14:00	3 h	100	+6
15:00	2 h	104	+2

Figure 9.6 plots the same sequence. The horizontal axis is the issue time, running from 12:00 to the 17:00 target; the vertical axis is megawatts; the blue markers are the four forecasts and the dashed horizontal line is the realized value. The vertical distance from each marker to the dashed line is the error, and it shrinks monotonically as the horizon closes. That pattern is what a functioning system should produce, and the discipline it implies is to evaluate each issuance separately: pooling the four into one average would hide the only question the sequence answers, which is whether the forecast improves as information arrives.

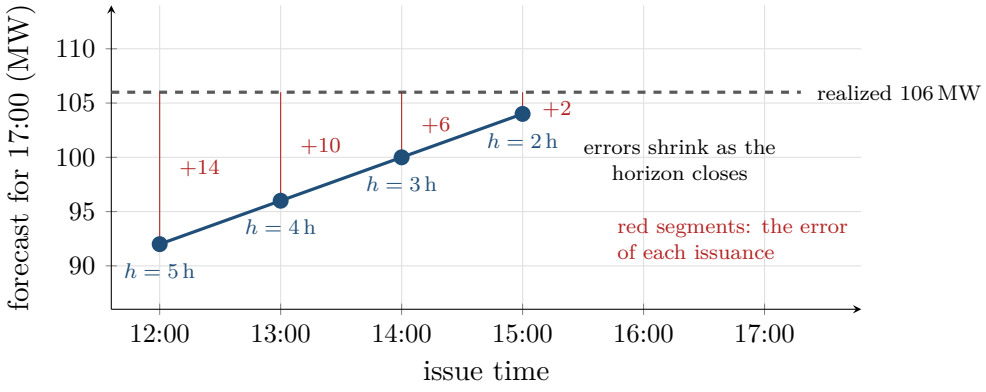


Figure 9.6: The revision sequence of Table 9.3. Blue markers are the forecasts issued at 12:00 through 15:00 for a 17:00 target; the dashed line is the realized 106 MW and the thin red segments are the errors. The monotone shrinking is the behavior a system that gains information should show.

Revision behavior as a diagnostic. As h decreases the uncertainty should decrease, because $\mathcal{I}_t$ grows. When it does not, when the one-hour-ahead forecast is consistently worse than the

six-hour-ahead forecast for the same target, something specific is wrong and the list of candidates is short: the real-time telemetry feeding the short horizon is noisier than the model assumes; a feature is computed with the wrong timing, so the short-horizon model is reading a stale or a future value; retraining is unstable, so the deployed short-horizon model differs from the validated one; the revised weather forecast is worse than the earlier one, which happens in rapidly changing conditions; or there is a bug in the feature pipeline. Revision behavior is cheap to monitor and points at the failure, which makes it one of the most useful diagnostics in production.

9.5.4 Drift across test years

Rolling-origin backtesting produces one error per fold, and the sequence of them is more informative than their mean. Suppose the same model specification, refitted on the data available before each test year, gives

$$\text{RMSE}: \quad 4.8,\ 5.0,\ 5.4,\ 6.5,\ 7.3\,\text{MW} \quad \text{for test years } 2022,\ 2023,\ 2024,\ 2025,\ 2026. \tag{9.17}$$

Figure 9.7 plots these as bars: the horizontal axis is the test year, the vertical axis is the root-mean-square error in megawatts, and the dashed line marks the first year's level for reference. The error grows every year and is 52 % above its starting value by the last, which is model degradation rather than a run of bad luck.

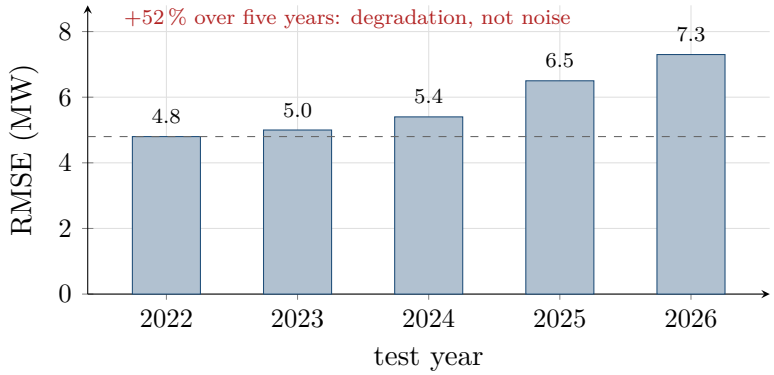


Figure 9.7: Error by test year from equation (9.17), each fold refitted on the data available before its test year. The dashed line is the 2022 level. Values are illustrative.

The causes are the ones that made the rolling window of Section 9.1.3 attractive: the fleet grew and changed composition, so the mix of thermostats, vehicles and batteries behind the aggregate is not the mix the model was fitted to; the customer population turned over, bringing different behavior; the weather regime shifted, so the hot tail of the training distribution is no longer the hot tail of the operating distribution; firmware updates changed how devices execute a command; and the feature pipeline itself was modified, so a feature does not mean in 2026 what it meant in 2021. A backtest is a snapshot, and the quantity an operator needs is the rate at which that snapshot goes stale. Monitoring it is the subject of distribution shift, taken up later in the book; the point here is that the fold-by-fold sequence, not its average, is what reveals it.

9.5.5 Residual plots

Aggregate metrics say how large the errors are. Residual structure says what the model still does not understand, and it is diagnosed by plotting the residual $e_t = y_t - \hat{y}_t$ against things it should be independent of. Six plots carry most of the information: the residual against outdoor

temperature, against the horizon, against the hour of day, against the predicted value, against calendar time, and the residual autocorrelation function of Chapter 2.

The logic is the same in every case. If the model has extracted all the signal available in a variable, the residual carries no remaining dependence on it: the cloud of points is flat and featureless. Any visible pattern is unextracted signal, and the shape of the pattern names the omission. Curvature against the predicted value indicates a missing nonlinearity; a step between weekdays and weekends indicates a calendar feature that is too coarse; a drift against calendar time is the degradation of Section 9.5.4; and a fan that widens with the predicted value indicates heteroscedasticity, which matters for the interval forecasts of Chapter 10 even where it leaves the point forecast unbiased.

9.5.6 Error increasing with temperature

The most common instance in this domain is residual error that grows with outdoor temperature. Table 9.4 gives a representative pattern: the first column is the temperature bin, the second the mean absolute error within it, and the third that error relative to the mildest bin. The error is five times larger in the hottest bin than in the mildest one, and that is precisely the bin in which the forecast is used.

Table 9.4: Mean absolute error by outdoor-temperature bin for a single model, showing error that grows with temperature. Values are illustrative.

Temperature bin	MAE (MW)	Relative to the 20–25°C bin
20–25°C	2	1.0×
25–30°C	3	1.5×
30–35°C	5	2.5×
35–40°C	10	5.0×

Four explanations compete, and they call for different work. The physical relationship may saturate, as air-conditioning load does when compressors reach capacity and can no longer respond to further temperature increase, which a linear or an inadequately featured model cannot represent. The training data may be sparse above 35°C, since extreme days are rare by construction, so the model is extrapolating. The weather forecast itself may be less accurate in extreme conditions, in which case the load model is being blamed for an input error, a distinction Section 9.6.6 makes operational. Or the fleet may genuinely be more variable near capacity, because small differences in building thermal mass and in customer override behavior have larger consequences when every system is saturated. Distinguishing them requires the attribution of Section 9.6.6 and the per-regime sample counts of Section 9.6.3, not a better average metric.

9.5.7 Autocorrelated errors

The last diagnostic is the residual autocorrelation. If

$$\operatorname{Corr}(e_t,\, e_{t-1}) = 0.7, \tag{9.18}$$

the model's mistakes persist: when it is reading too high now it will very likely be reading too high in the next interval. The statistical reading is that the residuals are not the independent noise the fitting procedure assumed, so standard errors and prediction intervals computed under independence are too narrow. The physical reading is more useful: there is a temporal state the model does not represent. Something about the fleet persists across intervals, building thermal mass, a dispatch-induced recovery, a weather-forecast bias on that particular day, and the features do not contain it.

Chapter 6 gave the machinery for responding. The options are to engineer the missing state into the feature vector explicitly, by adding the lags, rolling statistics or physical state variables that carry it; to fit a residual-correction model on top of the point forecast, which is what an autoregressive error term does; or to move to a sequence model whose hidden state $\mathbf{h}_t$ can hold the information, as in Chapters 7 and 8. In all three cases the validation diagnostic has set the development agenda, which is the point of computing it: validation is not only a gate at the end of model development but an input to it.

9.6 Validation protocol and a worked comparison

The pieces assembled so far become a protocol only when the order of operations is fixed, because the act of choosing a model on an evaluation set spends that set. This section states the protocol, distinguishes the three kinds of generalization a forecaster is asked about, treats the uncertainty of a metric, and works a four-model comparison through to a decision.

9.6.1 Model selection must happen inside validation

Suppose six candidates are available: ridge regression, gradient boosting, a random forest, an LSTM, a GRU and a temporal fusion transformer. Fitting all six, scoring them on the 2024 test year and reporting the best score is a biased estimate of that model's future performance, and the bias has a clear mechanism: the maximum of six noisy estimates is larger than any one of them in expectation, so part of the winner's apparent advantage is the luck of that particular test year. The more candidates are compared, the larger the effect.

The protocol that avoids it splits the record three ways rather than two. Train on 2019–2022; use 2023 as a validation period on which hyperparameters, architectures and feature sets are chosen; and use 2024 once, at the end, as the test period that estimates the chosen model's performance. Everything that involves a choice happens on the validation period, and the test period is touched exactly once. If the test number disappoints, the honest response is to report it, not to return to the test period with a new candidate, because that converts the test period into a second validation period and reintroduces the bias.

Time-series cross-validation. One validation period is subject to the same accident of timing as one test period, so the validation stage is itself usually several folds:

$$\begin{aligned} &\text{fold 1:} \quad \text{train 2019–2021,} \quad \text{validate 2022,} \\ &\text{fold 2:} \quad \text{train 2019–2022,} \quad \text{validate 2023,} \\ &\text{fold 3:} \quad \text{train 2019–2023,} \quad \text{validate 2024,} \end{aligned} \tag{9.19}$$

with 2025 reserved as an untouched final test period. A candidate is selected on the average validation error across the three folds, and is preferred when it wins consistently rather than on one fold.

Hyperparameter tuning respects time too. The same rule applies one level down. Choosing a boosted-ensemble depth of 6 is a decision, and it must be made on the folds of equation (9.19) rather than by randomly cross-validating rows. Random k-fold cross-validation over rows is the default in every general-purpose machine-learning library, and using it here leaks future information into the selection of depth, learning rate, number of estimators, regularization weight λ and feature set, with the consequences of Section 9.1.1. The chosen configuration is then tuned to a task nobody will run.

9.6.2 Generalization across devices, geographies and conditions

"Does the model generalize" is three questions, and a validation design answers only the one it was built for. The distinction is clearest at device level. A random split of device-time rows puts thermostat 47 in both the training and the evaluation set, at different hours. The resulting score answers: can the model forecast devices it has already seen, on dates it has not? That is a legitimate and common operational question, since most of an aggregator's fleet on any given day is already enrolled. It is not the question asked when the sales team enrolls ten thousand new customers next month. Answering that one requires a device-held-out design, in which entire devices are absent from training and the model is scored on them.

Table 9.5 sets out the three tests: the first column names the test, the second and third state what is in the training and evaluation sets, and the fourth gives the question it answers. The three stress different capabilities, and a model can pass one and fail another. A global model of the kind Chapter 7 described, fitted across many devices, usually does well on new devices and can still fail on new conditions, because nothing in its training distribution resembled a 40°C afternoon.

Table 9.5: Three generalization tests. Each requires its own split design, and each answers a different operational question.

Test	Training set	Evaluation set	Question answered
Future-time	known devices, past dates	the same devices, later dates	can the model forecast the fleet it serves, forward in time?
New-device	a subset of devices, all dates	devices never seen, familiar conditions	can the model forecast a newly enrolled thermostat?
New-condition	all devices, ordinary conditions	the same fleet, unusual weather or event types	can the model forecast the regime that creates the value?

Geographic generalization. The same argument applies across territories. An aggregator operating in Texas, New York and California faces different climates, different building stocks, different air-conditioning penetration and different market rules. A model fitted and evaluated on the pooled record can score well while generalizing poorly to any single region, because the pooled evaluation set contains mostly the regions that are easy. The test that answers the deployment question is leave-one-region-out: train on Texas and New York, evaluate on California, and repeat. A large gap between pooled and held-out-region performance says that the model has learned region-specific effects it cannot transfer, which is the information needed before launching in a fourth state.

9.6.3 Fair comparison and the uncertainty of a metric

Comparisons must be made on identical ground. Reporting that model A achieved RMSE = 6.0 MW over 2022–2024 while model B achieved 6.3 MW over 2023–2025 compares two models and two different periods, and the period difference can easily exceed the model difference, as Figure 9.7 showed. A fair comparison holds four things fixed: the same target quantity, the same issue times, the same information available at each issue time, and the same evaluation periods. Anything else is a comparison of experiments rather than of models.

Sample size in the regime that matters. A claim about a regime is only as strong as the number of occasions in it. "Model B reduced extreme-heat RMSE by 30 %" computed over

three heat events is weak evidence: three events can differ in duration, time of day, antecedent conditions and fleet state so much that the comparison is dominated by which three they were. The same claim over five hundred events is convincing. Every sliced metric should therefore be reported with the count of observations behind it, and a slice with a handful of occasions should be described as a signal to investigate rather than as a result.

Uncertainty of a metric. A metric is itself an estimate and carries sampling error. $\mathrm{RMSE}_{\mathrm{A}} = 5.0\,\mathrm{MW}$ against $\mathrm{RMSE}_{\mathrm{B}} = 4.9\,\mathrm{MW}$ on one test year is not evidence that B is better: on another year the numbers might be 4.8 and 5.1, with the ranking reversed. Two devices address this. The first is to use several backtest periods, as in equation (9.19), and ask for consistency: a model that wins on three folds out of three is a different proposition from one that wins on the average. The second is to resample the evaluation set, computing the metric on many resamples to obtain its spread. The resampling must respect the dependence structure: drawing individual timestamps with replacement destroys the autocorrelation of Section 9.5.7 and understates the spread badly. The unit of resampling is a block that is close to independent of its neighbors, a whole day, a whole event, or a multi-day block, which is a block bootstrap rather than an ordinary one.

9.6.4 Production validation and a validation plan

Research validation asks one question: which model attains the lowest test error? Production validation asks a longer list, and a model can win the first and fail the second. Does the model still produce a forecast when a quarter of the telemetry is missing? What does it do when the weather application programming interface responds late or not at all, and does it fall back to a persistence forecast or return nothing? Does it fail gracefully, with a degraded but usable output and a flag, or catastrophically? How fast does it degrade without retraining, which is the drift rate of Figure 9.7? Is the forecast reproducible, so that the same inputs give the same output and an incident can be investigated? Does scheduled retraining change the model's behavior in ways downstream optimization has to absorb? And is there drift detection and alerting, so that degradation is noticed before an operator notices it? These are the concerns of operating a forecasting system, taken up later in the book.

A validation plan collects all of the above for one concrete task. For a three-hour-ahead forecast of thermostat-fleet flexibility, the plan has six parts.

1. **Baselines.** The previous similar event, and a weather regression, following Chapter 3. The skill of equation (9.13) is reported against both, and against whatever is currently in production.

2. **Models.** Ridge regression, gradient boosting, and an LSTM, so that the comparison spans a linear model, a tree ensemble and a sequence model.

3. **Periods.** Rolling-origin folds as in equation (9.19), spanning several seasons and several years, so that seasonal and interannual variation both enter.

4. **Metrics.** MAE, RMSE and Bias at each horizon, plus the event-level quantities of equation (9.15).

5. **Slices.** Horizon, temperature, fleet size, geography, event duration, previous dispatch, and extreme heat as its own slice, each reported with its sample count.

6. **Operational failures.** The occasions with $\hat{F} \gg F$, counted and described individually rather than averaged, since these are the events that cost money.

9.6.5 A worked model comparison

Table 9.6 applies that plan to four candidates. The first column names the model; the second gives the overall root-mean-square error across the evaluation periods; the third gives the same metric restricted to heat-wave hours; the fourth gives the bias, positive meaning overprediction as in equation (9.8); the fifth gives the fraction of events in which the fleet delivered less than was committed; and the last gives the skill of equation (9.13) against the seasonal baseline.

Table 9.6: Four candidates for the three-hour-ahead flexibility forecast, evaluated on identical periods and issue times. Errors and bias in MW; underdelivery as a percentage of events; skill against the seasonal baseline by equation (9.13). Values are illustrative.

Model	Overall RMSE	Heat-wave RMSE	Bias	Under-delivery	Skill
Seasonal baseline	10.0	18.0	+3.0	22 %	0.00
Ridge regression	7.5	13.0	+2.0	16 %	0.25
Gradient boosting	6.2	8.0	+0.8	8 %	0.38
LSTM	6.0	9.5	+0.5	10 %	0.40

Read column by column, the table does not produce a single winner. On overall error the LSTM is best, at 6.0 MW against the boosted ensemble's 6.2 MW, a margin of 3.2 %. On heat-wave error the ranking reverses and the margin is much larger: 8.0 against 9.5 MW, so the boosted ensemble is 16 % better in the regime that generates most of the program's value. On underdelivery the boosted ensemble is better again, 8 % of events against 10 %. On bias the LSTM is slightly better, +0.5 against +0.8 MW, and both are far better than the baseline's +3.0 MW, which by Section 9.2.4 is the kind of systematic overprediction that produces repeated overcommitment.

The decision therefore depends on the objective, and the objective must be stated before the table is read. If the program is paid for average energy across the year, the LSTM's 0.2 MW advantage in overall error is the relevant quantity. If it is paid for capacity during system peaks and penalized for underdelivery, the boosted ensemble wins on both of the columns that matter, and its simplicity and lower inference cost are additional arguments. Nothing in the data resolves the choice; what resolves it is a decision about what the forecast is for.

What "better" means. Generalizing, a forecast is better only relative to a stated objective, and at least six are in use: minimum mean absolute error, minimum root-mean-square error, minimum absolute bias, minimum risk of underdelivery, minimum expected dispatch cost, and maximum capacity that can be safely committed. The first three are statistical metrics computed on the forecast alone. The last two are decision metrics: they depend on the forecast, on the optimizer that consumes it and on the contract that prices the outcome, and they are the ones an operator is accountable for. Validation against a decision metric is taken up later in the book; the step available now is to notice which objective a reported comparison has implicitly assumed.

Conceptual questions

"How would you validate a new load forecasting model?"
A strong answer describes a procedure rather than a metric. It begins with time-ordered backtesting over multiple rolling or expanding windows, so that every evaluation is out-of-time and no single test period dominates. It insists that each historical prediction be reconstructed from the information available at its issue time, including the historical weather forecasts rather than the observed weather. It compares the model against credible baselines on exactly the same periods and issue times, and reports skill against the system currently in production as well as against a simple reference. It reports mean absolute error, root-mean-square error and bias broken down by horizon, not pooled across horizons. It slices by extreme temperature and by dispatch status, with the sample count in each slice. And it inspects the residual structure, including the residual autocorrelation, to say what the model has not learned rather than only how large its errors are.

Conceptual questions

"Our new model improves annual RMSE by 5 %. Would you deploy it?"
The answer is that the number alone is not sufficient grounds, and the reply should list what would be. Where does the improvement come from: is it spread across the year or concentrated in a few periods? Is it consistent across backtest folds and across horizons, or does it appear at one horizon and reverse at another? How does the model perform during dispatch events, in extreme weather, and when telemetry is incomplete? How do the bias and the underdelivery rate compare, since a model that lowers average error while becoming overconfident in critical events is operationally worse than the model it replaces? What is the added operational cost in retraining, monitoring and inference? A 5 % annual improvement that is absent on heat-wave afternoons is not a reason to change a production system.

Conceptual questions

"Why can't we randomly cross-validate a time-series model?"
Because forecasting is directional in time and a random split removes the direction. A randomly chosen training set contains periods later than the evaluation set, so the model is scored on a task in which the future is already known, which is not the task it will perform. The effect is amplified by the structure of the data: lagged and rolling features make adjacent rows near-duplicates, so a random split places near-copies of the evaluation rows into training, and the correlation between neighboring observations means the evaluation set is not independent of the training set even where no row is shared. Valid validation trains on the past and predicts a future it has not seen, which is what rolling-origin backtesting does.

9.6.6 Paired comparison, attribution and oracle validation

Three refinements turn a comparison into a diagnosis.

Paired comparison. Comparing two models through their aggregate metrics discards the information that both were evaluated on the same occasions. Running both over identical timestamps and forming the paired difference

$$d_t = |e_{\mathrm{A},t}| - |e_{\mathrm{B},t}| \tag{9.20}$$

gives a series whose sign says which model was closer at each instant. The mean of d_t recovers the difference in mean absolute error, and its structure says much more: inspecting when $d_t > 0$ and when $d_t < 0$ locates each model's advantage in time, in temperature, in horizon and in

dispatch status. A model that wins on average by losing badly on a few hot afternoons and winning slightly on many mild nights is identified immediately by this plot and is invisible in the aggregate.

Error attribution. When the largest misses are examined individually, a pattern often emerges: they occur on the days when the weather forecast was poor. That changes the diagnosis. The chain is

$$\text{weather-forecast error} \implies \text{load-forecast error}, \tag{9.21}$$

and the second term is partly a consequence of the first. No downstream model can recover information the weather forecast does not contain, so effort spent on the load model will not remove that component of the error. The useful decomposition separates model error given perfect weather from total operational error, and the gap between them is the part that must be bought from a better weather provider, from an ensemble of weather forecasts, or by accepting it as irreducible.

Oracle validation. The decomposition is measured by running the model twice over the same period: once with the weather forecasts that were operationally available, and once with the temperatures that were later observed. The second run is an oracle, impossible in operation and informative as a diagnostic. Suppose it gives

$$\mathrm{RMSE}_{\text{operational}} = 8\,\mathrm{MW}, \qquad \mathrm{RMSE}_{\text{oracle}} = 5\,\mathrm{MW}. \tag{9.22}$$

Figure 9.8 draws the two numbers as stacked bars: the horizontal axis distinguishes the operational run from the oracle run, the vertical axis is root-mean-square error in megawatts, the green portion of each bar is the 5 MW attained with observed weather, and the red portion is the 3 MW difference. The caption records an important caveat about the stack, which is that the two components are not additive in root-mean-square error: if the weather-driven and model-driven errors were independent, the implied weather component would be $\sqrt{8^2 - 5^2} = 6.2\,\mathrm{MW}$ rather than 3 MW.

The number to take from the exercise is the 8 MW. A significant part of the operational error is weather-forecast uncertainty, which sets a floor on what model development can achieve and tells the forecaster where to spend effort. The 5 MW must never be reported as achievable operational performance, because no operational system has the observed future temperature. Quoting an oracle number as a forecast accuracy is the leakage of Section 9.5.1 with a diagnostic justification attached.

9.6.7 From a number to a distribution

Every source of error in this chapter is a source of uncertainty: the weather forecast, customer behavior, device availability, and the model itself. A single point forecast $\hat{y}_{t+h}$ cannot express any of them, and the decisions the forecast feeds, how much capacity to commit, which quantile of equation (9.16) to report, whether a 75 MW commitment is safe, are decisions about the tail of a distribution rather than about its center.

The object that answers them is the conditional distribution $p(y_{t+h} \mid \mathcal{I}_t)$ of Chapter 1, and once that is the output, validation acquires a new question. It is no longer only whether the center is close but whether the stated uncertainty is honest: does an interval declared to contain the outcome 90 % of the time actually contain it on 90 % of occasions? That property is calibration, and the metrics and procedures for it are the subject of Chapter 10.

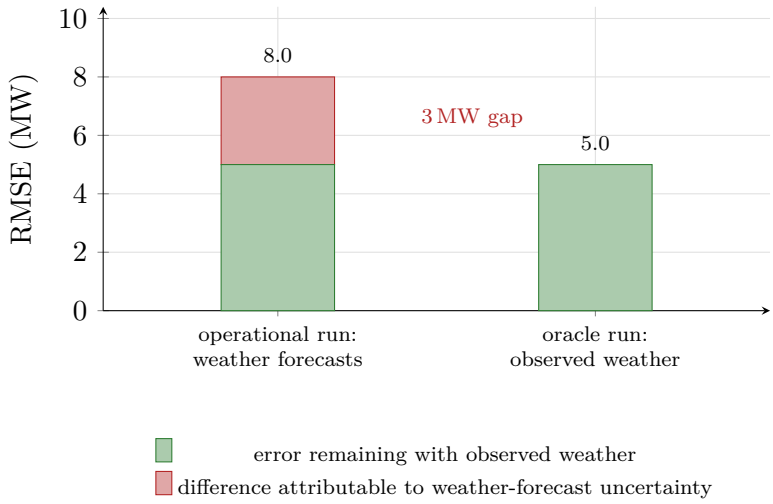


Figure 9.8: Oracle validation, from equation (9.22). The left bar is the operational error obtained with the weather forecasts that existed at each issue time; the right bar is the error obtained with the later-observed weather. The split of the left bar marks the two numbers and is not a variance decomposition: root-mean-square errors do not add, and under independence the implied weather component would be $\sqrt{8^2 - 5^2} = 6.2\,\text{MW}$. Values are illustrative.

9.7 What to retain

Definition 9.1 (Forecast validation)**.** Validating a forecast means recreating the actual information flow, past to issue time to unseen future, and then reducing the resulting out-of-time forecasts to the quantities a decision depends on: mean absolute error for the typical miss, root-mean-square error where a large miss deserves a stronger penalty, and bias for systematic error, each broken down by horizon, operating regime, event type and fleet condition, and each compared against a credible baseline. The crucial question is not which model has the lowest average error but which forecast supports reliable dispatch when the system actually matters.

Table 9.7 collects the chapter's objects with the subsection that defines each.

9.8 Exercises

Exercise 9.1. A model produces the signed errors $e_i = y_i - \hat{y}_i$ equal to $[+3, -1, -6, +2, +2]\,\text{MW}$ on five occasions. Compute MAE, RMSE, the ratio RMSE/MAE and Bias as defined in equations (9.4), (9.6) and (9.8). Then construct a second error vector of the same length with the same MAE and the largest RMSE you can, and state the maximum value of RMSE/MAE attainable with $N = 5$.

Exercise 9.2. A seasonal baseline attains RMSE $= 12.0\,\text{MW}$ on a test year, the incumbent production model attains $8.4\,\text{MW}$, and a candidate attains $8.0\,\text{MW}$. Compute the skill of equation (9.13) for both models against the baseline, and the candidate's improvement over the incumbent. Using Table 9.6 as a template, list the three additional columns you would require before recommending the candidate, and say what value of each would change your recommendation.

Exercise 9.3. Hourly data is available from January 2020 through December 2025, and the task is a three-hour-ahead flexibility forecast. Design a backtest: state the folds for an expanding window and for a rolling window of two years, following equations (9.2) and (9.3), and say which periods are used for hyperparameter selection and which is reserved as the final test.

Table 9.7: The objects of forecast validation.

Object	Definition	Section
rolling origin	repeated out-of-time evaluation with the issue time advanced	9.1.2
expanding window	training set keeps all history, $D_1 \subset D_2 \subset D_3$	9.1.3
rolling window	training set of fixed length, oldest data discarded	9.1.3
issue-time replay	every feature rebuilt from $\mathcal{I}_t$, including $\hat{T}_{t+h\|t}$	9.1.4
MAE	$\frac{1}{N}\sum\|y_i - \hat{y}_i\|$: typical error magnitude	9.2.1
RMSE	$\sqrt{\frac{1}{N}\sum(y_i - \hat{y}_i)^2}$: penalizes large misses	9.2.2
Bias	$\frac{1}{N}\sum(\hat{y}_i - y_i)$; positive means overprediction	9.2.4
MAPE, NRMSE	scaled errors; MAPE unusable as $y_i \to 0$	9.2.5
regime slice	metric restricted to a condition that carries the value	9.3.2
MAE_h	error as a function of horizon; the error-growth curve	9.3.3
Skill	$1 - \text{RMSE}_{\text{model}}/\text{RMSE}_{\text{base}}$; name the baseline	9.3.4
event metrics	peak, average, curtailed energy, minimum sustained, rebound	9.4.1
$L(e)$	asymmetric loss, $c_{\text{over}} > c_{\text{under}}$	9.4.4
leakage	use of anything absent from $\mathcal{I}_t$, including late arrivals	9.5.1, 9.5.2
revision sequence	forecasts for one target at decreasing h; a diagnostic	9.5.3
drift	fold-by-fold growth of the error across test years	9.5.4
residual structure	e_t plotted against what it should not depend on	9.5.5
block bootstrap	resampling days or events to obtain a metric's spread	9.6.3
oracle run	the model re-run with observed weather, as a diagnostic only	9.6.6

Then list, for each of the following features, whether it is admissible at a 14:00 issue time and why: the 13:00 fleet load; the observed 17:00 temperature; the 13:55 duty cycle, which arrives in the database at 14:03; the event duration as finally executed; and the day-ahead temperature forecast issued at 06:00.

Exercise 9.4. An event runs from 16:00 to 20:00 against a 60 MW target, with

$$\hat{F} = [65,\ 65,\ 60,\ 60]\,\mathrm{MW}, \qquad F = [66,\ 58,\ 48,\ 44]\,\mathrm{MW}$$

in hourly steps. Compute the four event metrics of equation (9.15) for both the forecast and the delivery, and the timestep MAE. Then evaluate the asymmetric loss of equation (9.16) with $c_{\mathrm{over}} = 4$ and $c_{\mathrm{under}} = 1$, summed over the four hours, and compare the ranking it gives with the ranking that MAE gives for an alternative forecast of $[55, 50, 45, 42]$ MW.

Chapter 10

Probabilistic forecasting: quantiles, calibration and sharpness

10.1 From a point forecast to a distribution

Every forecast so far has been one number. A model trained as in Chapters 5 to 8 and validated as in Chapter 9 returns $\hat{y}_{t+h} = 45\,\text{MW}$, and the evaluation asks how far that number lands from the outcome. The operational question an aggregator actually faces is different, and this chapter's question is

How much flexibility is likely to be available, and how uncertain are we?

The object that answers it is not a number but a distribution,

$$p(F \mid \mathcal{I}_t) \qquad \text{or} \qquad p(F \mid \mathcal{I}_t, u) \tag{10.1}$$

when the flexibility depends on a planned control action u. The information set $\mathcal{I}_t$ is the one defined in Chapter 1: everything genuinely available at issue time. What changes in this chapter is only the shape of the output.

Jargon: probabilistic forecast, quantile, prediction interval, nominal coverage, calibration, sharpness

A *point forecast* is a single number $\hat{y}_{t+h}$. A *probabilistic* (or distributional) *forecast* is the whole distribution $p(y_{t+h} \mid \mathcal{I}_t)$, usually reported as a few quantiles. The *τ-quantile* q_τ of a distribution is the value below which a fraction τ of the probability lies, so $\Pr(Y \le q_\tau) = \tau$; $q_{0.10}$, $q_{0.50}$ and $q_{0.90}$ are written P10, P50 (the median) and P90. A *prediction interval* is a pair of quantiles reported as a range, and its *nominal coverage* is the probability the pair is supposed to enclose: [P05, P95] is a nominal 90 % interval. A forecaster is *calibrated* when the nominal coverage matches the frequency observed over many forecasts, and *sharp* when its intervals are narrow.

Key idea

Two forecasts can carry the same point estimate and imply completely different operational risk. The deliverable of a forecasting system is therefore the conditional distribution of the target, reported as quantiles, not its center.

This section shows with one example why the center alone is insufficient, then fixes the vocabulary used for the rest of the chapter: distribution, quantile, interval.

10.1.1 A point forecast can be misleading

Consider two models forecasting the flexibility available from a thermostat fleet at a future hour. Model A reports $F \sim \mathcal{N}(45, 2^2)$ MW and model B reports $F \sim \mathcal{N}(45, 15^2)$ MW. Both have the same mean, 45 MW, so every point metric of Chapter 9 computed on the central estimate alone is identical. The statements they make about the world are not. Model A says the outcome will be tightly concentrated near 45 MW; model B says it could plausibly land anywhere over a 30 MW span.

The two statements differ operationally as soon as a requirement is attached. Suppose the utility needs at least 40 MW. The quantity that decides whether the commitment is safe is not $\mathbb{E}[F]$ but the exceedance probability $\Pr(F \geq 40)$, which for a normal model is read off the standard normal distribution function Φ,

$$\Pr(F \geq 40) = \Phi\left(\frac{45 - 40}{\sigma}\right) = \begin{cases} \Phi(2.50) = 0.994, & \sigma = 2\,\text{MW (model A)}, \\ \Phi(0.33) = 0.631, & \sigma = 15\,\text{MW (model B)}. \end{cases} \tag{10.2}$$

Figure 10.1 draws the two densities. Both panels put delivered flexibility on the horizontal axis and forecast density on the vertical axis, each with its own vertical scale because the narrow density is almost eight times taller; the vertical red line in each panel is the 40 MW requirement and the shaded area to its left is the probability of falling short. In the upper panel the shaded sliver is six outcomes in a thousand. In the lower panel it is more than a third of the mass. The same 45 MW central estimate therefore supports a confident commitment in one case and a coin-flip in the other, and nothing in the point forecast distinguishes the two.

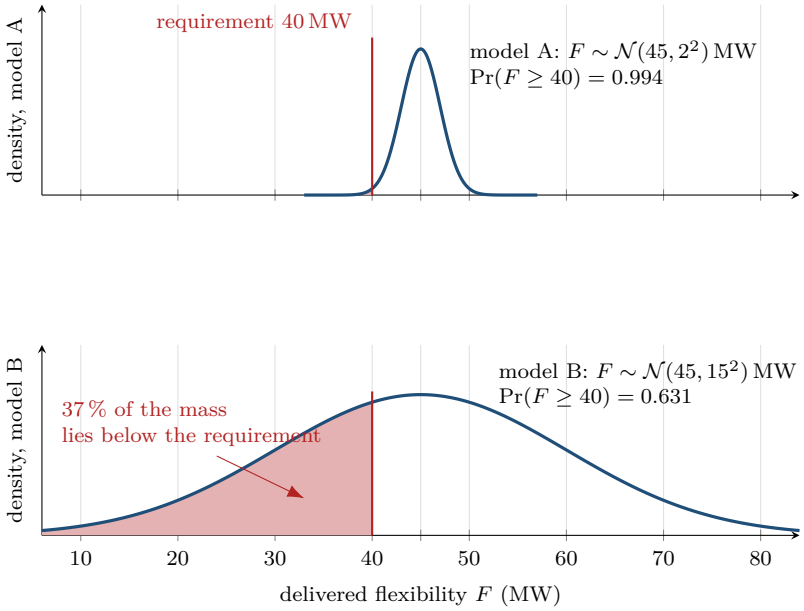


Figure 10.1: Two forecasts with the same mean and different spread. Both densities are normal with mean 45 MW; the upper has $\sigma = 2$ MW and the lower $\sigma = 15$ MW. The vertical scales differ between panels. The red line is a 40 MW requirement and the shaded areas are the shortfall probabilities of equation (10.2): 0.6 % for model A and 36.9 % for model B.

10.1.2 Point forecast and distributional forecast

The two outputs can be stated side by side. A point forecast is

$$\hat{y}_{t+h} = 45\,\text{MW}, \tag{10.3}$$

a single best guess. A probabilistic forecast is the function $p(y_{t+h} \mid \mathcal{I}_t)$, the whole range of plausible outcomes with their relative plausibility. In practice nobody transmits a density; the forecast is summarized by a handful of quantiles. A typical report is

$$\text{P10} = 35\,\text{MW}, \qquad \text{P50} = 45\,\text{MW}, \qquad \text{P90} = 55\,\text{MW}, \tag{10.4}$$

read as: there is a 10 % chance the outcome falls below 35 MW, the median outcome is 45 MW, and there is a 90 % chance it falls below 55 MW, equivalently a 10 % chance of exceeding 55 MW. Three numbers instead of one, and the extra two are what the exceedance calculation of equation (10.2) needs.

The choice of which quantiles to report is a design decision, not a statistical one. A commitment process that cares about underdelivery wants dense coverage of the lower tail, P05 and P10 and P20; a process that bids surplus capacity into a market wants the upper tail. A symmetric triple such as equation (10.4) is a default, not a requirement.

10.1.3 Quantiles

The formal definition is the one in the jargon box above. For a target Y with distribution function F, the τ-quantile is the value q_τ satisfying

$$\Pr(Y \le q_\tau) = \tau, \qquad 0 < \tau < 1, \tag{10.5}$$

so $q_{0.10}$ is P10, $q_{0.50}$ is P50, the median, and $q_{0.90}$ is P90. The forecast version carries a hat and is conditional on the information set: $\hat{y}_\tau$ is the model's estimate of the τ-quantile of $p(y_{t+h} \mid \mathcal{I}_t)$, and for flexibility it is written F_τ.

Quantiles have two properties that make them the natural report format. They are invariant under any increasing transformation of the target, so a quantile of a power is a quantile of the corresponding energy; and they are defined without reference to any distributional family, so reporting P10, P50 and P90 commits the forecaster to no assumption of normality. By construction they are ordered, $q_{\tau_1} \le q_{\tau_2}$ whenever $\tau_1 \le \tau_2$, a constraint that a model fitted separately at each level can violate; Section 10.6.6 returns to that failure.

10.1.4 Prediction intervals and what their coverage means

A pair of quantiles reported as a range is a prediction interval. The pair [P05, P95] has nominal coverage $0.95 - 0.05 = 0.90$ and is called a 90 % interval. With P05 = 30 MW and P95 = 60 MW the forecast is

$$F \in [30, 60]\,\text{MW} \quad \text{with nominal coverage } 90\,\%. \tag{10.6}$$

The meaning of that statement needs care. It is not the claim that the true value has a 90 % probability of lying inside this particular fixed interval; in a strict frequentist reading the true value either is or is not inside it, and the probability is not a property of the realized pair of numbers. The operational reading is a frequency statement across forecasts: if the forecaster issues many 90 % intervals under comparable conditions, about 90 % of the realized outcomes should fall inside their respective intervals. That property is calibration, and it is the subject of the next section. It is also, in contrast to the interval itself, something that can be measured from the historical record.

10.2 Calibration and sharpness

A distributional forecast is useful only if its stated probabilities mean what they claim and the ranges they imply are narrow enough to act on. This section defines the two properties, shows the two ways a forecast fails each, and states the tension between them.

10.2.1 Calibration

Calibration is checked by counting. Take the last thousand 90 % intervals the forecaster issued, look up what actually happened at each target time, and count how many outcomes fell inside their interval. If 897 of the 1000 outcomes fell inside, the observed coverage is 89.7 %, against a nominal 90 %, and the forecaster is well calibrated: its 90 % intervals behave like 90 % intervals. If only 650 fell inside, the observed coverage is 65 % and the forecaster is *overconfident*: it is issuing intervals that are too narrow for the uncertainty actually present, and every downstream decision that treats them as 90 % intervals is taking more risk than it believes.

The size of that error is easy to quantify under a normal model. A central 90 % interval spans $\pm 1.645\sigma$ and a central 65 % interval spans $\pm 0.935\sigma$, so a forecaster whose nominal 90 % intervals achieve 65 % coverage is understating the standard deviation by a factor of $1.645/0.935 = 1.76$. Its intervals would need to be roughly three quarters wider to be honest. Nothing about the point forecast is implicated: the center can be unbiased while the width is wrong by a factor of two.

10.2.2 Underconfidence

The failure in the other direction is less dangerous and still a failure. Suppose 990 of the 1000 outcomes fell inside the nominal 90 % intervals, an observed coverage of 99 %. The intervals are too wide. The forecaster is *underconfident*; in effect it is reporting that it knows very little, and a range wide enough to be right 99 % of the time when 90 % was requested is a range that excludes very few operational possibilities.

Underconfidence is not caught by any coverage test that only asks whether coverage is at least nominal, which is why coverage cannot be the sole criterion. A forecaster can achieve any coverage it likes by widening its intervals without limit, and the limiting case, reporting the full physical range of the target on every occasion, is perfectly calibrated at every nominal level and carries no information at all. The missing criterion is sharpness.

10.2.3 Sharpness

Sharpness is how concentrated the predictive distribution is: narrow intervals are sharp, wide intervals are not. It is a property of the forecast alone and can be computed without ever observing an outcome, which is precisely why it must be paired with calibration.

Two calibrated forecasters can differ enormously in sharpness. Take one whose 90 % interval for a given hour is $[40, 50]$ MW and another whose interval for the same hour is $[10, 80]$ MW. If both are calibrated, so that the first catches the outcome 90 % of the time within a 10 MW band and the second needs a 70 MW band to achieve the same thing, the first is far more useful: it supports a commitment near 40 MW, while the second supports almost nothing. The ordering matches the two densities of Section 10.1.1, whose exact central 90 % intervals are $45 \pm 1.645 \times 2 = [41.7, 48.3]$ MW and $45 \pm 1.645 \times 15 = [20.3, 69.7]$ MW. The round figures $[40, 50]$ and $[10, 80]$ are illustrative; the ratio of widths is the point.

10.2.4 The tension between calibration and sharpness

The two criteria pull in opposite directions, and each alone can be satisfied trivially. Maximizing coverage drives intervals toward uselessness: an interval wide enough to contain every conceivable outcome has perfect coverage and no information. Maximizing sharpness drives intervals toward dishonesty: a very narrow interval looks decisive and undercovers, with the consequences of Section 10.2.1. The objective is the joint one.

Principle 10.1. *The target of a probabilistic forecaster is calibration first and sharpness second: correct empirical coverage, with intervals as narrow as the real uncertainty allows. Neither property is meaningful in isolation, and the tension between them is fundamental rather than an artifact of any particular method.*

The joint objective is also what makes the proper scoring rules of Section 10.5.1 necessary. A single number that rewards calibration and sharpness simultaneously, and is minimized only by the true conditional distribution, is the only way to compare two forecasters that trade one property against the other.

10.3 Learning a predictive distribution

Two routes lead from a training set to a predictive distribution: fit the quantiles directly, or fit the parameters of a distributional family. This section develops both, with the loss function that makes the first work, and then examines the two features of real energy targets that force the uncertainty to be conditional: dependence on weather regime and dependence on horizon.

10.3.1 Quantile regression and the pinball loss

The direct route predicts a chosen quantile by training against a loss whose minimizer is that quantile. The loss is the *pinball* or quantile loss,

$$L_\tau(y, \hat{y}_\tau) = \begin{cases} \tau\,(y - \hat{y}_\tau), & y \geq \hat{y}_\tau, \\ (1-\tau)\,(\hat{y}_\tau - y), & y < \hat{y}_\tau, \end{cases} \tag{10.7}$$

two straight lines meeting at zero error, with the branch weights τ and $1-\tau$. Fitting any model class of Chapters 5 to 8 with equation (10.7) in place of squared error turns it into a predictor of the conditional τ-quantile; three fits at $\tau = 0.1, 0.5, 0.9$ give the triple of equation (10.4).

The asymmetry is what does the work. Consider estimating P90, so $\tau = 0.9$. If the actual outcome lands above the forecast, the forecaster underpredicted the 90th percentile and pays at weight 0.9. If it lands below, it pays at weight 0.1. Underprediction is nine times as expensive as overprediction, so the fitted value is pushed upward until the two expected penalties balance, which happens when about 90 % of observations lie below it. The argument is exact. Writing F for the conditional distribution of the target and differentiating the expected loss with respect to the forecast q,

$$\frac{\mathrm{d}}{\mathrm{d}q}\,\mathbb{E}[L_\tau(Y, q)] = -\tau\,[1 - F(q)] + (1-\tau)\,F(q) = 0 \quad \Longrightarrow \quad F(q) = \tau, \tag{10.8}$$

so the minimizer is the τ-quantile of equation (10.5) and nothing else. Figure 10.2 plots the three losses. The horizontal axis is the forecast error $e = y - \hat{y}_\tau$, positive when the outcome exceeds the forecast, and the vertical axis is the loss in the units of the target; each curve is the V-shaped function of equation (10.7) for one level. The $\tau = 0.5$ curve is symmetric. The $\tau = 0.9$ curve is steep to the right of zero and nearly flat to the left, which is the statement that overshooting P90 costs little and undershooting it costs a great deal; the $\tau = 0.1$ curve is its mirror image.

10.3.2 Median regression

The level $\tau = 0.5$ is the symmetric special case. Both branch weights of equation (10.7) equal one half, so the loss reduces to $\frac{1}{2}|y - \hat{y}|$ and the fitted value is the conditional median, P50. Up to the constant factor this is the mean absolute error of Chapter 9, which makes a connection worth stating explicitly: training against absolute error does not estimate the conditional mean, it

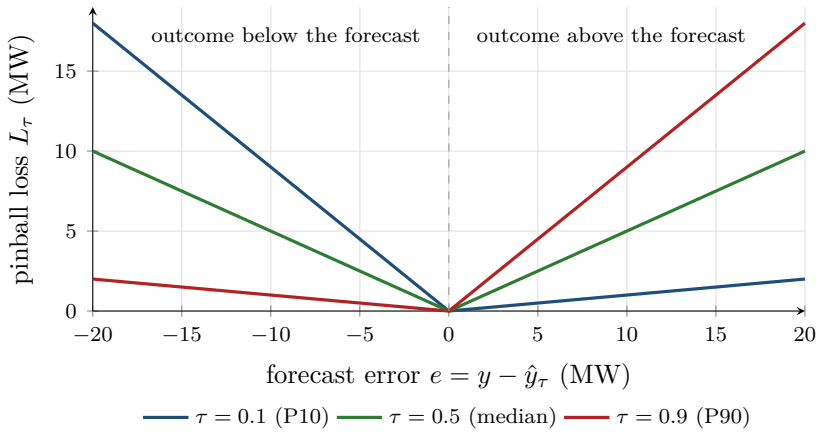


Figure 10.2: The pinball loss of equation (10.7) at three quantile levels, plotted against the forecast error $e = y - \hat{y}_\tau$ in MW. Each curve is $\tau\, e$ for $e \geq 0$ and $-(1-\tau)\, e$ for $e < 0$. The asymmetry of the outer two curves is what drives the fitted value to the corresponding quantile by equation (10.8).

estimates the conditional median. Training against squared error estimates the conditional mean. The loss function chooses the functional, and a forecaster who switches from root-mean-square error to mean absolute error has quietly changed the quantity being predicted.

10.3.3 Mean and median are different numbers

For a symmetric predictive distribution the distinction is immaterial: mean and median coincide, and $\mathbb{E}[F] \approx \text{P50}$. Real flexibility distributions are not symmetric. A thermostat fleet has a long *downside* tail, because the mechanisms that reduce delivered response, devices dropping off the network, customers overriding, a batch of controllers failing to acknowledge, are one-sided: they can subtract a large amount of response and cannot add any. A handful of such days pulls the mean well below the median while leaving the bulk of the distribution where it was.

The practical consequence is that "the forecast" is ambiguous until the functional is named. When a model reports 46 MW, the question to ask is whether 46 is the mean, the median or the mode of the predictive distribution, because under a skewed distribution those are three different numbers with three different operational meanings. The mean is the right summary for anything that will be averaged over many occasions, such as settlement revenue; the median is the right summary for the question "will this hour be above or below the figure"; neither is the right summary for a reliability commitment, which is the subject of Section 10.3.4.

10.3.4 The downside tail and conservative estimates

Suppose a forecast reports P50 = 50 MW and P10 = 30 MW. The median is comfortable and the lower tail is not: on one occasion in ten the fleet delivers less than 30 MW, which is 40 % below the median. An operator who commits 50 MW because the median says 50 MW will underdeliver frequently, and the frequency is readable directly off the quantiles.

A dispatch optimizer handed this distribution does not use the median. It uses a lower quantile as a conservative estimate of what can be promised, for example $F_{0.10} = 30$ MW, and commits that. The capacity it gives up relative to the median, 20 MW, is the price of reliability, and the width of the lower tail sets that price. This is the first place in the chapter where the distribution does something a point forecast cannot: it converts a reliability requirement into a megawatt number. Section 10.6.2 makes the conversion precise.

10.3.5 A parametric predictive distribution

The second route assumes a distributional family and predicts its parameters. Assuming the target is conditionally normal,

$$y_{t+h} \mid X_t \sim \mathcal{N}\Big(\mu(X_t),\ \sigma(X_t)^2\Big)\,, \tag{10.9}$$

turns the learning problem into two regressions: one for the center $\mu(X_t)$ and one for the spread $\sigma(X_t)$. Both are functions of the same feature vector X_t built in Chapter 4, and both are fitted together by minimizing the negative log likelihood of equation (10.9) over the training set. The model now predicts not only the expected value but the uncertainty around it, and any quantile follows in closed form, $\hat{y}_\tau = \mu(X_t) + \Phi^{-1}(\tau)\,\sigma(X_t)$.

The parametric route buys compactness and sampling. Two numbers describe the whole predictive distribution, scenarios are generated by drawing from it directly, and the quantiles cannot cross because they are generated from one monotone formula. It pays for this with the assumption: if the true conditional distribution is skewed or bounded, as flexibility is, the normal family misrepresents the tails that matter most. Section 10.6.6 sets the two routes against each other.

10.3.6 Heteroskedastic uncertainty

The reason σ in equation (10.9) is written as a function of X_t rather than as a constant is that forecast uncertainty in this domain is strongly condition dependent. On a mild day a thermostat fleet might deliver $F \sim \mathcal{N}(20, 2^2)$ MW: little load is available to curtail, and the little that is available is predictable. In extreme heat the same fleet might deliver $F \sim \mathcal{N}(50, 10^2)$ MW: far more response is available, and far less certainty about how much. The conditional variance $\mathrm{Var}(Y \mid X)$ is not constant, a property called *heteroskedasticity*, in contrast to the constant-variance (homoskedastic) assumption behind a single reported error standard deviation.

Figure 10.3 shows the pattern. The horizontal axis is outdoor temperature, the vertical axis is delivered flexibility, each gray dot is one historical event drawn from the heteroskedastic model stated in the caption, the solid line is the conditional mean and the dashed lines are the conditional P10 and P90. Two things widen together from left to right: the center rises, because hot weather means more compressors running, and the scatter about the center grows, because hot weather also means more variable occupancy, more overrides and more building-to-building variation in how fast indoor temperature drifts. A model that reports one global error standard deviation describes neither end of this plot.

Temperature is only the most visible driver. The conditional spread also depends on fleet availability, since a smaller available population averages less; on telemetry quality, since an unobserved device contributes pure prior uncertainty; on the forecast horizon, treated next; on device composition, since a cohort of identical new controllers behaves more predictably than a mixed legacy fleet; and on recent control history, since a fleet dispatched an hour ago is in a thermal state the model has seen less often. All of these belong in $\sigma(X_t)$.

10.3.7 Uncertainty grows with horizon

The horizon is a special case of the same point and deserves its own treatment because it is the dimension operators ask about most. A forecast issued now for fifteen minutes ahead might report P10 = 42, P50 = 45, P90 = 48 MW. The same forecaster for six hours ahead might report P10 = 30, P50 = 45, P90 = 60 MW. The center is unchanged; the 80 % range has grown from 6 MW to 30 MW. The distribution $p(y_{t+h} \mid \mathcal{I}_t)$ is a different object for each h, and a forecasting system must produce one per horizon.

Figure 10.4 is the usual way to display this. The horizontal axis is the horizon h in hours and the vertical axis is flexibility; the solid central line is P50, the two bounding curves are P10

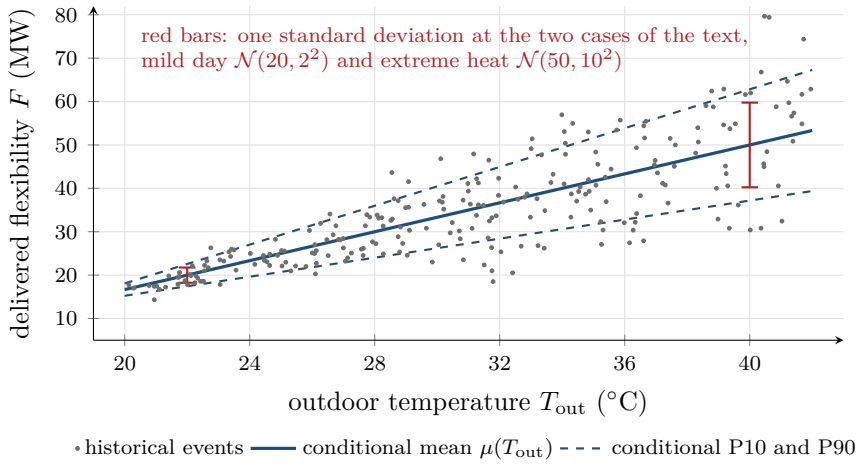


Figure 10.3: Heteroskedastic forecast uncertainty. The 240 points are a seeded sample from $F \mid T_{\text{out}} \sim \mathcal{N}(\mu(T_{\text{out}}), \sigma(T_{\text{out}})^2)$ with $\mu(T_{\text{out}}) = 20 + (30/18)(T_{\text{out}} - 22)$ and $\sigma(T_{\text{out}}) = 2 + (8/18)(T_{\text{out}} - 22)$, both in MW, generated by `scripts/ch10_data.py`; the dashed curves are $\mu \pm 1.2816\,\sigma$, the conditional P10 and P90. The red bars mark the one-standard-deviation spread at the mild and extreme cases of Section 10.3.6. Values are illustrative.

and P90, and the shaded region between them is the central 80 % band. The band is drawn from the square-root growth law stated in the caption, which is the behavior a random walk of independent per-step disturbances produces and a reasonable first description of what real systems do over a few hours. The two marked horizons are the 15-minute and 6-hour cases of the previous paragraph. The shape of the figure is the operational message: a commitment made six hours ahead must be sized against a much wider lower tail than the same commitment made fifteen minutes ahead, which is why intraday revision, and not only day-ahead accuracy, is worth paying for.

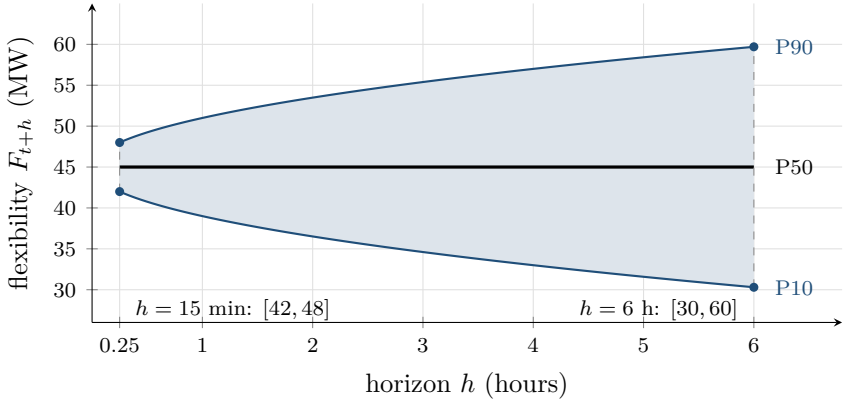


Figure 10.4: A fan chart: the central 80 % band of $p(F_{t+h} \mid \mathcal{I}_t)$ against horizon. The curves are P50 $= 45$ MW and P10, P90 $= 45 \mp 6\sqrt{h}$ MW with h in hours, a square-root growth law chosen so that the band is $[42, 48]$ MW at fifteen minutes and $[30.3, 59.7]$ MW at six hours, the two cases of Section 10.3.7. Values are illustrative.

10.4 Sources of uncertainty and their propagation

The width of the band in Figure 10.4 is not a single quantity to be estimated but the accumulation of several physically distinct uncertainties. This section lists them, follows one of them through

the response model, and gives the generic computational recipe for pushing any of them through to the output.

10.4.1 Six sources

Table 10.1 lists the sources that contribute to $p(F \mid \mathcal{I}_t, u)$, with what varies in each and whether more data reduces it. The last column anticipates the aleatoric and epistemic split of Section 10.5.4. Reading the table from the top: the weather is not known at issue time; occupants do not behave identically from day to day; devices disconnect, lose network or opt out, so the controllable population itself is uncertain; the model's structure and parameters are estimates; telemetry is noisy, delayed and incomplete, so even the present state is uncertain; and the same command issued twice to the same fleet does not produce the same response.

Table 10.1: Sources of uncertainty in a flexibility forecast.

Source	What varies	Reducible by more data?
Weather	future temperature, humidity, irradiance	partly, through a better weather forecast
Behavior	occupancy, setpoints, override decisions	largely irreducible
Availability	devices disconnected, unenrolled, opted out	partly, through communications and enrollment work
Model	structure, parameters, functional form	yes: this is estimation error
Observation	noisy, delayed, missing telemetry	partly, through metering
Control response	the same command yields different responses	largely irreducible

The table is the reason a single calibration constant cannot fix a probabilistic forecast. The six rows have different magnitudes under different conditions, they grow at different rates with horizon, and three of them are reducible by engineering work while three are not. A forecaster who can attribute the width of its interval to these rows can say what would have to change to narrow it.

10.4.2 Propagation of weather uncertainty

The weather row is worth following in detail because it is the largest contributor at horizons beyond an hour or two and because it illustrates the general mechanism. Suppose flexibility is a deterministic function of temperature, $F = f(T)$, and the future temperature is itself uncertain with distribution $p(T \mid \mathcal{I}_t)$. Then F is a random variable whether or not f is known exactly: the uncertainty in T propagates through f to the output. The chain is

$$p(T_{t+h} \mid \mathcal{I}_t) \implies p(T_{\text{in}}, \text{compressor state}) \implies p(F_{t+h} \mid \mathcal{I}_t), \tag{10.10}$$

weather uncertainty to future indoor conditions to available flexibility. The practical conclusion is uncomfortable and important: a perfect model of how the fleet responds to temperature removes none of this uncertainty. Improving f narrows the band only to the extent that model error contributed to it; the weather term is bounded below by the quality of the weather forecast, which the aggregator does not control.

10.4.3 Monte Carlo propagation

Equation (10.10) is rarely available in closed form, because f is a trained model or a simulation rather than an algebraic expression. The generic recipe is sampling. For $m = 1, \dots, M$,

$$T^{(m)} \sim p(T \mid \mathcal{I}_t), \qquad F^{(m)} = f\Big(T^{(m)}, X_t, u\Big), \tag{10.11}$$

and the collection $\{F^{(1)}, \dots, F^{(M)}\}$ is a sample from $p(F \mid \mathcal{I}_t, u)$ whose empirical quantiles estimate P10, P50 and P90 to a precision that improves as M grows. Nothing in equation (10.11) requires f to be differentiable, linear or even deterministic; if the response model has its own noise, drawing from it inside the loop includes the control-response row of Table 10.1 in the same sample.

Two practical points govern whether the sample is honest. The input distribution must be the real one: sampling temperature from a symmetric band around a deterministic weather forecast understates the tails that an ensemble weather forecast would reveal. And the draws must preserve dependence; Section 10.6.3 returns to this for the multi-horizon case, where sampling each hour's temperature independently produces trajectories no real day resembles.

10.4.4 Uncertainty is correlated across a fleet

A fleet of 100 000 thermostats might appear to be a setting where errors average away: with independent device-level errors, the relative uncertainty of the aggregate falls as $1/\sqrt{N}$ and is negligible at that scale. Device errors are not independent. The devices share a weather field, so a temperature forecast that is two degrees too low is two degrees too low for all of them at once. They share grid conditions, firmware versions, controller logic, tariff signals and regional behavior patterns. Every one of these is a common driver, and a common driver produces an error component that aggregation does not reduce at all.

The consequence is that fleet uncertainty is not the sum of independent device uncertainties, and treating it as such produces a forecast that is overconfident by a large factor at exactly fleet scale. The quantitative version of this statement, and what it implies for how a fleet forecast should be built, is the subject of Chapter 11.

10.5 Evaluating probabilistic forecasts

Point metrics cannot detect any of the failures of Section 10.2, so the evaluation apparatus of Chapter 9 needs extending. This section builds the diagnostics: level-by-level calibration, the reliability diagram, calibration conditional on regime, the split between irreducible and reducible uncertainty, and the two common ways of producing intervals without a distributional model.

10.5.1 Scoring and quantile calibration

For point forecasts the metrics were mean absolute error and root-mean-square error. For probabilistic forecasts there are three things to measure: calibration, sharpness, and overall distributional quality. The first two were defined in Section 10.2. The third needs a score, and for a set of predicted quantiles the natural one is the pinball loss of equation (10.7) averaged over levels and over the evaluation window, which is minimized only by the true conditional quantiles and therefore rewards calibration and sharpness together.

Calibration itself is checked level by level, not only for intervals. Take every occasion on which the model predicted P90 and count the outcomes at or below it. Over a thousand such forecasts, about 900 outcomes should satisfy $y \le$ P90. If 990 do, the predicted quantile is systematically too high: it is behaving like a P99. If 700 do, it is too low, behaving like a P70.

The test applies to every quantile the model emits, and running it at one level only, typically the 90 % interval, hides the case of a distribution whose center is calibrated and whose tails are not.

10.5.2 The reliability diagram

Collecting the level-by-level counts into one picture gives the reliability diagram of Figure 10.5. The horizontal axis is the nominal quantile level τ the model claimed, from 0.05 to 0.95; the vertical axis is the observed frequency with which outcomes fell at or below the predicted τ-quantile; the dashed diagonal is perfect calibration, where the stated probabilities mean what they claim. Reading a curve is mechanical. A curve on the diagonal is calibrated. A curve that is above the diagonal at low levels and below it at high levels, as the red curve is, belongs to a forecaster whose quantiles are pulled toward the center: its low quantiles are too high and its high quantiles too low, so its intervals are too narrow and it is overconfident. The mirror-image shape belongs to an underconfident forecaster.

The diagram also reads off coverage. The vertical distance between the curve's values at $\tau = 0.95$ and $\tau = 0.05$ is the observed coverage of the nominal 90 % interval. For the calibrated curve that distance is $0.945 - 0.060 = 0.885$, close to nominal. For the overconfident curve it is $0.795 - 0.165 = 0.630$: a nominal 90 % interval that covers 63 % of outcomes.

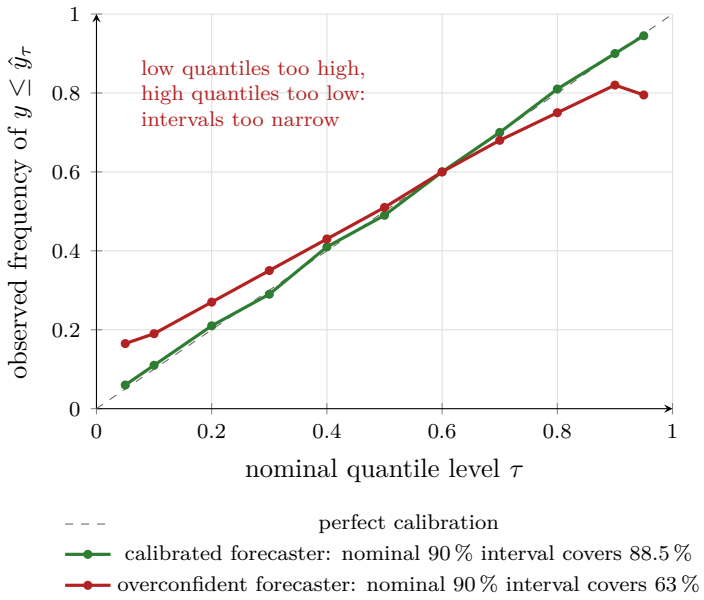


Figure 10.5: A reliability diagram for predicted quantiles. Each point compares a nominal level τ with the fraction of outcomes that fell at or below the predicted τ-quantile over the evaluation window. The dashed diagonal is perfect calibration. The red curve's values at $\tau = 0.05$ and 0.95 differ by 0.63, the coverage of its nominal 90 % interval. Values are illustrative and follow Section 10.5.2.

Note that the red curve in Figure 10.5 is not monotone at the top: its value at $\tau = 0.95$ is lower than at $\tau = 0.9$. That is a symptom of quantile crossing, discussed in Section 10.6.6, and a diagram is one of the places it becomes visible.

10.5.3 Conditional calibration and interval width by regime

Aggregate calibration can be excellent and conceal a dangerous failure. A forecaster might achieve 89.7 % coverage over a full year of 90 % intervals and, on the subset of days with

$T_{\text{out}} > 38°\text{C}$, cover only 65 %. Those are exactly the days on which the grid needs the forecast, so the aggregate figure is not merely incomplete, it is misleading. The arithmetic makes the concealment easy: if extreme-heat days are 4 % of the record, a drop in coverage from 91 % to 65 % on those days moves the annual figure by only one percentage point.

Calibration must therefore be evaluated conditionally, slicing the evaluation window by every variable along which the problem's difficulty changes: horizon, temperature or weather regime, geography, event type, fleet availability and telemetry quality. The property to aim for is local reliability, not global reliability. A globally calibrated and locally unreliable forecaster is a forecaster that is wrong when it matters.

Interval width must also vary. The companion diagnostic is to look at the widths rather than the coverages. A model that outputs ± 10 MW on every day of the year has learned a global error magnitude and nothing about condition-dependent uncertainty; it will undercover in hard conditions and overcover in easy ones, possibly achieving acceptable aggregate coverage while being wrong in both regimes. The behavior to want is the one Figure 10.3 displays: an interval of roughly $[43, 47]$ MW on an easy day and $[30, 60]$ MW on a hard one. A good probabilistic model knows when the problem is harder, and its interval width is where that knowledge appears.

10.5.4 Aleatoric and epistemic uncertainty

The rows of Table 10.1 divide into two kinds. *Aleatoric* uncertainty is inherent randomness in the system: occupant behavior, random device availability, the irreducible variation in how a fleet responds to an identical command. It does not shrink as the training set grows, because it is a property of the world rather than of the model's ignorance. *Epistemic* uncertainty is what the model does not know: regions of feature space with few observations, imprecisely estimated parameters, an uncertain functional form. It does shrink with more information, either more data or better structure.

The distinction matters because the two call for different responses. Aleatoric uncertainty must be represented honestly and planned around with reserves; epistemic uncertainty is a target for data collection, instrumentation or model work. A concrete case makes both visible. Suppose the training record covers 20 to 35 °C with almost no observations above 40, and tomorrow's forecast is 42 °C. Part of the resulting uncertainty is aleatoric: customers behave differently in unprecedented heat, and that variation would persist however much data existed. Part is epistemic: the model is extrapolating into a region it has barely observed, and more data from hot days would reduce it. Standard quantile regression does not separate the two. It fits the quantiles of the training distribution and reports them; it has no mechanism for saying "this input is unlike anything I was trained on".

10.5.5 Precision without knowledge

The practical consequence of that gap is worth stating as its own failure mode. A gradient-boosted quantile model of Chapter 5 fitted on the record just described, and asked for a forecast at 42 °C, may return P10 = 48, P50 = 50, P90 = 52 MW: a very narrow interval, apparently a confident forecast. The narrowness is an artifact. Tree ensembles extrapolate by holding the prediction of the nearest observed region constant, so outside the training domain the model reports the spread it learned inside the domain, which has no bearing on the spread that applies outside it. The forecast is precise and wrong.

Principle 10.2. *A narrow prediction interval is not evidence that the model knows. It is a statement about the dispersion the model learned on its training distribution, and it carries no information about inputs that distribution did not contain. Interval width must always be read together with a check on whether the input lies inside the training domain.*

This is the mechanism behind distribution shift, where the relationship between features and target changes after training; a later chapter treats detection and mitigation. For this chapter the point is narrower: no quantile model can be trusted to signal its own extrapolation, so the signal has to be computed separately and reported alongside the interval.

10.5.6 Ensembles as a signal of model uncertainty

One way to get at the epistemic component is to train several models $f_1, \dots, f_M$, differing in random seed, resampled training data, hyperparameters or structure, and inspect their disagreement at the input in question. If the four members predict $[44.5, 45.0, 45.2, 44.8]$ MW, the models agree and the disagreement contributes little uncertainty. If they predict $[30, 42, 55, 65]$ MW, the choice of model is itself a major source of uncertainty at this input, which is a signal worth surfacing even though no single member reports it.

Ensemble spread is a useful diagnostic and not a calibrated predictive uncertainty. The members share a training set, a feature set and usually a model family, so their disagreement measures variation within that shared setup and omits everything common to all of them; a systematically missing feature produces agreement, not disagreement. Empirically, ensemble spread tends to understate true predictive uncertainty. It must therefore be validated against outcomes and rescaled like any other uncertainty estimate, using the reliability diagram of Figure 10.5 on the ensemble's own intervals.

10.5.7 Conformal prediction

The last construction produces intervals from a point model without touching the model. Hold out a calibration set the model never trained on, compute the absolute errors $|e_1|, \dots, |e_N|$ on it, and take a high empirical quantile q of those errors, for instance the one at level $\lceil (N+1)(1-\alpha) \rceil / N$ for a nominal $1-\alpha$ interval. The prediction interval is then

$$[\, \hat{y}_{t+h} - q,\ \hat{y}_{t+h} + q \,], \tag{10.12}$$

and under an exchangeability assumption between the calibration data and the future this interval has a finite-sample coverage guarantee: the long-run coverage is at least $1-\alpha$ regardless of the model or the data distribution. Adaptive variants let q scale with a predicted difficulty, recovering the regime dependence of Section 10.5.3, and asymmetric variants use separate quantiles of the positive and negative errors to handle skew.

The appeal is operational. The construction wraps an existing model: gradient boosting of Chapter 5 supplies the point forecast, conformal calibration supplies the interval, and the production model needs no redesign, no new loss function and no distributional assumption. For a team with a tuned point forecaster already in service, this is the cheapest route to a defensible interval.

The assumption is where it ends. Exchangeability requires the calibration data and the forecast conditions to be drawn from the same distribution, and that is exactly what fails in the case that matters: if the calibration period was mild and the forecast day is an unprecedented heat wave, the errors observed on the calibration set are not a sample of the errors that will occur, and the coverage guarantee does not hold. The guarantee is conditional on a premise the operator cannot verify in advance. No uncertainty-quantification method removes the need to reason explicitly about changing conditions.

10.6 From distribution to decision

A distribution is only worth producing if a decision consumes it. This section closes the loop: the commitment question the distribution answers, the way an asymmetric cost selects a quantile, the joint structure a multi-hour commitment requires, the reporting that exposes conditional miscalibration, and the choices of output format and model family that follow.

10.6.1 A commitment decision

At 2 PM an aggregator forecasts the flexibility available at 5 PM and reports

$$\mathrm{P10} = 35\,\mathrm{MW}, \qquad \mathrm{P50} = 46\,\mathrm{MW}, \qquad \mathrm{P90} = 58\,\mathrm{MW}. \tag{10.13}$$

The operational question is whether 40 MW can safely be committed. A point forecast answers immediately and badly: 46 exceeds 40, so yes. The distributional answer is a number with a different meaning,

$$\Pr(F \geq 40\,\mathrm{MW}) = 0.78, \tag{10.14}$$

a 22 % chance of underdelivery. Whether 22 % is acceptable is not a forecasting question. It depends on the penalty for underdelivery, on whether reserve capacity can be called, and on the operator's risk tolerance, all of which live outside the forecasting system. The forecast does not make the decision; it supplies the quantity the decision needs.

Figure 10.6 shows the calculation. The horizontal axis is flexibility at 5 PM and the curve is the predictive density implied by equation (10.13); the three dashed verticals mark P10, P50 and P90; the vertical red line is the 40 MW commitment, with the region to its right shaded green as the probability of delivering and the region to its left shaded red as the probability of falling short. The red area is the 22 % that the point forecast concealed. The same picture with the commitment moved to 30 MW or 50 MW would show the exceedance probability rising or falling, which is what makes the distribution, rather than any single one of its summaries, the right deliverable.

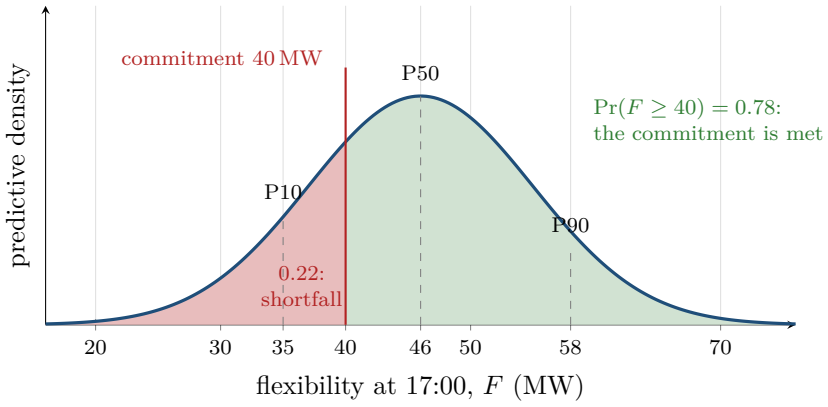


Figure 10.6: The commitment decision of equations (10.13) and (10.14). The density drawn is the normal fit to the reported quantiles, $\mathcal{N}(46, 8.97^2)$ MW with $\sigma = (58 - 35)/(2 \times 1.2816)$; its own exceedance probability at 40 MW is $\Phi(0.669) = 0.75$, against the 0.78 the model's slightly right-skewed distribution reports. Values are illustrative.

Conservative capacity is a convention, not a law. A common rule is to commit the tenth percentile, $F_{\mathrm{safe}} = \mathrm{P10} = 35\,\mathrm{MW}$, on the grounds that it will be delivered nine times in ten. The rule is defensible and the particular level is arbitrary. The right percentile follows from the penalty for underdelivery, the availability and cost of reserves, the market's rules

on non-performance and the customer comfort constraints that cap how hard the fleet can be pushed. There is nothing special about P10, and an operator who uses it without having checked those four inputs has chosen a risk level by habit.

10.6.2 Asymmetric loss and the operational quantile

The connection between the cost structure and the percentile is exact. Suppose the operational loss is piecewise linear with different weights for the two error directions,

$$L(\hat{F}, F) = \begin{cases} \lambda_{\text{over}}\,(\hat{F} - F), & \hat{F} > F \quad \text{(committed too much)}, \\ \lambda_{\text{under}}\,(F - \hat{F}), & \hat{F} < F \quad \text{(committed too little)}. \end{cases} \tag{10.15}$$

This is the pinball loss of equation (10.7) up to a scale factor, so by equation (10.8) its expected value is minimized at a quantile, not at the mean,

$$\tau^* = \frac{\lambda_{\text{under}}}{\lambda_{\text{under}} + \lambda_{\text{over}}}. \tag{10.16}$$

If overprediction is costlier, $\tau^* < 0.5$ and the optimal single number to report is below the median. Equation (10.16) is the bridge between probability and decision: an operational cost asymmetry maps to a forecast quantile.

A worked case. Suppose overcommitting costs four times as much as undercommitting, $\lambda_{\text{over}} = 4$ and $\lambda_{\text{under}} = 1$. Then $\tau^* = 1/5 = 0.2$, and for the predictive distribution of equation (10.13), with the normal fit $\mathcal{N}(46, 8.97^2)$, the loss-minimizing commitment is $46 - 0.8416 \times 8.97 = 38.5\,\text{MW}$, not the 46 MW median and not the 35 MW tenth percentile. The precise level depends on the cost structure, and that is the general lesson: the best point forecast in the statistical sense is not always the right operational forecast, and only a distribution lets the operator select the quantile its own costs call for.

10.6.3 Joint distributions across horizons, and scenarios

Most commitments span more than one hour, which changes the object required. A four-hour block starting at 17:00 needs

$$p(F_{17}, F_{18}, F_{19}, F_{20} \mid \mathcal{I}_t)\,, \tag{10.17}$$

the joint distribution, which is strictly harder than four separate marginal distributions because the errors are correlated in time. If 5 PM comes in unexpectedly low, because the weather forecast was wrong or availability was lower than assumed, 6 PM is likely to be low for the same reason.

The marginals do not determine the risk. Take $F_{17} \sim \mathcal{N}(50, 5^2)$ and $F_{18} \sim \mathcal{N}(50, 5^2)$, identical in both of two cases, and consider a commitment of 45 MW in each of the two hours. If the two hours are uncorrelated, the probability of delivering in both is $\Phi(1)^2 = 0.841^2 = 0.708$. If they are correlated at $\text{Corr}(F_{17}, F_{18}) \approx 0.95$, the two outcomes almost always move together and the probability of delivering in both is close to $\Phi(1) = 0.841$. Same marginals, materially different risk, and only the joint distribution distinguishes them.

Principle 10.3. *Per-hour percentiles do not determine the risk of a multi-hour commitment. The forecast must carry the dependence structure across horizons, either as a covariance or as a set of trajectories.*

Scenarios. The practical way to carry dependence is to report trajectories rather than per-hour quantiles. Three scenarios for the block above might be

$$s_1 = [45, 43, 40, 38], \qquad s_2 = [52, 50, 47, 45], \qquad s_3 = [40, 37, 33, 31] \quad \text{MW}, \tag{10.18}$$

each a plausible path through the four hours, generated by the Monte Carlo recipe of equation (10.11) with correlated draws so that each path is internally consistent. Figure 10.7 plots them. The horizontal axis is the target hour and the vertical axis is flexibility; the three marked lines are the trajectories of equation (10.18), the shaded band behind them is the per-hour P10-to-P90 range, and the horizontal red line is a 40 MW commitment held across all four hours. Each trajectory declines over the block, because a fleet held in curtailment drifts toward its comfort limits, and the three differ in level. Against the 40 MW commitment, scenario s_2 delivers in all four hours, s_1 falls short in the final hour, and s_3 falls short in three of four. An optimizer evaluates its candidate decision on each trajectory and aggregates the results; that procedure respects the time dependence, which independent per-hour percentiles destroy.

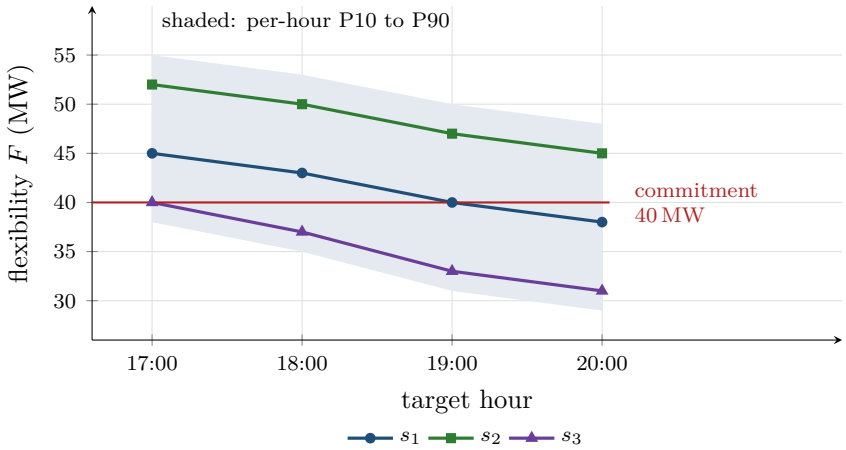


Figure 10.7: Three scenario trajectories from equation (10.18) over a four-hour block, against the per-hour P10-to-P90 band and a 40 MW commitment held across the block. Each trajectory is internally consistent in time, which per-hour percentiles alone are not. Values are illustrative.

What the optimizer does with it. Downstream, the distribution enters an optimization. With a decision x and an uncertain response $R(x)$, the two standard formulations are the expected-cost problem and the chance-constrained problem,

$$\min_x \ \mathbb{E}[C(x, R)] \qquad \text{subject to} \qquad \Pr(R(x) \geq R_{\text{target}}) \geq 0.95, \tag{10.19}$$

and the scenarios of equation (10.18) are precisely what makes the expectation and the probability computable. A later chapter develops this; what matters here is that the forecast's output format determines which formulations are available.

The cohort case. The same argument applies across cohorts rather than across time. Suppose a fleet is organized into three cohorts with responses $R_1 \sim p_1$, $R_2 \sim p_2$ and $R_3 \sim p_3$. The naive aggregate adds the expectations, $\mathbb{E}[R_1] + \mathbb{E}[R_2] + \mathbb{E}[R_3]$, which is the correct mean and says nothing about the risk. Risk-aware dispatch needs the distribution of the sum, $p(R_1 + R_2 + R_3)$, and that distribution depends on the dependence between the cohorts. They share a weather forecast, so the shared weather error correlates them positively; ignoring the correlation and adding variances as if independent produces a fleet forecast that is overconfident exactly where overconfidence is most expensive. This is the fleet-level version of Section 10.4.4 and the starting point of Chapter 11.

10.6.4 Coverage reported by horizon and by regime

Section 10.5.3 argued that calibration must be conditional. This subsection is the report that implements it, because a diagnostic nobody tabulates is a diagnostic nobody acts on.

Start with the aggregate counting. Over 100 events with nominal 90 % intervals, 74 outcomes inside is *underdispersed*: the intervals are too narrow, and the forecaster is overconfident. 99 inside is likely *overdispersed*: the intervals are too wide. The target is an observed coverage close to nominal with intervals as sharp as that constraint permits, which is the joint objective of Section 10.2.4 expressed as a count.

Table 10.2 slices the same count two ways, and Figure 10.8 plots it. In the figure, the vertical axis is observed coverage of a nominal 90 % interval and the dashed horizontal line is the 90 % nominal level; the left group of four bars is sliced by horizon and the right group of three by weather regime, separated by a vertical rule. The horizon slice degrades monotonically: 91 % at fifteen minutes, 89 % at one hour, 85 % at three hours and 72 % at six hours. The forecaster is calibrated at short horizons and increasingly overconfident at long ones, which means its intervals are not widening fast enough with h relative to the true growth in Figure 10.4. No point metric exposes this: root-mean-square error also grows with horizon, as it should, and gives no indication that the stated uncertainty failed to grow with it.

The regime slice is worse. Mild days cover 92 %, hot days 88 %, and extreme-heat days 63 %. The forecaster is dangerously overconfident in precisely the conditions under which the grid most needs the flexibility to materialize, and the overall annual figure, dominated by mild and hot days, looks acceptable. Two remedies follow directly from the table: widen the intervals as a function of horizon, and introduce an extreme-heat indicator into the model of $\sigma(X_t)$ so the width responds to the regime rather than to the annual average.

Table 10.2: Observed coverage of a nominal 90 % interval, sliced by horizon and by weather regime. Values are illustrative and follow Section 10.6.4.

Slice	Subset	Coverage	Reading
Horizon	15 min	91 %	calibrated
	1 h	89 %	calibrated
	3 h	85 %	mildly overconfident
	6 h	72 %	intervals grow too slowly with h
Regime	mild	92 %	calibrated, slightly wide
	hot	88 %	calibrated
	extreme heat	63 %	overconfident when it matters most

10.6.5 The forecast product

Collecting the chapter's output requirements gives the report a forecasting system should emit for each target hour. Table 10.3 is a template for the 17:00 target of Section 10.6.1, re-issued later in the afternoon with slightly more information, which is why its exceedance probability is 82 % rather than the 78 % of equation (10.14). The template carries five kinds of field: the target, a central estimate with its functional named, a set of quantiles, the exceedance probability at the threshold the operator cares about, and a named list of the dominant uncertainty sources from Table 10.1. The last field is what lets an operator distinguish "the band is wide because the weather forecast is uncertain" from "the band is wide because half the fleet is offline", which call for different responses.

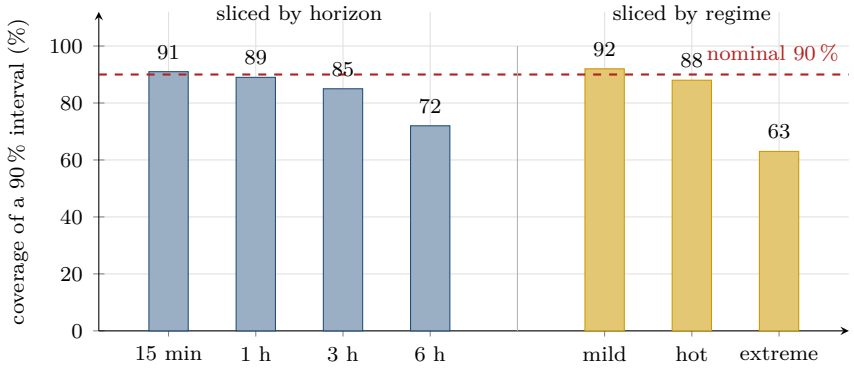


Figure 10.8: Conditional calibration of the same forecaster, sliced by horizon (left, blue) and by weather regime (right, gold). Bars are the observed coverage of a nominal 90 % interval; the dashed line is the nominal level. The numbers are those of Table 10.2 and are illustrative.

Table 10.3: A probabilistic forecast as a report for one target hour. Values are illustrative and follow Section 10.6.5.

Field	Value	Why it is reported
Target hour	17:00	fixes issue time and horizon
Expected flexibility	47 MW	the mean, for quantities that are averaged
P10	35 MW	conservative commitment level
P50	46 MW	the median; below the mean, so the distribution is right-skewed
P90	59 MW	surplus available to bid
$\Pr(F \geq 40\,\text{MW})$	82 %	the number the commitment decision uses
Dominant uncertainty	weather, device availability, response variability	says what would narrow the band

Uncertainty must be actionable. A report of $[20, 80]$ MW is not useful merely because it is honest. An interval that spans most of the physical range supports no decision, and a forecasting team that treats honesty as the whole objective has delivered the underconfidence of Section 10.2.2. The test of a probabilistic forecast is whether it answers the operator's questions: how much can be committed safely, how much reserve to hold behind it, whether to call an additional cohort, whether to wait for more telemetry before committing, and when to revise. A distribution that cannot change any of those answers has not earned the extra columns.

The chapter's arguments recur as conceptual questions, and the three the lesson raised are worth stating with the shape of a good answer.

Conceptual questions

"Why probabilistic forecasts if point RMSE is good?" A strong answer begins with the example of Section 10.1.1: two forecasts can share a point estimate, and therefore a root-mean-square error, while implying completely different operational risk. It then names what dispatch actually needs, the probability of underdelivery, and notes that the need is acute precisely under unusual weather or uncertain availability, where the spread is large and condition dependent. It closes on the mechanism: a calibrated distribution is what lets an optimizer trade committed capacity against risk, which no point forecast can support. A strong answer connects this to the commitment arithmetic of equations (10.13) and (10.14).

Conceptual questions

"What does it mean for a 90 % interval to be calibrated?" The answer is the frequency statement of Section 10.1.4: over many comparable forecasts, about 90 % of outcomes fall inside their respective intervals, not that the truth has 90 % probability of lying in one realized interval. A strong answer adds the conditional requirement of Section 10.5.3, that coverage must be checked by horizon and by regime, because an aggregate figure near 90 % can hide severe overconfidence on the few days that matter, as Table 10.2 shows.

Conceptual questions

"Would you prefer a wider interval for better coverage?" Not automatically. The answer should state that coverage and sharpness are a joint objective, give the degenerate case of an interval wide enough to cover everything and carry no information, and then state the real target: correct empirical coverage with intervals as sharp as the underlying uncertainty permits. Widening is the right fix only when the diagnostic shows undercoverage, and then preferably by widening conditionally, where the undercoverage occurs, rather than globally.

10.6.6 Quantile outputs and parametric outputs

Section 10.3.1 and Section 10.3.5 gave two routes to a predictive distribution, and Table 10.4 sets them against each other. Quantile regression is flexible: it assumes no family, so it accommodates skew and bounds, and it predicts exactly the percentiles the operator asked for. Its characteristic failure is crossing. Because each level is fitted by a separate model, nothing enforces the ordering of equation (10.5), and an unconstrained fit can produce $\text{P90} < \text{P50}$ at some inputs, which is not a distribution at all. Remedies exist, among them fitting the levels jointly, rearranging the predicted quantiles into sorted order, or constraining the model, and all of them must be applied deliberately.

A parametric output is compact and coherent: two predicted numbers give every quantile, sampling for the scenarios of equation (10.18) is direct, and crossing is impossible by construction. The risk is the assumption. A normal predictive distribution imposed on a right-skewed, lower-bounded target will misstate the lower tail, which is the part of the distribution the commitment

decision of Section 10.6.1 depends on. Neither route dominates; the choice follows from whether the skew or the coherence matters more in the application.

Table 10.4: Quantile regression and a parametric predictive distribution compared.

	Quantile regression	Parametric distribution
Output	$\hat{y}_\tau$ at chosen levels	$\mu(X_t)$, $\sigma(X_t)$
Assumption	none about the shape	a family, e.g. $\mathcal{N}$
Loss	pinball, equation (10.7)	negative log likelihood
Sampling	needs interpolation between levels	direct
Coherence	quantiles can cross	coherent by construction
Failure mode	crossing unless constrained	wrong family, misstated tails
Suits	skewed or bounded targets, named percentiles	compact transport, fast scenario generation

10.6.7 Model families and the bigger objective

Nothing in this chapter selected a model class. The recurrent networks of Chapter 7 and the attention models of Chapter 8 become probabilistic forecasters by changing the output layer and the loss: three output units trained with the pinball loss at $\tau = 0.1, 0.5, 0.9$, or two output units trained as μ and σ of equation (10.9), or the parameters of a mixture. Probabilistic forecasting is a different prediction objective, not a separate model family, and probabilistic linear models, boosted trees, recurrent networks and transformers all accommodate it.

Gradient boosting of Chapter 5 is worth singling out because the change is so cheap. Boosting optimizes whatever differentiable-enough loss it is given, so fitting three models under equation (10.7) yields $\hat{q}_{0.1}(X)$, $\hat{q}_{0.5}(X)$ and $\hat{q}_{0.9}(X)$ with the same features, the same tuning procedure and the same engineering as the point model it replaces. For tabular energy data this is among the strongest practical options available and a standing reason boosted ensembles remain in production.

The chapter's arc is a change in what the forecasting system is asked to produce. It began at

$$\hat{y}_{t+h} = f(\mathcal{I}_t), \tag{10.20}$$

a function returning one number, and ends at $p(y_{t+h} \mid \mathcal{I}_t)$ and, for controllable distributed resources, $p(y_{t+h} \mid \mathcal{I}_t, u)$ of equation (10.1). The output is no longer an estimate of a future value but a statement of the current state of information about a future uncertain system response, and the decision-making apparatus downstream is what converts that statement into a commitment.

10.7 What to retain

Definition 10.1 (Probabilistic forecast)**.** A point forecast hides the uncertainty that the decision depends on; a probabilistic forecast reports the distribution, in practice as quantiles such as P10 = 35, P50 = 45, P90 = 58 MW, learned by quantile regression under the pinball loss or by predicting the parameters of a distribution. It is judged on calibration, whether a nominal 90 % interval covers about 90 % of outcomes, and on sharpness, how narrow it is while remaining calibrated. The deliverable is not "we expect 50 MW" but the distribution of likely response together with the risk of underdelivery.

Table 10.5 collects the chapter's objects with the subsection that defines each.

Table 10.5: The objects of a probabilistic forecast.

Object	Definition	Section
$p(F \mid \mathcal{I}_t, u)$	the forecast: a distribution over the target given the information and the planned action	10.1
q_τ, $\hat{y}_\tau$	the τ-quantile, $\Pr(Y \le q_\tau) = \tau$, and its forecast	10.1.3
nominal coverage	the probability a reported interval is meant to enclose	10.1.4
calibration	observed coverage matches nominal coverage	10.2.1
sharpness	how narrow the predictive distribution is	10.2.3
L_τ	pinball loss; its minimizer is the τ-quantile	10.3.1
$\mu(X_t)$, $\sigma(X_t)$	parameters of a predicted distributional family	10.3.5
heteroskedasticity	$\mathrm{Var}(Y \mid X)$ depends on X	10.3.6
Monte Carlo propagation	sampling inputs and pushing them through the response model	10.4.3
reliability diagram	nominal level against observed frequency	10.5.2
conditional calibration	coverage checked within slices of the record	10.5.3
aleatoric, epistemic	irreducible randomness versus reducible ignorance	10.5.4
conformal interval	$[\hat{y}_{t+h} - q, \hat{y}_{t+h} + q]$ from held-out errors	10.5.7
τ^*	$\lambda_{\text{under}}/(\lambda_{\text{under}} + \lambda_{\text{over}})$: the quantile an asymmetric cost selects	10.6.2
scenarios	trajectories preserving dependence across horizons	10.6.3

10.8 Exercises

Exercise 10.1. A model predicts $\hat{y}_{0.1} = 38$, $\hat{y}_{0.5} = 46$ and $\hat{y}_{0.9} = 57\,\mathrm{MW}$ for an hour whose outcome is $y = 52\,\mathrm{MW}$. Evaluate the pinball loss of equation (10.7) at each level. Then recompute all three losses for an outcome of $y = 33\,\mathrm{MW}$ and state, for each level, in which direction the fitted value would move if many outcomes looked like the second case.

Exercise 10.2. Over 100 events, 74 outcomes fell inside their nominal 90 % intervals. State whether the forecaster is underdispersed or overdispersed. Assuming the predictive distribution is normal with the correct center, by what factor must the reported standard deviation be multiplied to achieve nominal coverage? Repeat for 99 outcomes inside, and comment on whether the two corrections are equally urgent.

Exercise 10.3. An operator's penalty for overcommitting is three times its penalty for undercommitting, and the predictive distribution for the target hour is $\mathcal{N}(46, 9.0^2)\,\mathrm{MW}$. Use equation (10.16) to find the loss-minimizing commitment, and compare it with the median and with P10. What cost ratio $\lambda_{\text{over}}/\lambda_{\text{under}}$ would make the loss-minimizing commitment equal to P10?

Exercise 10.4. Two consecutive hours have the marginals $F_{17} \sim \mathcal{N}(50, 5^2)$ and $F_{18} \sim \mathcal{N}(50, 5^2)$, in MW, and the operator must deliver at least 45 MW in each hour. Compute the probability of delivering in both hours under $\mathrm{Corr} = 0$ and estimate it under $\mathrm{Corr} = 0.95$, by simulation if necessary. Then, using the fan-chart law $\mathrm{P10}, \mathrm{P90} = 45 \mp 6\sqrt{h}$ of Figure 10.4, find the horizon at which the central 80 % band first exceeds 20 MW in width, and explain what that implies for how far ahead a commitment of this kind should be made.

Chapter 11

Multi-series and fleet forecasting: pooling, aggregation and common-mode uncertainty

11.1 The fleet problem

Every chapter so far has forecast one series: a load y_t, a flexibility F_t, a temperature T_t. An aggregator does not have one series. It has one series per device, $y_{i,t}$, where i indexes a thermostat, a battery, an electric vehicle or a cohort of them, and the quantity it must deliver to the grid is a sum over i. The question of this chapter is

Should every device be forecast separately, should everything be aggregated first, or should one model be trained across many related devices?

The three answers are not refinements of each other. They differ in accuracy, in what they can be used for, in how much engineering they cost, and above all in how their uncertainties combine.

Jargon: local, cohort and global models; pooling; cold start; hierarchical forecasting; reconciliation; common-mode uncertainty

A *local model* is fitted to one series and used only for that series. A *global model* has one set of parameters fitted across many series and is used for all of them. A *cohort model* sits between the two: one model per group of similar series. *Pooling* is the statistical act of using observations from one series to estimate parameters that another series shares; *full pooling* ignores the identity of the series, *no pooling* fits each separately, and *partial pooling* shares most of the structure while allowing the series to differ. *Cold start* is the problem of forecasting a device that has just been enrolled and has almost no history. *Hierarchical forecasting* produces forecasts at several aggregation levels (device, cohort, region, fleet), and *reconciliation* is the adjustment that makes those forecasts add up. *Common-mode uncertainty* is error shared by every device, such as a single weather forecast error that every thermostat in a metropolitan area experiences at once; it is the part of the uncertainty that aggregation does not average away.

Key idea

Fleet forecasting has two separate levers. Pooling decides how information is shared *across* devices when the model is trained. Aggregation decides at which *level* the forecast is issued, and it sets which part of the uncertainty cancels and which part does not.

This section states the fleet problem in symbols and lays out the three modeling strategies the rest of the chapter compares.

11.1.1 From one series to a fleet

Take a fleet of $N = 100\,000$ enrolled residential thermostats. Device i draws power $P_{i,t}$ at time t and, if dispatched, could change that draw by $F_{i,t}$. The two fleet-level quantities an operator trades are the sums

$$P_{\text{fleet}}(t) = \sum_{i=1}^{N} P_{i,t}, \qquad F_t = \sum_{i=1}^{N} F_{i,t}, \tag{11.1}$$

where the convention for the rest of the chapter is that an unsubscripted F_t is the fleet total and $F_{i,t}$ is the contribution of one device. The fleet flexibility is what the aggregator bids, settles and gets paid for; the device flexibilities are what physically produce it.

Equation (11.1) makes one strategy look obvious: forecast each $F_{i,t}$ and add the forecasts. The sum is exact, so a forecast of each term should give a forecast of the total. That reasoning is correct about the *identity* and wrong about the *statistics*. A sum of 100 000 noisy forecasts is not necessarily a good forecast of the sum, because the errors of the terms have a joint distribution that the summation inherits, and because nothing requires a model trained to predict one device well to be trained to predict the total well. The whole chapter is an unpacking of that gap.

11.1.2 Three strategies

There are three ways to put a model between device data and a device forecast. Writing $X_{i,t}$ for the feature vector of device i built from the information available at t (lags, weather, calendar, device state, exactly as in Chapter 4), they are

$$\text{A. one model per device:} \quad \hat{y}_{i,t+h} = f_i(X_{i,t}), \qquad i = 1, \dots, N, \tag{11.2}$$

$$\text{B. cohort models:} \quad \hat{y}_{i,t+h} = f_k(X_{i,t}), \qquad i \in \mathcal{C}_k, \tag{11.3}$$

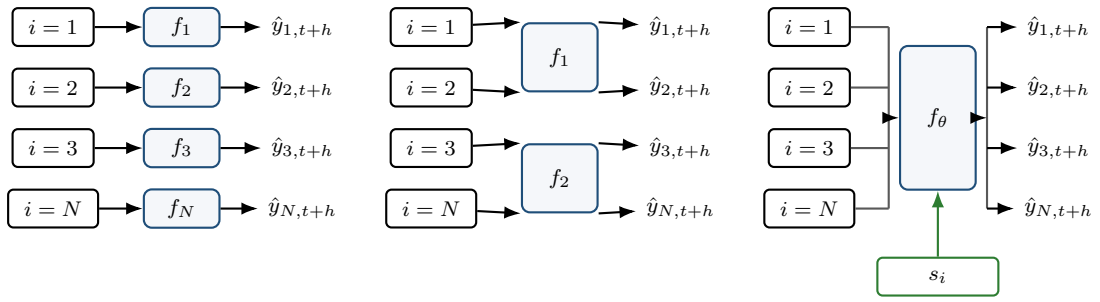
$$\text{C. one global model:} \quad \hat{y}_{i,t+h} = f_\theta(X_{i,t}, s_i), \tag{11.4}$$

with $\mathcal{C}_k$ the k-th cohort and s_i a vector of static, device-specific attributes. Strategy A has N parameter sets, strategy B has as many as there are cohorts, and strategy C has one.

Figure 11.1 draws the three. Each panel shows the same four devices on the left as rounded boxes, the model or models in the middle as blue boxes, and the per-device forecasts on the right. Panel (a) has one model per device and no arrow crosses between rows: no information passes from one device to another. Panel (b) groups the devices into two cohorts, so information is shared inside a group and not across groups. Panel (c) has a single model that every device's data trains and that every device's forecast uses, with the device's own identity entering through the static attribute vector s_i drawn below. Reading the three panels left to right, the number of parameter sets falls from four to two to one, and the amount of data behind each parameter set rises in the same proportion.

11.2 Local, cohort and global models

This section compares the three strategies on the thing that decides between them in practice: how much data stands behind each parameter that has to be estimated. It then introduces partial pooling, which is the arrangement most production fleet forecasters converge on, and ends with what makes a cohort worth defining.



(a) local: N models (b) cohort: one per group (c) global: one model

Figure 11.1: The three strategies of equations (11.2)–(11.4), drawn for four devices. Left column of each panel: devices. Middle: the fitted model or models. Right: the per-device forecasts. In (a) nothing is shared; in (b) devices in a group share a model; in (c) all devices share one model f_θ, and the green box carries the static attributes s_i that let the shared model produce different forecasts for different devices.

11.2.1 One model per device and its data budget

Strategy A fits $f_1, \ldots, f_{100\,000}$, one per thermostat. The appeal is genuine. Each model is personalized: it can learn that this particular household sets the thermostat to 23°C on weekday evenings, that its air conditioner short-cycles above 35°C, that its occupants override dispatch events about one time in four. No pooled model can know those things about a specific home unless they are measured and handed to it as features.

The difficulty is the data budget. Consider a thermostat enrolled thirty days ago, reporting at fifteen-minute resolution. Its history is

$$30 \text{ days} \times 96 \text{ intervals per day} = 2880 \text{ observations}, \tag{11.5}$$

which sounds adequate until the observations are sorted by the conditions that matter. Of those thirty days, perhaps four were hot afternoons; the device may have seen two dispatch events and one rebound; it has never seen a 38°C day, a holiday under peak pricing, or a four-hour curtailment. A local model can estimate the device's typical weekday evening behavior reasonably well and has essentially no information about the rare conditions under which the forecast actually earns money. The relevant sample size is not 2880; it is the handful of observations near the operating point of interest. This is the defining weakness of local models for fleets: they are data-starved exactly where accuracy matters.

11.2.2 Pooling: full, none, and partial

A global model attacks the data budget directly. Its training set is the union over devices and times,

$$\mathcal{D} = \{(X_{i,t},\, y_{i,t+h}) \;:\; i = 1, \ldots, N,\; t \in \mathcal{T}_i\}, \tag{11.6}$$

so a single fit sees millions of rows instead of thousands. More important than the row count is what the rows have in common. The relationship "higher outdoor temperature implies a higher compressor duty cycle" holds, with different coefficients, in every home in the fleet. A local model must rediscover it 100 000 times, each time from a few hundred useful observations. A global model estimates it once, from all of the data. Sharing observations across series to estimate shared structure is statistical *pooling*, and the gain is largest precisely for the rare conditions a single device rarely visits: the fleet as a whole visits them constantly, because at any hour some part of it is hot, some part is being dispatched, and some part is rebounding.

The two extremes are named. *No pooling* is equation (11.2): a separate f_i per device, maximum personalization and minimum data per parameter. *Full pooling* is a single f with no device-specific input at all: maximum data per parameter and no personalization, so two homes with the same weather get the same forecast even if one has a heat pump and the other a conventional air conditioner and a pool pump. Neither extreme is usually the right answer, and the useful arrangements lie between them.

11.2.3 Partial pooling

Partial pooling keeps one set of parameters but lets the model condition on who the device is. A concrete form for a thermostat fleet is

$$\hat{P}_{i,t+h} = f_\theta(X_{i,t},\ \underbrace{z_i,\ \mathrm{eq}_i,\ \mathrm{bt}_i,\ e_i}_{s_i}), \tag{11.7}$$

where z_i is the climate zone, eq_i the equipment type, bt_i the building type and e_i a learned device embedding (Section 11.4.6). The model learns fleet-wide behavior from all the data and expresses individual differences through s_i, which is a low-dimensional description of the device rather than a full private parameter set. The number of parameters does not grow with N; the number of *distinctions* the model can draw does.

Figure 11.2 draws the arrangement. The central blue box is the one shared model. Entering from the left are the time-varying inputs that any forecaster would use: the device's own recent history and the weather forecast of Chapter 4. Entering from the right are the static inputs that make the model partially pooled rather than fully pooled: the known attributes, the cohort indicators, and the learned embedding. The single output on the bottom is the device forecast. The figure's point is the direction of the two input groups: the left-hand inputs change every interval and are shared in *form* across devices, the right-hand inputs are constant in time and differ across devices, and it is their combination inside one function that buys both data efficiency and personalization. Partial pooling is the single most useful idea in fleet forecasting, because it is the only one of the three strategies that does not pay for personalization with data.

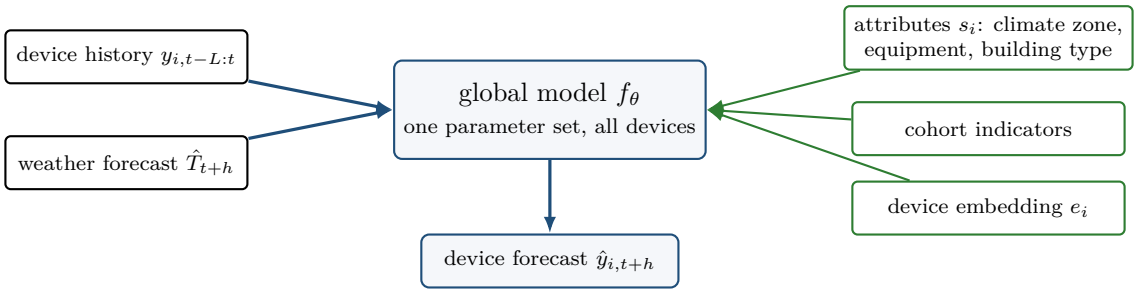


Figure 11.2: Partial pooling, equation (11.7). One shared model takes time-varying inputs from the left and device-specific static inputs from the right, and emits a forecast for the individual device. Full pooling deletes the right-hand boxes; no pooling replaces the central box by one model per device.

11.2.4 Why a global model can beat a local one, and cold start

The comparison that settles the argument is not between two well-observed devices. Take home A, enrolled three years ago, and home B, enrolled yesterday. A local model for A may be excellent. A local model for B cannot exist: there is nothing to fit. A global model, by contrast, can forecast B immediately, because what it has learned is of the form "devices with these characteristics, under these conditions, behave this way", and B's characteristics are known

from enrollment even when its history is not. The thousands of devices that resemble B supply the information B itself has not yet produced.

Cold start. The newly enrolled device is the standing case of this situation, and in a fleet that grows by thousands of devices a month it is never a corner case. What is available at enrollment is equipment type, geography, building type and program terms; what arrives within hours is initial telemetry (indoor temperature, compressor cycling) and the current weather; what is available immediately is the behavior of the rest of the fleet under the same conditions. A global model consumes exactly those inputs. As the device's own history accumulates, its forecasts personalize automatically, because its history enters through $X_{i,t}$ and, if an embedding is used, because e_i is refined at the next retrain. There is no separate cold-start model to maintain and no discontinuity on the day the device becomes well observed.

> **Principle 11.1.** *A global model does not merely save engineering effort. It forecasts devices that have no usable history at all, which is a capability no collection of local models has.*

11.2.5 Cohort models and what makes a cohort useful

Cohort models, equation (11.3), are the practical compromise between the extremes. A thermostat fleet might be partitioned first by climate zone (hot and humid, hot and dry, temperate), then by equipment type (heat pump, conventional air conditioner), then by behavioral segment (high override frequency, reliable participation), and a separate model fitted per cell,

$$\hat{F}_{k,t+h} = f_k(X_{k,t}), \qquad k = 1, \dots, K, \tag{11.8}$$

with $X_{k,t}$ built from cohort-level aggregates. Each f_k has far more data than a local model and far more homogeneity than a fully pooled one, and K models are a tractable number to monitor, retrain and explain.

What makes a cohort useful. A cohort earns its existence only if its members share predictive structure. Variables that usually do carry such structure are climate zone, heating and cooling equipment type, geography at the scale on which weather is shared, building size and vintage, historical duty cycle, setpoint behavior, event participation rate and override frequency. The first four are physical, the last four behavioral, and a good partition usually mixes the two. Groupings chosen for administrative convenience are the common failure. A ZIP code is a billing artifact: two homes in the same ZIP code may sit in different microclimates, with different equipment, occupied on different schedules, and the cohort average then describes neither. The test to apply before defining a cohort is whether its members would be expected to respond similarly to the same weather and the same dispatch command. If they would not, the cohort will have a large within-group variance, its model will fit the average of two unrelated behaviors, and the partition will have made the forecast worse than full pooling.

11.3 Bottom-up, top-down and the statistics of aggregation

Pooling decided how the model is trained. This section turns to the other lever: at which level the forecast is issued. The two canonical answers behave very differently, and the reason is a short piece of variance arithmetic that also explains why very large fleets are not as predictable as a naive calculation suggests.

11.3.1 Device-level and aggregate forecasting

The *bottom-up* forecast predicts each device and sums:

$$\hat{F}_t^{\mathrm{BU}} = \sum_{i=1}^{N} \hat{F}_{i,t}. \tag{11.9}$$

The *top-down* forecast predicts the fleet total directly from fleet-level features:

$$\hat{F}_t^{\mathrm{TD}} = f\big(X_t^{\mathrm{fleet}}\big). \tag{11.10}$$

Both target the same quantity, F_t of equation (11.1). They differ in the dimension of the input, in the noise of the target they are fitted against, and in what their output can be used for. Table 11.1 collects the comparison that the next two subsections develop.

Table 11.1: Bottom-up (equation (11.9)) and top-down (equation (11.10)) fleet forecasting.

	Bottom-up	Top-down
Target fitted	each $F_{i,t}$	the total F_t
Input dimension	one feature vector per device	one short fleet vector
Signal-to-noise	low: individual response is erratic	high: idiosyncratic noise already averaged
Keeps	device detail, per-device availability, personalization	nothing below the fleet level
Enables	device-level targeting and dispatch	portfolio bidding only
Cost	N predictions per interval, model management at fleet scale	one model, simple production path
Uncertainty	must be aggregated across correlated devices	estimated directly at the level of use

11.3.2 The bottom-up approach

Bottom-up forecasting preserves everything about the fleet's interior. It knows which devices are online, which are already at the edge of their comfort band, which have overridden the last three events. It can therefore answer the operational questions that are posed about devices rather than about megawatts: which 20 000 thermostats to call, which to rest, which to exclude because the customer has opted out twice this week.

Its costs are equally concrete. It requires N predictions per interval, which at 100 000 devices and fifteen-minute resolution is 9.6 million predictions a day, plus the feature pipeline that feeds them. The individual target is noisy: one household's response to a curtailment depends on whether somebody happened to open a door, and that noise is irreducible at device level, so device-level error metrics look poor however good the model is. There is a model-management burden, which is severe for strategy A and mild for a global model evaluated per device. And the uncertainty of the sum is hard to obtain, because it requires the joint distribution of 100 000 errors rather than their marginals. Summing a large number of noisy forecasts is a defensible way to estimate a total, but it is not automatically the best one.

11.3.3 The top-down approach

The top-down forecast replaces the fleet's interior by a short vector of aggregate features, for example

$$X_t^{\mathrm{fleet}} = \big[\, N_{\mathrm{avail}}(t),\ \bar{T}_{\mathrm{in},t},\ \text{fraction cooling},\ \bar{D}_t,\ T_{\mathrm{out}},\ \ldots \,\big], \tag{11.11}$$

and fits $F_t = f(X_t^{\text{fleet}})$ directly. Three properties follow. The input dimension is small, so the model needs far fewer observations to fit well. The target is smooth, because the averaging in equation (11.1) has already removed most of the device-level noise before the model ever sees it. And production is simple: one model, one feature pipeline, one prediction per interval. The gradient-boosted trees of Chapter 5 are usually sufficient and often excellent on an input of this kind.

What is lost is everything below the fleet level. The model cannot say which devices are responsible for the number it emits, cannot be queried for a subset of the fleet, and cannot be used to select devices for dispatch. It also inherits a vulnerability examined in Section 11.5: because the fleet's composition is not among its inputs in any explicit way, it cannot distinguish a change in behavior from a change in the population.

11.3.4 Why aggregation makes forecasting easier

The reason the aggregate target is easier is worth deriving rather than asserting. Model device i's load as a predictable part plus noise,

$$P_{i,t} = \mu_i + \varepsilon_{i,t}, \qquad \mathbb{E}[\varepsilon_{i,t}] = 0, \qquad \operatorname{Var}(\varepsilon_{i,t}) = \sigma^2, \tag{11.12}$$

and assume for the moment that the noises are independent across devices. The fleet error is the sum of the device errors, $E_t = \sum_i \varepsilon_{i,t}$, and independence makes the variances add:

$$\operatorname{Var}(E_t) = \sum_{i=1}^{N} \operatorname{Var}(\varepsilon_{i,t}) = N\sigma^2, \qquad \sigma_{\text{fleet}} = \sqrt{N}\,\sigma. \tag{11.13}$$

The absolute uncertainty therefore *grows*, but only as $\sqrt{N}$, while the expected fleet load grows as N: with $\mu_i = \mu$ for all i, $\mathbb{E}[P_{\text{fleet}}(t)] = N\mu$. The ratio of the two is what a forecaster is judged on,

$$\frac{\sigma_{\text{fleet}}}{\mathbb{E}[P_{\text{fleet}}(t)]} = \frac{\sqrt{N}\,\sigma}{N\mu} = \frac{\sigma}{\mu\sqrt{N}}, \tag{11.14}$$

so relative uncertainty falls like $1/\sqrt{N}$.

Example 11.1 (One thermostat and ten thousand)**.** A single thermostat has mean load $\mu = 2\,\text{kW}$ and standard deviation $\sigma = 1\,\text{kW}$. Its relative uncertainty is $1/2 = 50\,\%$: as a forecast of one house, the number is nearly useless. Now take $N = 10\,000$ such thermostats. The expected fleet load is $10\,000 \times 2 = 20\,000\,\text{kW}$, the fleet standard deviation is $\sqrt{10\,000} \times 1 = 100\,\text{kW}$, and the relative uncertainty is $100/20\,000 = 0.5\,\%$. The same physics, the same per-device model and the same noise produce a forecast two orders of magnitude better in relative terms. This is why aggregators can quote fleet capacity to a percent while being unable to predict any single customer. The entire gain, however, rests on the independence assumed in equation (11.13).

11.3.5 Correlated errors and common-mode drivers

Dropping independence changes the conclusion, not the algebra. In general,

$$\operatorname{Var}\Big(\sum_{i=1}^{N} \varepsilon_i\Big) = \sum_{i=1}^{N} \operatorname{Var}(\varepsilon_i) + 2\sum_{i<j} \operatorname{Cov}(\varepsilon_i, \varepsilon_j), \tag{11.15}$$

and the second term was assumed away in equation (11.13). Whenever devices share a driver, their errors share a component, the covariances are positive, and the uncertainty does not vanish with scale.

Fleets share drivers by construction. A single weather forecast feeds every device in a metropolitan area, so a 2°C temperature error at 17:00 is an error in every thermostat's

prediction at once. Extreme heat changes behavior collectively: above some outdoor temperature many air conditioners run continuously instead of cycling, so the fleet leaves the regime the model was fitted on, all at the same moment. A firmware release changes the control logic of a software version, which is to say of a large block of devices simultaneously. A grid event, a price spike or a utility appeal to conserve reaches all customers at once. And the calendar is shared: a holiday, a school closure or a regional event shifts occupancy patterns fleet-wide, which is the calendar effect of Chapter 2 acting as a common mode rather than as a per-device feature. Each of these is a *common-mode* driver, and each one puts a positive term into equation (11.15).

11.3.6 The correlated-error formula and what it explains

A single parameter is enough to see the consequence. Suppose every device has error variance σ^2 and every pair has the same correlation ρ, so $\mathrm{Cov}(\varepsilon_i, \varepsilon_j) = \rho\sigma^2$ for $i \neq j$. There are $N(N-1)/2$ such pairs, and equation (11.15) becomes

$$\mathrm{Var}(E) = N\sigma^2 + N(N-1)\,\rho\,\sigma^2. \tag{11.16}$$

Setting $\rho = 0$ recovers $N\sigma^2$ and Example 11.1. Setting $\rho > 0$ adds a term that grows like N^2, and a term in N^2 eventually dominates a term in N no matter how small its coefficient. Dividing by the expected load $N\mu$ gives the relative uncertainty in closed form,

$$\frac{\sigma_{\text{fleet}}}{N\mu} = \frac{\sigma}{\mu}\sqrt{\frac{1 + (N-1)\rho}{N}} \xrightarrow[N\to\infty]{} \frac{\sigma}{\mu}\sqrt{\rho}, \tag{11.17}$$

which has a floor that no fleet size can go below.

Example 11.2 (One percent correlation)**.** Keep $N = 10\,000$ and $\sigma = 1\,\mathrm{kW}$ and set $\rho = 0.01$, a correlation so weak that no device-level diagnostic would notice it. Then

$$\mathrm{Var}(E) = 10\,000 + 10\,000 \times 9\,999 \times 0.01 = 10\,000 + 999\,900 = 1\,009\,900\ \mathrm{kW}^2,$$

so the covariance term is a hundred times the variance term. The fleet standard deviation is $\sqrt{1\,009\,900} \approx 1005\,\mathrm{kW}$ instead of $100\,\mathrm{kW}$, and the relative uncertainty is $5.0\,\%$ instead of $0.5\,\%$. One percent of pairwise correlation costs a factor of ten in fleet uncertainty.

Figure 11.3 plots equation (11.17) for the two cases. The horizontal axis is the fleet size N, with ticks at the powers of ten from 1 to 10^6; the vertical axis is relative uncertainty in percent, also on a logarithmic scale. The lower curve is the independent case $\rho = 0$, a straight line of slope $-1/2$ in these coordinates, which is the $1/\sqrt{N}$ law. The upper curve is $\rho = 0.01$: it follows the independent curve while $N \lesssim 1/\rho$ and then flattens onto the dashed horizontal line at $(\sigma/\mu)\sqrt{\rho} = 5\,\%$. The vertical dashed line marks the $N = 10\,000$ of the two examples, where the two curves differ by the factor of ten computed in Example 11.2.

What the formula explains. Three operational facts follow from the figure. First, at large N the idiosyncratic uncertainty is already gone; whatever error remains is systematic, shared and structural, so effort spent on per-device accuracy has almost no effect on the fleet number. Second, growth buys much less predictability than $1/\sqrt{N}$ suggests: going from 10 000 to 100 000 devices at $\rho = 0.01$ moves the relative uncertainty from $5.02\,\%$ to $5.00\,\%$, not from $5\,\%$ to $1.6\,\%$. Third, the effective fleet size is $1/\rho$: at $\rho = 0.01$, a fleet of 10 000 correlated devices carries the same relative uncertainty as 100 independent ones. The practical reading is that the limiting factor on a large fleet's forecast is usually the weather forecast, not the device model, and that is where additional effort belongs.

> **Principle 11.2.** *Aggregation removes idiosyncratic uncertainty and leaves common-mode uncertainty untouched. Beyond a fleet size of roughly $1/\rho$, the forecast is a forecast of the common drivers.*

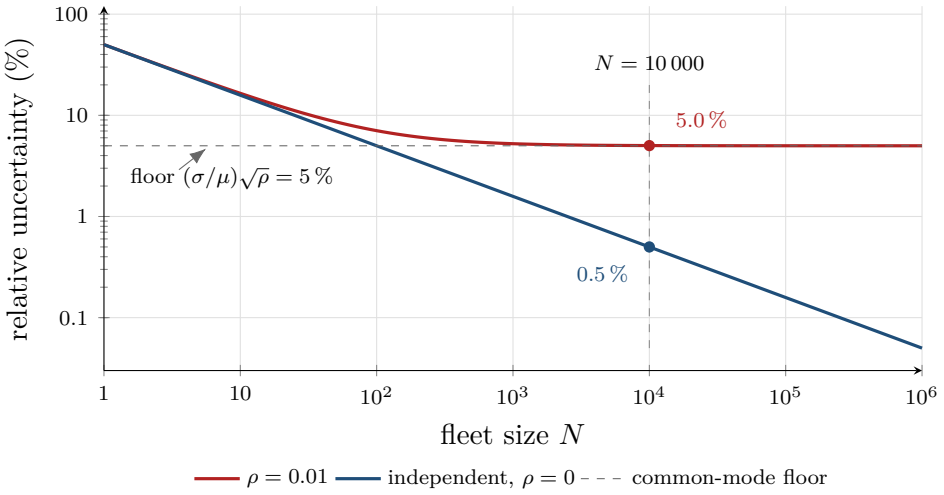


Figure 11.3: Relative fleet uncertainty against fleet size, from equation (11.17) with $\sigma/\mu = 0.5$ (the 2 kW mean and 1 kW standard deviation of Example 11.1). The plotted expression is $100 \cdot (\sigma/\mu)\sqrt{(1 + (N-1)\rho)/N}$ for $\rho = 0$ and $\rho = 0.01$; both axes are logarithmic. The dashed horizontal line is the asymptote $(\sigma/\mu)\sqrt{\rho}$, which the correlated case reaches at about $N = 1/\rho = 100$ and never leaves.

11.4 Hierarchical forecasting and reconciliation

Fleets are not flat. They are organized into regions, utility territories, cohorts and devices, and decisions are taken at every one of those levels. This section sets up the hierarchy, states the coherence requirement that forecasting each level separately violates, and reviews the four ways of satisfying it.

11.4.1 The fleet hierarchy

A realistic aggregator portfolio has at least four levels: the total fleet, regions or utility territories, cohorts within a region, and devices within a cohort. The levels are linked by exact summation,

$$F_{\text{fleet}} = \sum_{r} F_r, \qquad F_r = \sum_{k} F_{r,k}, \qquad F_{r,k} = \sum_{i \in \mathcal{C}_{r,k}} F_i, \tag{11.18}$$

and a set of forecasts is called *coherent* when the forecasts satisfy the same identities as the quantities.

Figure 11.4 draws a two-region example. The top box is the fleet, the middle row is two regions, the next row is two cohorts per region, and the bottom row stands for the devices. Numbers inside the boxes are independently produced forecasts in megawatts. The annotation on the right shows the consequence: the two regional forecasts sum to 110 MW while the fleet forecast says 100 MW, so the set is incoherent, and the red pointer marks the inconsistency that reconciliation has to remove.

11.4.2 The reconciliation problem

The incoherence in Figure 11.4 is not a bug in one of the models. It is what happens whenever levels are forecast independently: the fleet model sees smooth aggregate features and predicts $\hat{F}_{\text{fleet}} = 100$ MW, the regional models see regional weather and predict 60 MW and 50 MW, and nothing in the three fits enforces the identity $F_{\text{fleet}} = F_A + F_B$. An operator handed 100, 60 and 50 cannot act: the bid, the regional commitments and the settlement would use three mutually contradictory numbers.

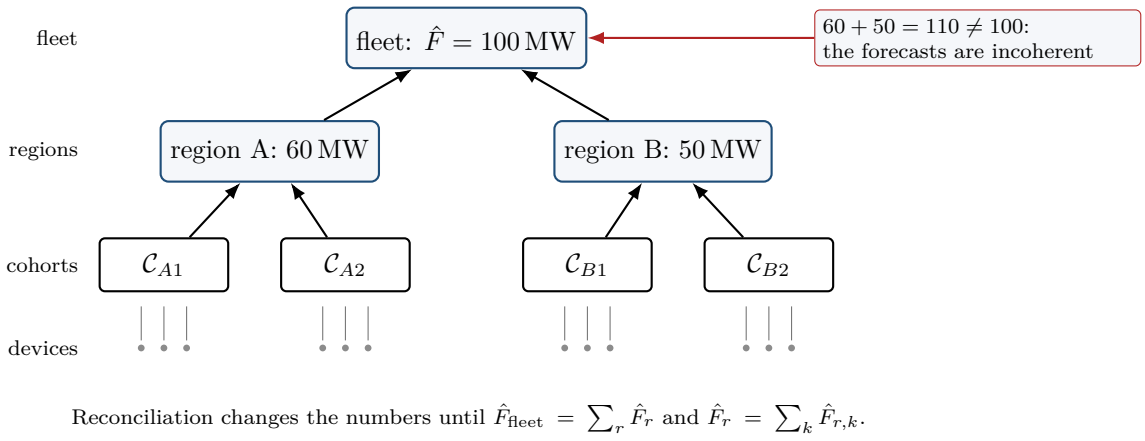


Figure 11.4: A four-level fleet hierarchy, equation (11.18), with illustrative independently produced forecasts. Arrows point from a level to the level that aggregates it. The regional forecasts and the fleet forecast disagree by 10 MW, which is the reconciliation problem of Section 11.4.2.

Reconciliation is the step that adjusts a set of incoherent forecasts into a coherent one. The requirement is not merely arithmetic agreement, which could be achieved by discarding all but one level; it is agreement that preserves as much of the predictive information as possible. A reconciliation that simply overwrites the regional forecasts with a historical split throws away the regional weather signal that justified computing them.

11.4.3 Why several levels are forecast at all

Levels are forecast because decisions are taken at each of them. Table 11.2 lists the levels of Figure 11.4 with the decision each one supports and the forecast quantity that decision needs. The pattern in the table is that aggregate levels support commercial decisions, while the lowest level supports physical ones; no single level serves both.

Table 11.2: Levels of the hierarchy and the decision each supports.

Level	Decision	Forecast needed
Fleet	how much capacity to bid into the market	total deliverable F_t with a lower quantile
Region	commitments per utility territory or node	$F_{r,t}$, limited by local network constraints
Cohort	which segments to call, and in what order	$F_{r,k,t}$ and its reliability
Device	which devices receive the command	$F_{i,t}$, availability, override risk

11.4.4 Bottom-up, top-down and middle-out reconciliation

Three classical reconciliation schemes differ in which level is believed.

Bottom-up. Forecast only the lowest level and sum upward, as in equation (11.9). Coherence is automatic, because every higher-level number is defined as a sum. The weakness is the one of Section 11.3.2: the fleet total inherits the noise and any systematic bias of 100 000 device forecasts, and a small common bias per device becomes a large bias in the total.

Top-down. Forecast the fleet and allocate downward by historical proportions,

$$\hat{F}_{r,t} = p_r \, \hat{F}_{\text{fleet},t}, \qquad \sum_r p_r = 1, \tag{11.19}$$

with p_r the share region r has historically contributed. Coherence is again automatic and the top-level forecast enjoys the aggregation benefit of Section 11.3.4. What is lost is local information: if region A is under a heat wave and region B is mild, the fixed shares p_r will misallocate the total, and the regional numbers will be wrong in opposite directions.

Middle-out. Forecast at an intermediate level, usually the cohort or the region, then aggregate upward to the fleet and allocate downward to devices. This is the common production compromise, because the intermediate level is where the data are plentiful enough for a reliable fit and still structured enough to carry local conditions. It is also an argument for treating the cohort as a first-class object of the forecasting system rather than as a reporting convenience.

11.4.5 Global model plus aggregation

The fourth arrangement combines this chapter's two levers. Train one global model, apply it per device, and aggregate the results:

$$\hat{F}_{i,t+h} = f_\theta(X_{i,t}, s_i) \quad \text{for all } i, \qquad \hat{F}_{\text{fleet},t+h} = \sum_{i=1}^{N} \hat{F}_{i,t+h}. \tag{11.20}$$

This keeps the shared learning of Section 11.2.2, the personalization of Section 11.2.3, the device-level output that dispatch needs, and the coherence of bottom-up reconciliation. It does not solve the uncertainty problem: the sum in equation (11.20) is a point forecast, and the distribution of its error still requires the correlation structure of Section 11.3.6, which a per-device model does not supply by itself. Arrangement and uncertainty are separate questions, and equation (11.20) answers only the first.

11.4.6 Device embeddings and static attributes

The s_i of equation (11.7) can be learned instead of measured. A *device embedding* assigns each device a vector

$$e_i \in \mathbb{R}^d, \qquad \hat{y}_{i,t} = f(X_{i,t}, e_i), \tag{11.21}$$

with e_i estimated jointly with the model parameters, d typically between 4 and 32. After training, the embedding of a device tends to encode persistent properties that no column of the data stated explicitly: heavy cooling use, a morning occupancy pattern, a high override frequency, an unusually well insulated building. The individual coordinates of e_i have no assigned meaning; only distances and directions in the embedding space do, so two devices with nearby embeddings are devices the model has found to behave alike.

Embeddings have two costs. They need enough history per device to be estimated, which reintroduces the cold-start problem for the very devices Section 11.2.4 used to motivate global models, and they are opaque, so a forecast anomaly cannot be traced to an interpretable property. Known static attributes have neither problem: equipment type, climate zone and building category are available at enrollment, are interpretable in a post-mortem, and can be reported to a utility. The sensible default is therefore to personalize with known attributes first and add a learned embedding only when the data support it and the accuracy gain is demonstrated by the rolling-origin validation of Chapter 9.

11.5 Composition, availability and the decomposition of flexibility

A fleet is not a fixed population. Devices join, leave, go offline and opt out, and the mix of equipment and climates changes as the aggregator grows. This section treats composition as a forecasting problem in its own right, decomposes fleet flexibility into a count and a per-device response, and returns to the choice between device-level and aggregate models in the light of both.

11.5.1 Fleet composition changes, and normalization

An aggregator with 100 000 devices today may have 120 000 next month, and the new ones will not be a random sample of the old: a new utility program brings a different climate, a different equipment mix, a different customer demographic. The joint distribution $p(X)$ of the training data therefore changes for reasons that have nothing to do with the behavior being modeled. Fleet forecasting has a distribution-shift problem built into it, and the shift arrives continuously rather than as an event.

The damage is worst for an aggregate model, which can confuse two very different statements. Suppose measured flexibility rises from 40 MW to 50 MW over a month. Either the enrolled customers became more responsive, or there are 25 % more of them and each is as responsive as before. The two have opposite operational meanings, and an aggregate model fitted on F_t alone cannot separate them. The remedy is to normalize, forecasting a per-device quantity and a count instead of a total:

$$\bar{R}_t = \frac{F_t}{N_{\text{avail}}(t)}, \qquad \text{so that} \qquad F_t = N_{\text{avail}}(t)\,\bar{R}_t, \tag{11.22}$$

which separates the scale of the fleet from the average behavior of its members. In the example, $\bar{R}_t$ is flat and N_{avail} has grown.

11.5.2 Availability, and availability versus responsiveness

The count in equation (11.22) is not the enrollment count. A realistic set of figures for one interval is

$$N_{\text{enrolled}} = 100\,000, \qquad N_{\text{online}} = 92\,000, \qquad N_{\text{eligible}} = 80\,000,$$

where devices drop out of the second number through connectivity and out of the third through program rules: already dispatched today, opted out, in a maintenance state, outside the eligible temperature band. The quantity to condition a flexibility forecast on is the last one, $N_{\text{avail}}(t) = 80\,000$, and because it varies by hour, by weather and by dispatch history, it is itself a forecast target. Availability is one of the most informative operational features a fleet forecaster has, and one of the most frequently omitted.

Availability is necessary and not sufficient. An available thermostat delivers nothing if its compressor happens to be off, if the house is already at the top of its comfort band, if the customer overrides within a minute, or if the connected load is small. Writing $A_{i,t} \in \{0, 1\}$ for availability and $R_{i,t} \geq 0$ for the response the device could deliver if called,

$$F_t = \sum_{i=1}^{N} A_{i,t} R_{i,t}, \tag{11.23}$$

and both factors are uncertain at forecast time. Equation (11.23) is the exact version of the informal sum in equation (11.1), and it shows that a fleet flexibility forecast is the forecast of a product, not of a single physical quantity.

11.5.3 Decomposing flexibility into a count and a mean response

Taking expectations in equation (11.23) and writing $N_t = \sum_i A_{i,t}$ for the number available and $\bar{R}_t$ for the mean response among those available gives the decomposition

$$F_t \approx N_t \, \bar{R}_t, \tag{11.24}$$

in words: how many devices can participate, multiplied by how much each one can deliver. The two factors have different drivers. N_t is driven by connectivity, program rules, opt-out behavior and recent dispatch history; $\bar{R}_t$ is driven by weather, thermal state, equipment capacity and the depth of the setpoint change. Forecasting them separately is therefore often easier than forecasting their product, and always more informative.

Example 11.3 (Availability dominates)**.** With $N_{\text{avail}} = 80\,000$ devices available and a mean response of 0.6 kW per device,

$$F_t \approx 80\,000 \times 0.6\,\text{kW} = 48\,000\,\text{kW} = 48\,\text{MW}.$$

Now suppose the response per device is unchanged but availability falls to 70 000, because a connectivity incident takes part of the fleet offline or because more customers have already been dispatched today. Then $F_t \approx 70\,000 \times 0.6 = 42$ MW: a loss of 6 MW, 12.5 % of the committed capacity, with no change whatsoever in device behavior. An availability forecast error of 10 000 devices costs exactly as much as a response forecast error of 0.075 kW per device across the whole fleet. Availability forecasting can matter as much as response forecasting, and it is the easier of the two to instrument.

Figure 11.5 draws equation (11.24) as an area. The horizontal axis is the number of available devices in thousands, the vertical axis is the mean response per available device in kilowatts, and the area of a rectangle is therefore a flexibility in megawatts. The full rectangle, blue plus red, is Example 11.3's $80\,000 \times 0.6\,\text{kW} = 48\,\text{MW}$. The red strip at its right edge is the 10 000 devices lost to availability, whose area is the 6 MW shortfall; the blue area that remains is the 42 MW the fleet still delivers. The figure makes the symmetry of the decomposition visible: a proportional error in either factor costs the same fraction of the area, so neither factor can be treated as a constant.

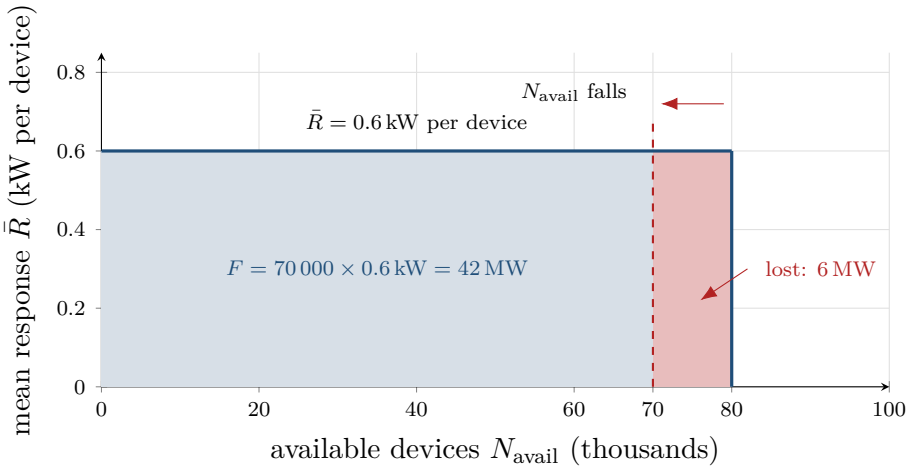


Figure 11.5: Flexibility as an area, equation (11.24). The full rectangle is $N_{\text{avail}} = 80\,000$ devices times $\bar{R} = 0.6$ kW per device, an area of 48 MW (Example 11.3). The red strip is the 10 000 devices lost to availability, an area of 6 MW; the blue area that remains is 42 MW. Numbers are illustrative.

11.5.4 The joint probabilistic view

Both factors in equation (11.24) are random at forecast time, so the distribution of the fleet flexibility is the distribution of a product,

$$p(F_t) = \int\!\!\int p\left(F_t \mid N_t = n, \bar{R}_t = r\right)\, p(n, r)\, \mathrm{d}n\, \mathrm{d}r, \tag{11.25}$$

and what matters is the joint density $p(n, r)$, not the two marginals. The dependence is real and signed. Extreme heat raises $\bar{R}_t$, because more air conditioners are running and there is more load to shed; the same heat lowers N_t, because customers opt out to stay comfortable and because some equipment is already at its capacity limit and cannot be curtailed without a comfort violation. The two effects work against each other, which compresses the upper tail of $p(F_t)$ relative to what independent marginals would predict. Treating N_t and $\bar{R}_t$ as independent and multiplying their distributions is therefore not conservative; it is wrong in a direction that overstates the fleet's capability on exactly the days the capacity is called. The probabilistic machinery of Chapter 10 applies here, with the warning that it must be applied to the pair and not to each factor in turn.

11.5.5 Direct or component forecasts

Two defensible designs follow. A *direct* forecast models F_t in one step: fewer models, a shorter pipeline, and a loss function that optimizes the operational quantity itself, which is the strongest argument for it, since minimizing the error of a product is not the same as minimizing the errors of its factors. A *component* forecast models N_t and $\bar{R}_t$ separately: it is interpretable, so a change in the forecast can be attributed to availability or to response; it diagnoses, so an anomaly can be localized; and it generalizes better when the fleet grows, because the count enters explicitly rather than being implicit in a fitted level. There is no universal answer. In practice the component form is used for monitoring and explanation, the direct form is used where the bid number must be as accurate as possible, and both are maintained against each other as a consistency check.

11.5.6 Global sequence models, and when aggregation still wins

Global training is what makes the sequence models of Chapters 7 and 8 viable in this setting. A global recurrent network shares one parameter set across all devices,

$$\mathbf{h}_{i,t} = f_\theta(x_{i,t}, \mathbf{h}_{i,t-1}), \qquad \hat{y}_{i,t+h} = g_\theta(\mathbf{h}_{i,t}, s_i), \tag{11.26}$$

so that the recurrence is fitted on millions of device sequences while the static vector s_i supplies the device's identity. The equivalent with attention replaces the recurrence by a window,

$$\hat{y}_{i,t+h} = f_\theta\big(x_{i,t-L:t},\ s_i,\ \hat{w}_{t:t+h}\big), \tag{11.27}$$

with $\hat{w}_{t:t+h}$ the known-future covariates, chiefly the weather forecast. Both forms are far more plausible than they were in the single-series chapters, for one reason: a network with hundreds of thousands of parameters fitted to one thermostat's 2880 observations overfits, while the same network fitted to 100 000 thermostats' sequences has data commensurate with its capacity.

Capacity is not the same as advantage. If the operational target is F_t at fleet level, and if the aggregate features of equation (11.11) already capture what drives it, then a gradient-boosted model on a dozen fleet-level features can outperform a device-level deep architecture, train in minutes instead of hours, and be debugged by inspection. The aggregation benefit of Section 11.3.4 is working for the simpler model and against the complicated one, which must predict a noisy device target before its output is summed. The complexity has to earn itself against that baseline, measured the way Chapter 9 prescribes, and it frequently does not.

11.6 Heterogeneity, diversification and the design of a fleet forecaster

The last section of the body is about what the averages hide, how cohort uncertainties combine into a portfolio uncertainty, at which levels a fleet forecaster should be validated, and what a defensible end-to-end design looks like.

11.6.1 Heterogeneity behind the mean

Consider two fleets, each with 10 000 available devices and an average response of 0.5 kW per device, hence 5 MW expected in both cases. In fleet A every device delivers about 0.5 kW. In fleet B half the devices deliver nothing and half deliver 1.0 kW. The mean is identical and nothing else is. Fleet B can be dispatched at half the device count for the same megawatts, so targeting matters; its comfort impact is concentrated on the responsive half, so customer fatigue accumulates faster in a smaller group; and its outcome depends on which half happens to be available, so its variance is larger. The aggregate statistic that both fleets share is the one statistic that does not distinguish them.

Heterogeneity as information. Dispersion is not only a hazard. Take two cohorts with the same expected flexibility, $F_A = 20 \pm 2$ MW and $F_B = 20 \pm 10$ MW, the second number being a standard deviation. They offer the same capacity and entirely different reliability. A dispatcher who must deliver a firm 18 MW will call cohort A, where 18 MW is one standard deviation below the mean, and will not rely on cohort B, where it is within a fifth of one. The useful characterization of a fleet is therefore a triple: expected capacity, heterogeneity across members, and uncertainty of the total. Reporting only the first invites a commitment the fleet cannot keep.

11.6.2 Portfolio diversification and the covariance of cohorts

A portfolio of cohorts is a sum of random quantities, $F = F_A + F_B + F_C$, and its variance is the standard quadratic form

$$\operatorname{Var}(F) = \mathbf{1}^{\mathsf{T}} \Sigma \, \mathbf{1} = \sum_{k} \sigma_k^2 + 2 \sum_{k<l} \Sigma_{kl}, \tag{11.28}$$

with Σ the covariance matrix of the cohort uncertainties. This is equation (11.15) again, written at cohort level, and the reading is the same: if cohort uncertainties are weakly correlated their errors partly offset and the relative risk of the portfolio is smaller than that of its parts, while if they are driven by one shared factor such as a regional heat wave, there is little diversification to be had.

Geography is the main lever on Σ. Weather forecast errors in New York, Texas and California are nearly unrelated at a given hour, so a geographically spread portfolio diversifies. Within one metropolitan area the same forecast error reaches every device, and the cohorts of that area are almost perfectly correlated; splitting them into finer cohorts does nothing for the portfolio variance, because the shared driver is unaffected by how the devices are grouped. A fleet's diversification is a property of its footprint, not of its taxonomy.

Reconciliation with uncertainty. The same arithmetic makes probabilistic hierarchical forecasting harder than point reconciliation. Let $F_A \sim \mathcal{N}(30, 5^2)$ and $F_B \sim \mathcal{N}(30, 5^2)$ in megawatts. If they are independent, $\operatorname{Var}(F_A + F_B) = 50$ and the total has standard deviation $\sqrt{50} \approx 7.1$ MW. If instead they are strongly positively correlated, at $\rho = 0.9$, the variance is $25 + 25 + 2 \times 0.9 \times 25 = 95$ and the standard deviation is 9.7 MW, nearly the 10 MW of

perfect correlation. The means add in both cases, so a point reconciliation cannot tell the two situations apart; only the second moment can, and a probabilistic reconciliation must therefore carry and reconcile a covariance, not a vector of numbers.

Figure 11.6 shows both halves of this argument. On the left is the covariance matrix Σ of three cohorts as a grid of nine cells: the green diagonal holds the variances σ_k^2 and the red off-diagonal cells hold the covariances, and equation (11.28) says that the portfolio variance is the sum of all nine cells, which is why the six off-diagonal cells outnumber the three on the diagonal and come to dominate as cohorts are subdivided. On the right the total standard deviation of the two-cohort example is plotted against the correlation ρ between them, from the independent value $\sqrt{50} \approx 7.1\,\mathrm{MW}$ at $\rho = 0$ to the perfectly correlated $10\,\mathrm{MW}$ at $\rho = 1$, with the two markers corresponding to cohorts in different regions and cohorts in the same metropolitan area. The curve's steepness near $\rho = 0$ is the practical warning: the first few percent of correlation cost more than the last few.

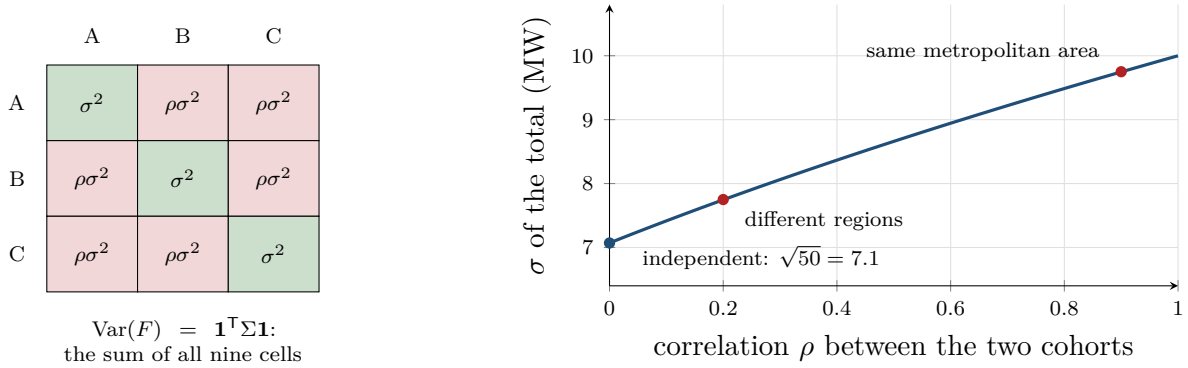


Figure 11.6: Left: the covariance matrix of three cohorts with common variance σ^2 and common correlation ρ; the portfolio variance of equation (11.28) is the sum of every cell, three on the diagonal and six off it. Right: the standard deviation of $F_A + F_B$ for $F_A, F_B \sim \mathcal{N}(30, 5^2)$ as a function of their correlation, $\sigma = \sqrt{50 + 50\rho}$ megawatts. The markers at $\rho = 0.2$ and $\rho = 0.9$ are illustrative values for cohorts in different regions and in the same metropolitan area.

11.6.3 Validation at several levels

A fleet forecaster must be validated at every level at which it is used: at device level, where the question is whether individual responses are predicted; at cohort level, where the question is whether segment behavior is predicted; and at fleet level, where the question is whether the total deliverable capacity is predicted. The three verdicts can differ sharply, and a model can be poor at device level and excellent in total. That combination is acceptable when the decision is taken at fleet level, and it is the normal outcome of Section 11.3.4's arithmetic rather than a sign of a problem.

Example 11.4 (Device error and fleet error)**.** A global model predicts individual thermostat response with MAE $= 0.5\,\mathrm{kW}$. Against a device whose response is of order $1\,\mathrm{kW}$, that is an enormous relative error, and a reviewer looking only at device-level metrics would reject the model. Aggregated over 80 000 devices, the same model's fleet error is 2 % of the total, which is comfortably adequate for portfolio dispatch and bidding. A fleet forecaster judged only by device accuracy will be rejected for failing at a task nobody asked it to perform.

When device resolution is genuinely required. The converse also holds, and it is the limit of pure aggregation. Suppose the dispatch plan calls for curtailing 20 000 specific thermostats out of 100 000, chosen to spread customer fatigue, to respect each customer's event limit and to

avoid homes that are already warm. A fleet-level forecast of 48 MW cannot name a single device, so it cannot produce that plan. Device-level or at least cohort-level resolution is a requirement of the optimizer, not a preference of the forecaster, and this is the clearest case in the book of forecasting and optimization interacting: what the downstream decision selects over determines the level at which the upstream forecast must be issued.

Principle 11.3. *Validate at the level of the decision, and issue the forecast at the finest level any decision selects over. Device accuracy matters only to the extent that a decision is taken about a device.*

11.6.4 A fleet forecasting design

The pieces of this chapter compose into one reasonable design, not into a claim about any particular product. Figure 11.7 draws it as a seven-stage pipeline. The first row runs left to right: device telemetry arrives; a state estimation step turns it into estimated device and cohort states, including availability and thermal state; and a global response model of the form of equation (11.7) is evaluated on those states. The model's output fans into the three levels of Table 11.2, drawn as the three stacked boxes, which are then reconciled into a coherent set. The second row carries the reconciled forecast into a probabilistic fleet forecast, where the common-mode correlation of Section 11.3.6 is applied, and hands that distribution to the dispatch optimizer. Two features of the diagram are the chapter's conclusions in graphical form: training happens once, in the single global-model box, while prediction happens at three levels; and the uncertainty step sits after reconciliation, because the correlation structure is a property of the fleet rather than of any one level's model.

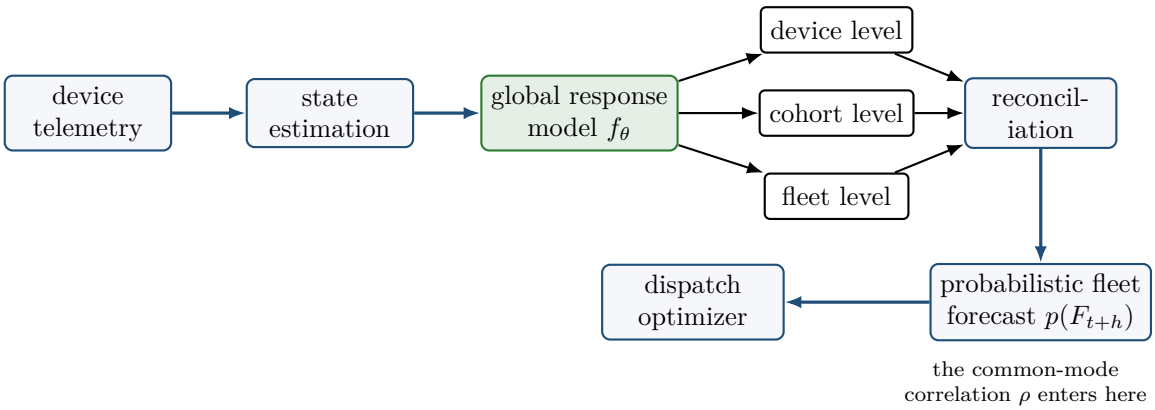


Figure 11.7: A fleet forecasting design. Telemetry is turned into estimated device and cohort states; one global model is evaluated per device; forecasts are produced at three levels, reconciled into a coherent set, given a joint distribution that includes the common-mode correlation, and handed to the optimizer. The flow is left to right on the upper row and right to left on the lower one.

The three conceptual questions about this design are the three choices the chapter has made.

Conceptual questions

"Would you build one model per thermostat?"
Probably not as a default. The reason is the data budget of equation (11.5): device histories are short, noisy and particularly sparse in exactly the rare conditions, such as a dispatch on an extreme afternoon, under which the forecast is worth money. A strong answer proposes a global or partially pooled model that conditions on device and cohort attributes, which recovers most of the personalization while keeping the data behind every parameter, and then adds the qualification that the level of the forecast depends on the decision: portfolio capacity can be forecast in aggregate, whereas device-level dispatch needs resolution at device or cohort level.

Conceptual questions

"Why does a large fleet become easier to forecast?"
Because idiosyncratic variability averages out. A strong answer gives equation (11.14): under independence, relative uncertainty falls like $1/\sqrt{N}$, which is the 50 % to 0.5 % of Example 11.1. It then supplies the limit immediately, without waiting to be asked: common-mode errors, from the shared weather forecast, a firmware release or correlated customer behavior, do not average away, equation (11.16) shows the covariance term growing like N^2, and at $\rho = 0.01$ and $N = 10\,000$ it is already a hundred times the independent term. At scale, the fleet forecast is a forecast of the common drivers.

Conceptual questions

"Forecast total fleet response directly, or sum device forecasts?"
Both should be built and compared. A strong answer gives the reason on each side: the direct forecast benefits from the aggregation of Section 11.3.4 and is trained on the operational quantity itself, while the bottom-up forecast preserves the heterogeneity and per-device availability that dispatch requires. It then describes what production systems actually do, which is neither extreme: global device and cohort models reconciled against an aggregate fleet forecast, so that the coherent set supports both the bid and the device selection.

Training level and prediction level. The design separates two things that are easy to conflate. The model is *trained* on millions of device-level observations, because that is where the information is; it is *used* to predict $\hat{F}_{\text{fleet}} = \sum_i f_\theta(X_{i,t}, s_i)$, because that is where the decision is. Training granularity and decision granularity are independent choices, and the fact that a quantity is only needed in aggregate is not an argument for training on aggregates.

Why this is not one aggregate series. It would be simpler to forecast $P_{\text{fleet}}(t)$ as a single univariate series with the methods of Chapters 3 to 6, and the simplification fails on composition. Suppose the fleet shifts over a year from 70 % conventional air conditioners to 50 % heat pumps. Heat pumps have a different load shape, a different temperature sensitivity and a different curtailment response, so the relationship between weather and aggregate flexibility changes even though no individual customer has changed behavior. A univariate model of P_{fleet} can only treat this as drift, and will track it with a lag. A global device or cohort model represents it explicitly: the equipment mix is an input, and the same fitted parameters give the right answer before and after the shift.

11.6.5 Fleet-level feature engineering

An aggregate model is only as good as the summary it is given, and the usual summary is too coarse because it reports means. A mean indoor temperature of 23°C is consistent with every

home at 23°C and with half the homes at 21°C and half at 25°C, and those two fleets have very different curtailable load. The correction is to feed the model *distributional* features: the 10th, 50th and 90th percentiles of indoor temperature rather than its mean, the fraction of devices in cooling mode, the fraction above setpoint, quantiles of the duty-cycle distribution, the equipment-type fractions, and the distribution of availability rather than its total. Each of these is a statistic of the fleet's interior that survives aggregation, which is precisely what a top-down model needs in order not to lose the interior entirely.

Table 11.3 lists a concrete fleet feature vector of this kind, the forecast being $\hat{F}_{t+h} = f(X_t^{\text{fleet}})$. The table gives each feature, what it measures and why it carries information the mean does not. A feature vector of a dozen entries like these, fitted with gradient-boosted trees, is a strong fleet-level forecaster and a baseline that any device-level architecture should be required to beat.

Table 11.3: A fleet-level feature vector X_t^{fleet} for a thermostat fleet, with the forecast $\hat{F}_{t+h} = f(X_t^{\text{fleet}})$.

Feature	What it measures	Why it is not redundant
$N_{\text{avail}}(t)$	devices online and program-eligible	sets the scale of equation (11.24)
T_{out}, $\hat{T}_{t+h}$	current and forecast outdoor temperature	the dominant common-mode driver
$\bar{P}_t$	mean device load	the level of connected load
$P_{90}(P_{i,t})$	90th percentile of device load	identifies the large-load tail that supplies most of the response
fraction cooling	share of devices with the compressor running	only running devices can be curtailed
$\bar{T}_{\text{in},t}$	mean indoor temperature	thermal headroom on average
$P_{90}(T_{\text{in}} - T_{\text{set}})$	90th percentile of the comfort margin	how much of the fleet is already at the top of its band
recent response	verified delivery in recent events	the behavioral calibration of the fleet
cohort fractions	equipment and climate mix	makes composition change an input rather than drift

11.6.6 Cohort forecasts, explainability and problem detection

Cohort-level forecasts pay for themselves in explanation even when they do not improve the fleet number. Suppose the total forecast for tomorrow evening drops from 60 MW to 45 MW between two runs. A fleet-level model can report the drop and nothing else. Table 11.4 shows what a cohort decomposition reports instead: the same 15 MW split across the cohorts and the availability term that produced it. The table's rows are the contributions, which sum to the total change; the point is not the arithmetic but that each row is separately actionable, and that the largest row is not the one an operator would have guessed from the weather.

Detecting problems. The same decomposition is the fleet's monitoring instrument. If one cohort starts behaving differently, because a firmware release changed its control logic, because a telemetry pipeline is dropping its reports, because a configuration change altered its setpoint limits, or because that customer segment genuinely changed its routine, an aggregate model may show nothing at all: the other cohorts move in the opposite direction and the total stays within its usual band. The problem is then discovered on the day the fleet is dispatched and

Table 11.4: Attribution of a 60 MW to 45 MW change in the fleet flexibility forecast, by cohort. Illustrative figures.

Contribution	Change (MW)	What it says
Texas heat-pump cohort	−2	milder forecast in that region
Northeast air-conditioner cohort	−1	minor, within normal variation
Low-duty-cycle cohort	−9	the dominant term: little load running to curtail
Availability	−3	fewer eligible devices than in the previous run
Total	−15	60 MW becomes 45 MW

underdelivers. Segment-level monitoring turns the same event into a visible, localized anomaly with a named cohort attached to it, which is the difference between a diagnosis and a surprise.

11.6.7 The core trade-off

Everything in the chapter is a position on one trade-off. Device-level detail buys personalization, availability and the ability to select devices, and it costs computation, model management and a target so noisy that device-level metrics look bad however good the model is. Fleet aggregation buys stability, a high signal-to-noise ratio and a simple production path, and it costs the fleet's interior: the heterogeneity of Section 11.6.1, the composition of Section 11.5.1 and the ability to answer any question about a subset.

The arrangement that resolves the trade-off rather than choosing a side has three parts: global learning, so that every parameter is estimated from the whole fleet's data; device and cohort context, so that the shared model can still distinguish a heat pump in Houston from an air conditioner in Boston; and an aggregate operational forecast with an honest joint distribution, so that the number handed to the market is the one the fleet will actually deliver.

11.7 What to retain

Definition 11.1 (Fleet forecasting)**.** Forecasting a fleet is forecasting $y_{i,t}$ for many i at once, and the two decisions are how much information to pool across devices (local, cohort, global, partially pooled) and at which level to issue the forecast. Aggregation reduces relative uncertainty like $1/\sqrt{N}$ only for the idiosyncratic part of the error; the common-mode part, set by shared drivers such as the weather forecast, does not average away and dominates at scale. The forecasting level must therefore match the decision: fleet level for portfolio capacity, cohort level for segment dispatch, device level for targeting.

Table 11.5 collects the chapter's objects with the subsection that defines each.

11.8 Exercises

Exercise 11.1. A fleet of identical devices has $\mu = 2\,\text{kW}$ and $\sigma = 1\,\text{kW}$ per device. Using equation (11.17), tabulate the relative fleet uncertainty for $N = 10^3, 10^4, 10^5$ and $\rho \in \{0,\ 0.001,\ 0.01\}$. For each $\rho > 0$, find the fleet size at which the covariance term equals the variance term, and state the floor the relative uncertainty approaches. Then say what fraction of the improvement from $N = 10^4$ to $N = 10^5$ survives at $\rho = 0.001$.

Table 11.5: The objects of fleet forecasting.

Object	Definition	Section
$y_{i,t}$	the series of device or cohort i	11.1.1
f_i, f_k, f_θ	local, cohort and global models	11.1.2
pooling	sharing observations across series to fit shared structure	11.2.2
s_i	static device attributes that personalize a shared model	11.2.3
cold start	forecasting a device with no usable history	11.2.4
$\mathcal{C}_k$	a cohort: devices sharing predictive structure	11.2.5
$\hat{F}_t^{\mathrm{BU}}$, $\hat{F}_t^{\mathrm{TD}}$	bottom-up sum and top-down direct forecast	11.3.1
X_t^{fleet}	aggregate feature vector of the fleet	11.3.3
$\sigma/(\mu\sqrt{N})$	relative fleet uncertainty under independence	11.3.4
ρ	pairwise correlation of device errors; floor $(\sigma/\mu)\sqrt{\rho}$	11.3.6
coherence	forecasts satisfying the hierarchy's summation identities	11.4.1
reconciliation	adjusting incoherent forecasts while keeping information	11.4.2
p_r	historical share used in top-down allocation	11.4.4
e_i	learned device embedding	11.4.6
$N_{\mathrm{avail}}(t)$	devices online and program-eligible	11.5.2
$A_{i,t}$, $R_{i,t}$	availability indicator and potential response	11.5.2
$F_t \approx N_t \bar{R}_t$	count times mean response	11.5.3
$\mathbf{1}^{\mathsf{T}}\Sigma\mathbf{1}$	portfolio variance of cohort uncertainties	11.6.2

Exercise 11.2. The forecasts of Figure 11.4 are $\hat{F}_{\mathrm{fleet}} = 100\,\mathrm{MW}$, $\hat{F}_A = 60\,\mathrm{MW}$, $\hat{F}_B = 50\,\mathrm{MW}$. Reconcile them three ways: bottom-up, top-down with historical shares $p_A = 0.55$ and $p_B = 0.45$, and a proportional adjustment that scales both regional forecasts to sum to 100 MW. State the three resulting triples. Then let $F_A \sim \mathcal{N}(30, 5^2)$ and $F_B \sim \mathcal{N}(30, 5^2)$ and compute the standard deviation of the reconciled total for $\rho = 0$ and $\rho = 0.8$, using equation (11.28).

Exercise 11.3. Start from Example 11.3: $N_{\mathrm{avail}} = 80\,000$ and $\bar{R} = 0.6\,\mathrm{kW}$ give 48 MW. (a) Availability falls to 70 000 with $\bar{R}$ unchanged. (b) Availability falls to 70 000 while heat raises $\bar{R}$ to 0.65 kW. (c) Availability holds at 80 000 while $\bar{R}$ falls to 0.525 kW. Compute the flexibility in each case, and say which of the two factors a forecaster should instrument first if the relative forecast error of N_{avail} is 5 % and that of $\bar{R}$ is 12 %.

Exercise 11.4. Using Table 11.4, verify that the four contributions account for the full 15 MW change and identify which of them an operator can act on within the hour. Then take the two cohorts of Section 11.6.1, $F_A \sim \mathcal{N}(20, 2^2)$ and $F_B \sim \mathcal{N}(20, 10^2)$ in megawatts, and compute $\Pr(F \geq 18\,\mathrm{MW})$ for each. Which cohort can support a firm 18 MW commitment at 95 % confidence, and what firm commitment can each support at that confidence?

Chapter 12

Flexibility and counterfactual forecasting: the baseline that is never observed

12.1 The missing world

Every forecasting problem so far has had an observable target. A load, a temperature, a fleet total: the forecast is issued, time passes, the quantity is measured, and the error is computed. Flexibility is not like that. The quantity an aggregator sells is a difference between two worlds, and at most one of the two ever occurs. The question of this chapter is

how to forecast a quantity defined as the difference between what did happen and what would have happened, when only one of the two is ever measured.

Ordinary forecasting asks what will happen. Flexibility forecasting asks what would happen if nothing is done, compared with what would happen if an action is taken. That is a counterfactual question, and it changes the statistics, the labeling, the validation and the architecture of the forecasting system.

Jargon: counterfactual, potential outcome, treatment effect, confounder, baseline, rebound, dose-response

A *counterfactual* is the outcome of a world that did not occur: the load a fleet would have drawn had it not been dispatched. A *potential outcome* is the value the target would take under a specified action, written $Y(0)$ for no action and $Y(1)$ for the action; each unit has a potential outcome for every action, and only one of them is realized. The *treatment effect* is the difference between two potential outcomes, here $\tau = Y(0) - Y(1)$, the reduction achieved. A *confounder* is a variable that influences both which action was taken and what the outcome was, so that comparing acted-on and not-acted-on cases measures the confounder rather than the action; the standing example is outdoor temperature. The *baseline* is the estimate of $Y(0)$ for an event that was in fact dispatched, in the demand-response sense of the word rather than the modeling sense of Chapter 3. *Rebound* is the excess consumption after a curtailment ends, as the equipment that was held back recovers. *Dose-response* describes a response that varies continuously with the strength of the action rather than being on or off.

Key idea

Flexibility is $F = P^{\text{base}} - P^{\text{ctrl}}$, and once the fleet is dispatched the first term never occurs. The estimate that replaces it is part of the model, not an input to it: its error becomes error in every label, every validation statistic and every dispatch decision downstream.

This section states the problem, introduces the notation causal inference uses for it, and says precisely which property of ordinary forecasting is lost.

12.1.1 The unobserved baseline

Take a concrete event. At 17:00 the aggregator raises the setpoints of a large thermostat fleet by two degrees. The fleet's metered consumption during the event is $P^{\text{ctrl}} = 80\,\text{MW}$. Without the intervention the fleet would, on the evidence of the weather and the hour, have drawn something like 120 MW. The flexibility delivered is therefore

$$F = P^{\text{base}} - P^{\text{ctrl}} = 120\,\text{MW} - 80\,\text{MW} = 40\,\text{MW}. \tag{12.1}$$

Of the three numbers in equation (12.1), exactly one was measured. The 80 MW came off the meters. The 40 MW is the quantity that was bid, dispatched and settled. The 120 MW never happened: the world in which the setpoints were left alone was not the world that occurred, and no instrument recorded it. The number that carries the whole result is the one with no observation behind it.

Figure 12.1 draws the two worlds. The horizontal axis is clock time from 15:00 to 21:00; the vertical axis is fleet load in megawatts. The gray band marks the event window, 17:00 to 18:00. The dashed gray curve is the baseline P^{base}, the load the fleet would have drawn with no intervention; it is drawn dashed because during and after the event it is an estimate and not a measurement. The solid blue curve is the controlled load P^{ctrl}, which is what the meters record: it tracks the baseline until 17:00, drops to 80 MW for the hour of the event, and then rises *above* the baseline after 18:00 before relaxing back onto it. The red double arrow inside the event window is the flexibility of equation (12.1). The two shaded areas are energies rather than powers: the area between the curves during the event is the energy not consumed, and the area between them after the event is the energy consumed in excess once the setpoints were released. Three features of the figure organize the rest of the chapter. The dashed curve is unobserved, so it must be modeled. The solid curve depends on the action taken, so it is a response and not a prediction. And the two curves separate after the event as well as during it, so an intervention has consequences beyond the interval in which it is applied.

12.1.2 Potential-outcomes notation

The vocabulary that handles this situation precisely comes from causal inference. For a unit under study, which here may be a single thermostat, a cohort or the whole fleet, define two *potential outcomes*: $Y(0)$, the load that would occur with no intervention, and $Y(1)$, the load that would occur under the intervention. In the notation of this book,

$$Y(0) = P^{\text{base}}, \qquad Y(1) = P^{\text{ctrl}}, \tag{12.2}$$

and the treatment effect is their difference,

$$\tau = Y(0) - Y(1), \qquad F = Y(0) - Y(1). \tag{12.3}$$

The sign convention is worth stating explicitly, because the usual convention in statistics is $Y(1) - Y(0)$. Here the useful quantity is a *reduction*, so the untreated outcome comes first and a positive F means the fleet consumed less because it was dispatched. With that convention, equation (12.3) is exactly equation (12.1): flexibility *is* a treatment effect.

The difficulty follows immediately from the definition. For any given event the aggregator either dispatched or did not, and what it observes is

$$Y = Y(u), \qquad u \in \{0, 1\}, \tag{12.4}$$

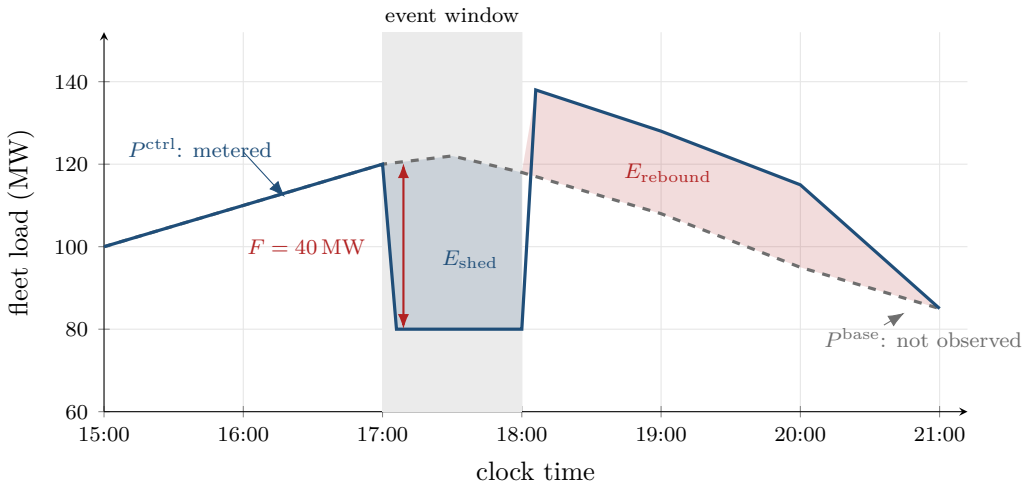


Figure 12.1: The two worlds of a 17:00 curtailment event, with illustrative piecewise-linear load trajectories. The dashed gray curve is the baseline the fleet would have followed with no intervention; the solid blue curve is the metered controlled load. The red double arrow is the flexibility of equation (12.1). The blue area is the curtailed energy E_{shed} and the red area the rebound energy E_{rebound} of equation (12.21); they are approximately equal here, which is the shifted-energy case of Section 12.3.5. Only the solid curve corresponds to a measurement.

the potential outcome corresponding to the action actually taken. The other potential outcome is not merely unmeasured; it does not exist as an event in the world. This is the *fundamental problem of causal inference*: the difference in equation (12.3) is a difference of two numbers of which at most one is ever available for the same unit at the same time. Every method in this chapter is a way of supplying the missing term, either by modeling it or by finding a comparable unit for which it was realized.

12.1.3 Error computation in ordinary and counterfactual forecasting

The contrast is sharpest in the error calculation. In ordinary load forecasting the procedure is a loop that closes. A forecast $\hat{P}_{t+h}$ is issued at time t; at time $t+h$ the load P_{t+h} is metered; the error

$$e_{t+h} = P_{t+h} - \hat{P}_{t+h} \tag{12.5}$$

is computed, accumulated into the MAE and RMSE of Chapter 9, and used to retrain and recalibrate. Every forecast eventually receives a verdict.

For a counterfactual baseline the loop does not close. Once the fleet is dispatched, P^{base} for that event is never metered, so the analog of equation (12.5),

$$e^{\text{base}} = P^{\text{base}} - \hat{P}^{\text{base}}, \tag{12.6}$$

cannot be evaluated for the events that matter. The baseline model can be validated on days when no dispatch occurred, which tests it under conditions that differ systematically from event conditions, and it cannot be validated on the events themselves at all. The same gap propagates: the flexibility delivered cannot be measured, so the settlement figure, the performance statistics and the training labels all rest on a model whose error is unobservable by construction. Validation, treated in Chapter 9 as a matter of arranging the data in time, becomes the hardest part of the problem and is taken up in Section 12.6.2.

12.1.4 The three quantities

Chapter 1 introduced three quantities and used them informally. They can now be stated with the action in them. For a fleet observed at time t and an action u,

$$P^{\text{base}}(t) = \text{load with no intervention, written } P_0(t), \tag{12.7}$$

$$P^{\text{ctrl}}(t \mid u) = \text{load under action } u, \text{ written } P_u(t), \tag{12.8}$$

$$F(t, u) = P^{\text{base}}(t) - P^{\text{ctrl}}(t \mid u). \tag{12.9}$$

Three observations about equations (12.7)–(12.9) set up the chapter. First, P^{base} carries no u: it is the single no-action world, and the subscript 0 in P_0 is the action $u = 0$. Second, P^{ctrl} carries u as an argument, not as a label, because the load under a one-degree change for thirty minutes is a different number from the load under a three-degree change for two hours. Third, and as a consequence, F is a function of two arguments. Writing it as $F(t)$, a function of time alone, discards the dependence on the action, which is precisely the dependence an optimizer needs.

12.2 Two architectures and the baseline problem

There are two ways to build a flexibility forecaster: predict the two worlds and subtract, or predict their difference directly. This section compares them and then examines the error that both inherit, because the historical labels needed to train either one are themselves products of a baseline estimate.

12.2.1 Approach A: forecast both worlds separately

The first architecture mirrors the definition. Fit one model for the no-action world and another for the response, and subtract their outputs:

$$\hat{P}^{\text{base}} = f_0(X), \qquad \hat{P}^{\text{ctrl}} = f_u(X, u), \qquad \hat{F} = \hat{P}^{\text{base}} - \hat{P}^{\text{ctrl}}. \tag{12.10}$$

The pipeline reads: current state, then a no-control model producing P_0; current state together with the proposed control u, then a response model producing P_u; then the difference. Both terms are physically meaningful quantities with units of power, each can be inspected, plotted against metered load on non-event days and reasoned about by an operator, and the first of the two is an ordinary load forecast of the kind Chapters 4 to 8 already solve. The structure also matches how settlement works, since a utility is usually paid against a baseline and a meter rather than against a modeled difference.

12.2.2 Approach B: predict flexibility directly

The second architecture treats flexibility as the target and never forms the two worlds separately:

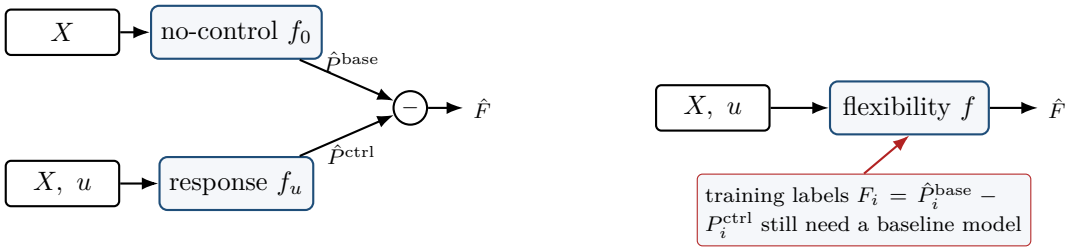
$$\hat{F} = f(X, u), \qquad X = \left[\, T_{\text{out}},\ D,\ N_{\text{avail}},\ T_{\text{in}} - T_{\text{set}},\ \text{event history} \,\right], \tag{12.11}$$

with the proposed action u appended to the feature vector. The features in equation (12.11) are the fleet-state features of Chapter 11: outdoor temperature, the fleet duty cycle, the number of available devices, the margin between indoor temperature and setpoint, and the record of recent dispatches. Operationally this is the easier object. It has one model instead of two, one error to monitor, no cancellation between two large numbers to produce a small one, and the target is the quantity the business actually uses.

The counterfactual problem, however, does not disappear; it moves. To fit equation (12.11) the training set must contain historical values of F, and those values were never metered. Each

one was computed as a difference between a metered controlled load and an *estimated* baseline. The labels of approach B are therefore outputs of exactly the baseline model that approach A makes explicit, with the difference that in approach B the model is hidden inside the data preparation rather than visible in the architecture. Hiding it does not remove its error.

Figure 12.2 draws the two architectures as block diagrams. In panel (a) the state X enters a no-control model f_0 at the top and, together with the action u, a response model f_u at the bottom; the two outputs meet at the subtraction node on the right, whose output is $\hat{F}$. In panel (b) the state and the action enter a single model f whose output is $\hat{F}$ directly; the red box below records where the baseline reappears, in the construction of the training labels. Comparing the panels, approach A places the baseline model on the main signal path, where it can be inspected and validated separately, and approach B places it on the training path, where it silently sets the labels.



(a) forecast both worlds, then subtract

(b) forecast the difference directly

Figure 12.2: The two architectures of equations (12.10) and (12.11). Rounded white boxes are inputs, blue boxes are fitted models, the circle is subtraction. In (a) the baseline model sits on the signal path; in (b) it sits on the training path, inside the construction of the labels, where its error is less visible but no smaller.

12.2.3 Baseline error as label error

The mechanism by which a baseline error damages a response model is worth writing out, because it is not simply a matter of noisier data.

Example 12.1 (A label that is wrong by ten megawatts)**.** An event is dispatched and the fleet's metered consumption during it is $P^{\text{ctrl}} = 75\,\text{MW}$. The baseline algorithm reports $\hat{P}^{\text{base}} = 115\,\text{MW}$, so the recorded flexibility is $\hat{F} = 115 - 75 = 40\,\text{MW}$. Suppose the true, unobserved baseline was in fact 105 MW. Then the true flexibility was $105 - 75 = 30\,\text{MW}$, and the label stored in the training set is wrong by 10 MW, a third of the true value. Nothing in the data indicates this. The meter reading is correct, the arithmetic is correct, and the error lives entirely in a number that cannot be checked.

In general, writing $\varepsilon_{0,i}$ for the baseline error of event i, the recorded label is

$$\hat{P}_i^{\text{base}} = P_i^{\text{base}} + \varepsilon_{0,i}, \qquad \hat{F}_i = \hat{P}_i^{\text{base}} - P_i^{\text{ctrl}} = F_i + \varepsilon_{0,i}. \tag{12.12}$$

Equation (12.12) says that baseline error enters the training set as error in the dependent variable. If $\varepsilon_{0,i}$ has mean zero and is unrelated to anything in X or u, it is *label noise*: it inflates the apparent irreducible error, makes the fitted response look weaker than it is, and demands more data, but the fitted relationship is still centered on the truth. If instead $\varepsilon_{0,i}$ is correlated with the weather, the hour, the fleet state or the way events were selected, it is *label bias*, and the model learns a relationship that is wrong rather than merely imprecise. The distinction between noise and bias is the single most important question to ask about a flexibility data set, and it is a question about the baseline algorithm, not about the response model.

12.2.4 Systematic baseline bias under extreme conditions

Label bias is the normal case rather than the exception, because baselines fail hardest exactly where events are called.

Example 12.2 (Flexibility that appears to vanish in the heat)**.** A regression baseline is fitted on ordinary days and therefore underpredicts load in extreme heat, where air conditioners leave their cycling regime and run continuously. On a very hot afternoon the true baseline is 130 MW and the baseline model reports 120 MW. The dispatched fleet meters 90 MW. The true flexibility is $130 - 90 = 40$ MW; the recorded flexibility is $120 - 90 = 30$ MW. Repeat this across a season and the data set acquires a definite pattern: the hotter the day, the smaller the recorded flexibility. A response model fitted to those labels learns that extreme heat reduces the fleet's responsiveness, which is the opposite of the physics, since extreme heat means more compressors running and therefore more load available to curtail. The aggregator then bids conservatively on the days its capacity is worth most.

The structure of Example 12.2 is confounding in the sense of the chapter's jargon box. Temperature drives the baseline error, temperature drives the recorded flexibility through that error, and temperature is also a feature of the response model, so the model attributes to temperature an effect that belongs to the baseline algorithm. The remedy is not a better response model. It is a baseline whose error does not depend on the conditions under which events are called, or a labeling scheme that does not use a modeled baseline at all, which is the subject of Section 12.4.4.

> **Principle 12.1.** *Baseline error enters a flexibility data set as error in the label. If it is unrelated to conditions it costs precision; if it is related to the conditions under which events are called, it costs correctness, and the response model will be confidently wrong in exactly the operating region that matters.*

12.2.5 Similar-day and regression baselines

Two families of baseline estimator are in common use, and both are descendants of the simple forecasters of Chapter 3.

Similar-day averaging. Choose a small set of recent non-event days judged comparable to the event day and average their load at the same clock time:

$$\hat{P}_t^{\text{base}} = \tfrac{1}{3}\big(P_{t,\text{day }1} + P_{t,\text{day }2} + P_{t,\text{day }3}\big), \tag{12.13}$$

with the three days selected by weekday type, temperature proximity, season and recent consumption level. The method is transparent, is easy to specify in a tariff document, requires no fitting, and is what most utility demand-response programs settle against. Its weakness is that it can only reproduce days that resemble days already seen. If the event day is unusual, which is why it was selected for an event, no average of ordinary days describes it, and the method fails in the direction of Example 12.2.

Weather and calendar regression. Fit the baseline as a function of the drivers instead of selecting analog days:

$$\hat{P}_t^{\text{base}} = \beta_0 + \beta_1 T_t + \beta_2 P_{t-24} + \beta_3 \mathbb{I}_{\text{weekday}}, \tag{12.14}$$

or the same inputs fed to the gradient-boosted trees of Chapter 5, which handle the saturating response of load to temperature far better than the linear term in equation (12.14). A regression baseline interpolates between conditions rather than requiring a matching day, and it is the

better counterfactual whenever the fitted form is adequate. It carries one requirement that is easy to violate: it must be trained on periods that were *not* dispatched, or it must carry the control history as an input. A baseline fitted on all historical data, events included, learns a load shape that already contains the effect of past curtailment, and will then understate the baseline on event-like days because event-like days in its training set were curtailed.

Table 12.1 collects the baseline methods of this section together with the two experimental alternatives introduced later, each with the assumption it needs and its characteristic failure. The table is ordered roughly by increasing strength of evidence, and the pattern in it is that the methods get stronger as they rely less on a model and more on a contemporaneous observation.

Table 12.1: Ways of supplying the unobserved $Y(0)$, ordered by increasing strength of evidence.

Method	What plays the role of $Y(0)$	What it assumes, and how it fails
Similar-day average, equation (12.13)	mean load of three comparable non-event days	assumes the event day resembles ordinary days; fails when the event day is extreme, which is when events are called
Weather regression, equation (12.14)	a fitted function of temperature, calendar and lagged load	assumes the fitted form extrapolates; fails if trained on dispatched periods, or outside the temperature range it saw
Pre-event adjustment, equation (12.15)	a modeled baseline shifted by the recent observed offset	assumes pre-event behavior is unaffected by the coming event; fails under pre-cooling
Matched historical days	load on days matched on state and weather but not dispatched	assumes matching captures every confounder; fails on unmeasured ones
Holdout population, Section 12.4.4	metered load of devices withheld from the dispatch	assumes the holdout is representative; costs deliverable capacity
Randomized control group, Section 12.4.4	metered load of a randomly assigned untreated arm	assumes only randomization and adequate sample size; costs capacity and program complexity

12.2.6 Pre-event adjustment and anticipation effects

A modeled baseline can be corrected using the hours immediately before the event. Suppose the baseline model predicts $\hat{P}^{\text{base}}(17{:}00) = 110\,\text{MW}$, and that over the three hours before the event the same model ran consistently 5 MW below the metered load. The natural correction is to carry that offset forward:

$$\hat{P}^{\text{base,adj}}(17{:}00) = \hat{P}^{\text{base}}(17{:}00) + 5\,\text{MW} = 115\,\text{MW}. \tag{12.15}$$

This is called baseline adjustment, or pre-event scaling when the correction is multiplicative rather than additive. Its justification is sound: the historical baseline describes typical behavior, the current day may not be typical, and the most recent evidence about today is the load observed an hour ago. Nearly every demand-response settlement rule includes an adjustment of this form, usually capped at some percentage.

The danger is that the adjustment window is not necessarily free of the treatment. If customers know an event is coming and pre-cool their homes, or if the aggregator itself pre-cools the fleet to bank thermal energy, the load in the adjustment window is *higher* than it would

otherwise have been. Equation (12.15) then shifts the baseline upward by an amount caused by the event, the counterfactual is inflated, and the measured reduction is larger than the reduction actually delivered. The failure is not a modeling error but a conceptual one: an adjustment taken from a period that the treatment has already affected is no longer an adjustment for untreated conditions. Pre-event behavior must be checked for anticipation before it is used to correct a baseline, and anticipation is itself a response to the intervention and therefore part of the effect being measured.

12.3 Response as a function of action and state

The second term of equation (12.9) has its own structure. This section establishes that the controlled load depends continuously on what is commanded, that it depends on the physical condition of the fleet when the command arrives, and that it depends on what was commanded earlier, so that flexibility behaves like a stored quantity rather than like a rate.

12.3.1 Dose-response dependence on the action

Equation (12.4) wrote $u \in \{0, 1\}$ for clarity. Real dispatch decisions are not binary. A thermostat command is a vector,

$$u = [\, \Delta T_{\text{set}},\ \text{start time},\ \text{duration},\ \text{cohort} \,], \tag{12.16}$$

and over a horizon the fleet receives a whole sequence of such commands, so the controlled load is

$$P_{t+h} = f(X_t,\ u_{t:t+h}). \tag{12.17}$$

Equation (12.17) is a *dose-response* relation: the outcome varies with the strength and duration of the action and not only with its presence.

Example 12.3 (Two actions, two responses)**.** Action A raises setpoints by 1°C for thirty minutes. Action B raises them by 3°C for two hours. These are not the same intervention with different labels. Action A removes a modest fraction of the running compressor load for a short window and leaves indoor temperatures almost unchanged. Action B removes most of it, holds the removal long enough for indoor temperatures to drift materially, and produces a substantial recovery afterward. The flexibility delivered, the energy shifted, the rebound, the comfort cost and the override rate all differ. Reporting a single number for "the fleet's flexibility" without naming the action is therefore incomplete, and the quantity to estimate is $F(X, u)$, not $F(X)$.

Example 12.3 matters most for the optimization of Chapter 14. An optimizer chooses among candidate actions, so it needs the response as a function over the action space. A model that returns one number cannot be optimized over; it can only be reported.

12.3.2 State dependence of the response

The same action applied to two devices in different physical conditions produces different responses, and the difference can be the whole response.

Consider two homes that both receive a 2°C setpoint increase. Home A has indoor temperature $T_{\text{in}} = 22$°C against a setpoint $T_{\text{set}} = 22$°C, and its air conditioner is running. Home B has $T_{\text{in}} = 21$°C against the same 22°C setpoint, and its air conditioner is off because the house is already cooler than it needs to be. Raising A's setpoint to 24°C stops a compressor that was drawing power, so the shed is immediate and equal to that compressor's draw. Raising B's setpoint to 24°C stops nothing, because nothing was running; B contributes approximately zero

at the moment of dispatch and will contribute nothing until its indoor temperature rises past the new setpoint, which under the new setpoint may be a long time. The response is therefore

$$F = f(x_t, u), \tag{12.18}$$

where x_t is the physical state of the device or fleet: indoor temperature, setpoint, whether the compressor is running, thermal mass, recent duty cycle. This is why the fleet-state features of equation (12.11) are not optional extras. Without them the model is averaging over the single variable that decides whether a command produces a response at all.

12.3.3 The state-space view and system identification

Equations (12.17) and (12.18) together are the standard form of a dynamical system with inputs. Writing x_t for the latent physical state, u_t for the control and w_t for the disturbances,

$$x_{t+1} = g(x_t, u_t, w_t), \qquad y_t = h(x_t), \tag{12.19}$$

with g the state transition and h the observation map from state to metered power. Forecasting the fleet's response to a candidate control sequence means learning g and h well enough to roll equation (12.19) forward under that sequence. Learning g and h from input-output data is *system identification*, a discipline older than machine learning and with its own tools.

The practical consequence is a change in what the job is. Flexibility forecasting is the composition of four problems: forecasting, because the weather and the uncontrolled part of the load must be predicted; causal response estimation, because the quantity of interest is an effect of an action; system identification, because the response depends on a latent state with its own dynamics; and control, because the action is chosen rather than given. Each of the four has a literature, and a flexibility forecaster that treats the task as tabular regression on a wide feature matrix is solving one of the four and assuming the other three away.

12.3.4 Rebound and the controlled trajectory

Return to Figure 12.1 and look after 18:00. During the curtailment the fleet consumed less than its baseline, so indoor temperatures drifted upward; the heat the air conditioners did not remove is still in the buildings. When the setpoints are released, the compressors restart, many of them simultaneously, and run until the houses are back at their original temperatures. The fleet therefore consumes *more* than its baseline for a period after the event. That excess is *rebound*, sometimes called payback or snapback, and it is not a side effect to be noted and ignored: it is the direct physical consequence of having moved stored thermal energy in time.

Example 12.4 (One hour down, two hours up)**.** A fleet delivers a 40 MW reduction for one hour. After the event it draws 20 MW more than its baseline for two hours. Whether this is a good outcome depends entirely on the grid's problem. If the system peak is a single hour and the following hours are comfortable, the event solved the problem. If the system is capacity-constrained for four hours, the event moved 40 MWh out of the first hour and put it back into the next two, possibly creating a new peak at 19:00 where none existed. The operator cares about both numbers, so the forecast must deliver both.

Example 12.4 determines the shape of the output. The useful object is not the scalar F_t but the whole controlled trajectory,

$$P^{\text{ctrl}}_{t:t+H} \mid u_{t:t+H}, \tag{12.20}$$

against the corresponding baseline trajectory, so that the reduction and the recovery are both visible. This is a multi-horizon forecasting problem of the kind Chapter 7 and Chapter 8 addressed, with the additional requirement that the horizon extend past the end of the event, since a model whose horizon stops at 18:00 cannot see the rebound at all.

12.3.5 Curtailed energy and shifted energy

The two shaded areas of Figure 12.1 are the two energies that distinguish a genuine reduction from a rescheduling:

$$E_{\text{shed}} = \int_{\text{event}} (P^{\text{base}} - P^{\text{ctrl}})\,\mathrm{d}t, \qquad E_{\text{rebound}} = \int_{\text{post}} (P^{\text{ctrl}} - P^{\text{base}})\,\mathrm{d}t. \tag{12.21}$$

If $E_{\text{rebound}} \approx E_{\text{shed}}$, as in the figure, essentially no energy was saved and the event moved consumption in time. If E_{rebound} is much smaller, some of the service was lost as heat gained by the buildings and never recovered, which is a genuine energy reduction paid for in comfort.

Neither case is automatically better. A grid service that shifts energy out of the peak hour is exactly what capacity markets and congestion management pay for, and a battery does the same thing with explicit losses. But the distinction must be stated rather than left implicit, because the two cases settle differently, have different emissions consequences, and are worth different amounts under different market products. Reporting F during the event and nothing else makes the two indistinguishable.

12.3.6 Trajectory dependence

Because the action changes the state and the state determines the response, the flexibility available at one hour depends on what was done in the hours before it. The question "how much can the fleet shed at 18:00" has no answer until the 17:00 action is specified:

$$F_{t+h} = f(x_t,\ u_{t:t+h}). \tag{12.22}$$

Example 12.5 (Spending flexibility at 17:00)**.** A fleet has $F_{17:00} = 40\,\text{MW}$ available. Two plans are considered. Under the first, nothing is dispatched at 17:00; indoor temperatures stay at setpoint, the compressors keep cycling, and by 18:00 the fleet has $F_{18:00} \approx 45\,\text{MW}$ available, slightly more than at 17:00 because the afternoon is still warming and more connected load is running. Under the second, the full 40 MW is dispatched at 17:00; by 18:00 the buildings have warmed, part of the fleet is at the top of its comfort band, some customers have overridden, and the compressors that remain must run to recover, so $F_{18:00} = 25\,\text{MW}$. The same fleet, the same weather, the same hour: the available flexibility differs by 20 MW because of a decision taken sixty minutes earlier.

Example 12.5 is what makes the resource *state-dependent* rather than merely weather-dependent. A dispatch decision is not a draw from a renewable pool; it is a withdrawal.

12.3.7 Thermal headroom and the battery analogy

The closest familiar analog is a battery. A battery's state of charge SOC_t obeys a known recursion, discharging now reduces the energy available to discharge later, and every optimizer that schedules batteries carries SOC as a state variable with explicit bounds. A thermostat fleet behaves the same way with a less explicit state. Call it *thermal headroom*: the amount by which indoor temperatures may rise before customers are outside their comfort band. A crude scalar version for one home is

$$H = T_{\max} - T_{\text{in}}, \tag{12.23}$$

with $T_{\max}$ the top of the acceptable band, and the fleet version is some distributional summary of the per-home values, since a fleet with every home at 0.5°C of headroom is a different resource from one in which half the homes have 1°C and half have none. Headroom evolves under dispatch and recovers afterward,

$$H_{t+1} = g(H_t,\ u_t,\ T_t,\ \dots), \tag{12.24}$$

which is equation (12.19) restricted to the one state variable that limits the service. When H is large there is a lot of flexibility available; when H is small there is little room to move, whatever the connected load. Flexibility is constrained by state in precisely the way battery flexibility is constrained by state of charge, and the analogy is useful because it tells the forecaster what to instrument: not only the load, but the quantity whose depletion ends the service.

Figure 12.3 plots headroom through an event. The horizontal axis is clock time from 17:00 to 20:00; the vertical axis is fleet-average thermal headroom in degrees Celsius. The gray band is the one-hour event. The curve starts at 2.0°C, falls while the setpoints are raised because the buildings are absorbing heat that is no longer being removed, reaches 1.33°C at the end of the event, and then recovers as the air conditioners restart and pull the temperatures back down. The two annotations tie the state to the service: the flexibility the fleet can offer at the start of the event is 42 MW and at the end of it is 30 MW, the figures of the worked event in Section 12.6.4, and the reduction in available flexibility follows the reduction in headroom. The curve does not return to its starting value within the window plotted, which is the graphical statement of Example 12.5: a dispatch leaves the fleet a weaker resource for some time afterward.

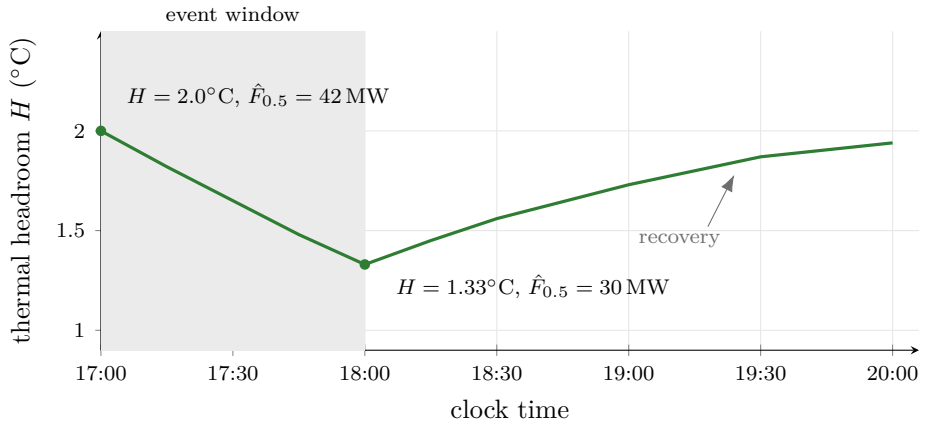


Figure 12.3: Fleet-average thermal headroom (equation (12.23)) through a one-hour curtailment, with illustrative values. The trajectory is generated by the recursion $H_{k+1} = H_k - \alpha\, u_k\, \Delta t + \beta\,(H_{\max} - H_k)\,\Delta t$ of equation (12.24), with $u_k = 2$°C during the event and $u_k = 0$ after it, $\Delta t = 15$ minutes, $\alpha = 1.4\,\mathrm{h}^{-1}$, $\beta = 0.9\,\mathrm{h}^{-1}$ and $H_{\max} = 2.0$°C. The annotated flexibilities are those of the worked event of Section 12.6.4.

Principle 12.2. *Flexibility is a function of state and action, not of time. Using it now reduces what is available later, so a flexibility forecast must carry a state variable whose depletion it tracks.*

12.4 Causal estimation at fleet scale

The counterfactual term can be supplied by a model or by an observation on a comparable unit that was not acted on. This section develops the second route, which is the stronger one, and the obstacles the first route faces: heterogeneous response across devices, confounding in historical dispatch records, and the cost of deliberately withholding capacity in order to learn.

12.4.1 Individual and conditional treatment effects

Define, for device i,

$$\tau_i = Y_i(0) - Y_i(1), \tag{12.25}$$

the reduction that this specific thermostat would provide if dispatched. The individual treatment effect of equation (12.25) is not identifiable: it needs both potential outcomes for the same device at the same moment, and equation (12.4) supplies only one. What can be estimated is its conditional mean, the *conditional average treatment effect*,

$$\mathrm{CATE}(X) = \mathbb{E}[\, Y(0) - Y(1) \mid X \,], \tag{12.26}$$

the average reduction among devices whose observable characteristics are X. Equation (12.26) is a weaker object than equation (12.25) and is sufficient for the decision at hand, because dispatch is applied to groups of devices selected by their characteristics.

Heterogeneity in equation (12.26) is what makes targeting worthwhile. Suppose device A has $\mathbb{E}[\tau_A] = 0.1\,\mathrm{kW}$ and device B has $\mathbb{E}[\tau_B] = 1.5\,\mathrm{kW}$, at a comparable comfort cost to the customer. Device B is fifteen times as useful per dispatch, and a fleet of a hundred thousand devices contains large populations of both. An aggregator that can rank devices by their conditional expected response can deliver a required megawatt figure while dispatching a fraction of the fleet, which reduces comfort impact, override risk and the depletion of Section 12.3.7. Estimating heterogeneous response is therefore not an academic refinement of an average; it is what converts a flexibility forecast into a dispatch list.

12.4.2 Observational bias and confounding

The obvious way to estimate a treatment effect from history is to compare dispatched and non-dispatched records. That comparison is valid only if the two groups were otherwise alike, and in an operational data set they are not, because the aggregator chose who to dispatch. If it historically called the devices that looked most responsive, or the cohorts with the best connectivity, or the customers who had not recently opted out, then the action u is a function of the characteristics X, the treated and untreated populations differ systematically, and

$$\mathbb{E}[\, Y \mid u = 1 \,] - \mathbb{E}[\, Y \mid u = 0 \,] \tag{12.27}$$

measures the difference between two unlike populations plus whatever effect the action had. Equation (12.27) is an association, not a causal effect.

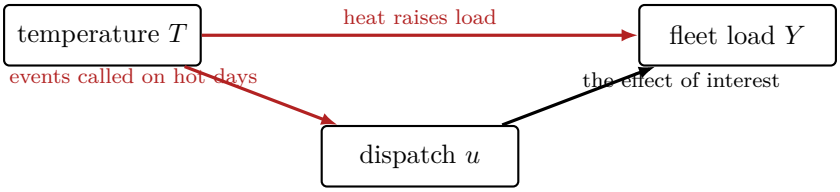
Example 12.6 (Dispatch appears to increase load)**.** An aggregator calls events almost exclusively on very hot days. Over a season, mean fleet load on event days is 90 MW and on non-event days 70 MW. Taken at face value, equation (12.27) says that dispatching the fleet *increases* its load by 20 MW. The conclusion is absurd and the arithmetic is correct. Temperature drives both: it drives the decision to dispatch, and it drives load. The 20 MW is the temperature difference between event and non-event days, partially offset by the curtailment, and nothing in the comparison separates the two.

Figure 12.4 draws the causal structure of Example 12.6. The three boxes are the variables: outdoor temperature T on the left, the dispatch decision u at the bottom, and the fleet load Y on the right. The black arrow from u to Y is the effect of interest, the one the aggregator is paid for. The two red arrows, from T to u and from T to Y, are the confounding path: temperature determines whether an event is called and independently determines how much load there is to begin with. The naive comparison of equation (12.27) runs along all three arrows at once and reports their sum. Blocking the red path, either by conditioning on T or by breaking the arrow from T to u through randomization, is the whole content of the next two subsections.

12.4.3 Conditional comparison

The first remedy is to compare like with like. Condition on the variables that drive both the action and the outcome, and take the difference within each condition:

$$\tau(X) = \mathbb{E}[\, Y \mid u = 0,\ X \,] - \mathbb{E}[\, Y \mid u = 1,\ X \,], \qquad X = [\, T,\ \text{hour},\ \text{device state},\ \ldots \,]. \tag{12.28}$$



Event days: 90 MW. Non-event days: 70 MW. The naive difference of equation (12.27) reports +20 MW, which is the sum of all three arrows.

Figure 12.4: The confounding structure of Example 12.6. Temperature influences both the dispatch decision and the load, so the association between dispatch and load contains the temperature effect. The black arrow is the causal effect the aggregator needs; the red arrows are the path that must be blocked by conditioning or broken by randomization.

Within a narrow stratum of temperature, hour and fleet state, dispatched and non-dispatched records are more nearly comparable, and the difference in equation (12.28) is closer to a causal effect than the difference in equation (12.27). This is what a flexible regression with a treatment indicator does implicitly, and it is what matched-day baselines do explicitly.

The method rests on an assumption that cannot be verified from the data: that X contains every variable influencing both u and Y. If events were also called more often when the operator judged the fleet to be in good condition, and that judgment used information not recorded in X, the conditional comparison remains biased by the unrecorded part. There is no diagnostic for a confounder that was never measured. Conditioning is a genuine improvement and not a solution, and the honest summary is that it converts an obviously wrong estimate into a possibly wrong one.

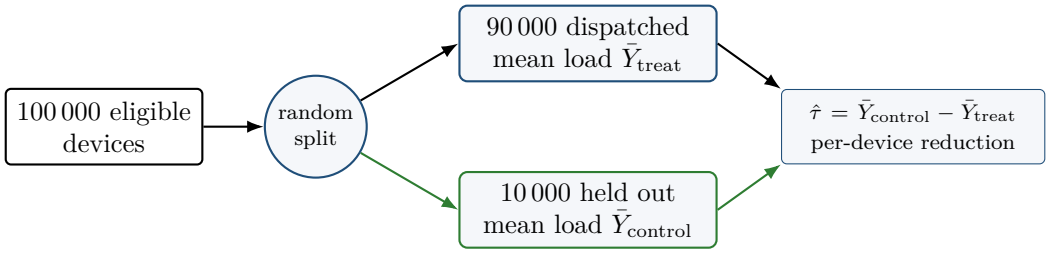
12.4.4 Randomized experiments and holdout control groups

The second remedy removes the confounding path rather than adjusting for it. If the decision of which devices to dispatch is made at random among eligible devices, then by construction u is independent of X, the red arrow from T to u in Figure 12.4 does not exist, and the untreated group's metered load is an unbiased estimate of the treated group's counterfactual under the same weather, at the same hour, in the same conditions.

Two arrangements are used. In a *randomized experiment*, similar devices are assigned to treatment and control: 50 000 are dispatched and 10 000 comparable devices are deliberately left untreated. The control group's metered consumption is then a contemporaneous measurement of P^{base} per device, which is a far stronger object than any historical baseline because it experiences the same weather forecast error, the same unexpected humidity and the same regional events as the treated group. In a *holdout control group*, the same idea is expressed as a split of an operational dispatch: of 100 000 eligible devices, 90 000 receive the command and 10 000 are held out, giving both the controlled response and its baseline from one event.

Figure 12.5 draws the holdout split. The box on the left is the eligible population of 100 000 devices. The randomization node divides it into the dispatched arm of 90 000 devices, whose mean per-device load is $\bar{Y}_{\text{treat}}$, and the holdout arm of 10 000 devices, whose mean per-device load $\bar{Y}_{\text{control}}$ plays the role of the baseline. The two means meet at the estimator on the right. The annotation beneath records what the arrangement costs: the held-out devices deliver nothing during the event, so a tenth of the fleet's capacity is spent on measurement.

The opportunity cost. The arrangement is not free, and the cost is immediate and quantifiable. Suppose the utility needs 75 MW and the full fleet can deliver 80 MW. Holding out ten



Same weather, same hour, same conditions for both arms. Cost: the held-out tenth of the fleet delivers no capacity during the event.

Figure 12.5: A holdout control group. Randomization severs the arrow from temperature to dispatch in Figure 12.4, so the holdout arm's metered load is a contemporaneous measurement of the counterfactual rather than a model of it.

percent of the devices leaves roughly 72 MW deliverable, and the commitment is missed. The holdout fraction therefore trades information against delivered value: a larger holdout gives a tighter estimate of the counterfactual and leaves less capacity for the market, and a smaller one does the reverse. This is the exploration-and-exploitation trade-off in its operational form, and it is usually resolved by holding out a large fraction on days when the system is not stressed, a small one or none at all when it is, and never a fixed percentage regardless of conditions.

12.4.5 Experimentation as dual control

Randomization can be applied to the action as well as to the assignment. An aggregator that occasionally varies the setpoint change, the duration or the device subset at random, instead of always choosing the action its current model recommends, collects observations spread across the action space and learns the response surface $F(X, u)$ rather than only the one slice of it that its policy happens to visit. Each such event delivers two things: the grid service itself, and information about the response to an action the model is uncertain about.

Controlling a system while deliberately injecting excitation in order to identify it is known in control theory as *dual control*, and the name is worth keeping because it states the design intent: the dispatch policy has two objectives, and treating the second as a nuisance is what produces a model that can only evaluate the actions already in use. The connection to Section 12.5.5 is direct: deliberate exploration is how a model acquires support in regions of the action space the optimizer will eventually want to enter.

12.4.6 Direct response modeling and the response surface

With credible labels in hand, the object to fit is the response as a function of state and action,

$$F = f\big(T_{\text{out}},\ T_{\text{in}},\ D,\ N_{\text{avail}},\ \Delta T_{\text{set}},\ \text{duration}\big), \tag{12.29}$$

so that for any candidate action u the model returns $\hat{F}(u)$ and the optimizer of Chapter 14 can compare actions rather than receive one.

The shape of equation (12.29) in the action direction is strongly nonlinear. Table 12.2 gives an illustrative slice: the expected fleet reduction as a function of the setpoint increase, holding state fixed. The first column is the setpoint increase, the second the expected reduction, and the third the average reduction gained per additional degree over the preceding interval. The third column is the informative one: it falls from 20 MW per degree at the start to 5 MW per degree between two and three degrees. The reason is physical *saturation*. A setpoint increase sheds load by switching off compressors that are running; once most of the ones that can be switched

off have been, a further increase has little left to act on, and the remaining load belongs to devices that must keep running for reasons the setpoint does not control.

Table 12.2: An illustrative slice of the response surface equation (12.29): expected fleet reduction against setpoint increase, fleet state held fixed. The third column shows saturation.

Setpoint increase ΔT_{set}	$\mathbb{E}[F]$ (MW)	Marginal gain over the preceding interval
0.5°C	10	20 MW per °C
1.0°C	18	16 MW per °C
2.0°C	30	12 MW per °C
3.0°C	35	5 MW per °C

Figure 12.6 plots the same slice. The horizontal axis is the setpoint increase in degrees Celsius from zero to four; the vertical axis is the expected fleet reduction in megawatts. The four circles are the values of Table 12.2 and the solid curve is a saturating function fitted through them, whose formula is given in the caption. The dashed horizontal line is the asymptote, the reduction that no setpoint increase can exceed because the load it would have to remove is not switchable by a thermostat. The curve's concavity is the whole operational content of the figure: the first degree is worth far more than the third, so an optimizer facing a required reduction should prefer a shallow action on many devices to a deep action on few, other things being equal. A linear model fitted to these points would recommend the opposite and would promise a reduction at four degrees that the fleet cannot deliver. Capturing the curvature is what the gradient-boosted trees of Chapter 5 and the neural models of Chapter 7 are for.

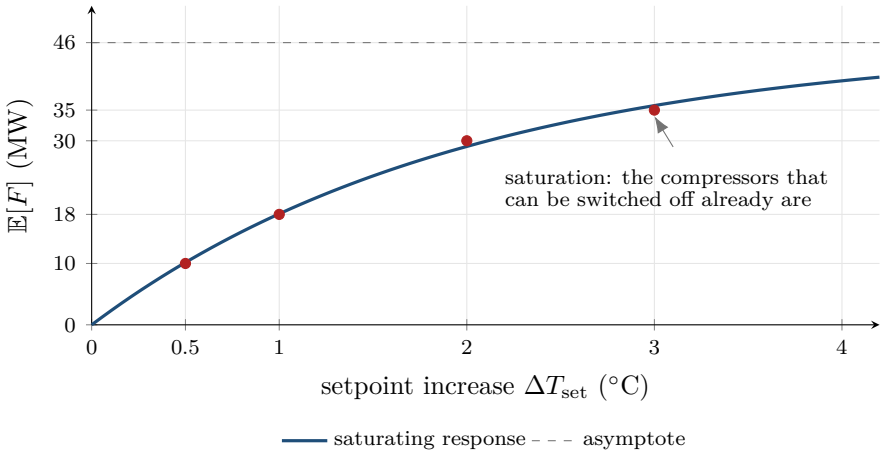


Figure 12.6: The response surface of Table 12.2 in the action direction. The red markers are the four illustrative points; the solid curve is $\mathbb{E}[F] = 46\,(1 - e^{-\Delta T_{\text{set}}/2})$ MW, fitted to them, and the dashed line is its asymptote of 46 MW. The marginal reduction per additional degree falls monotonically, which is the saturation of Section 12.4.6.

12.4.7 Comfort cost and override probability

The response surface is only half of the dispatch problem. Every action that sheds load also imposes a cost on the customer, and the optimizer must trade the two. Writing $C_{\text{comfort}}(u)$ for the comfort cost of action u and $V_{\text{grid}}(\cdot)$ for the value of the reduction delivered, the decision is

$$\max_{u \in \mathcal{U}} \Big[V_{\text{grid}}(F(u)) - C_{\text{comfort}}(u) \Big], \tag{12.30}$$

over the admissible action set $\mathcal{U}$. Forecasting the response supplies the first term of equation (12.30) and says nothing about the second, so a flexibility forecaster that reports only the megawatts available has answered half the question it is asked.

The customer's reaction also feeds back into the physics, through the override. A thermostat program lets the occupant cancel an event, and the probability that they do so,

$$\Pr(\text{override} \mid u,\ X), \tag{12.31}$$

rises with the intensity and duration of the action and depends on the household's history and the conditions. Equation (12.31) makes the response surface of Figure 12.6 turn over at large u even in the aggregate: a more aggressive action yields more flexibility in the first minutes and then loses devices as occupants cancel, and over a season it loses them permanently through opt-out and attrition. A generator's response to a dispatch instruction has no analog of equation (12.31); a thermostat fleet's does, which is what makes behavioral dynamics part of the model rather than a footnote to it.

12.5 Uncertainty and the interventional distribution

Flexibility is uncertain for two reasons at once, and the two combine in a way that a naive addition of variances gets wrong. This section gives the distributional object the optimizer actually needs, decomposes its uncertainty, and then states the condition under which an association learned from history may be read as the effect of an intervention.

12.5.1 The distribution of the controlled response

Chapter 10 argued that a point forecast is insufficient for a decision under asymmetric costs, and a dispatch decision is such a decision: underdelivering against a market commitment carries a penalty that overdelivering does not refund. The object to produce is therefore a distribution conditional on both the state and the action,

$$p(P^{\mathrm{ctrl}} \mid X,\ u) \qquad \text{or equivalently} \qquad p(F \mid X,\ u). \tag{12.32}$$

Conditioning on u changes the shape of the distribution and not only its location. Suppose that for a given fleet state,

$$u = 1^{\circ}\mathrm{C}: \quad F \sim \mathcal{N}(20,\ 3^2), \qquad u = 3^{\circ}\mathrm{C}: \quad F \sim \mathcal{N}(35,\ 10^2), \tag{12.33}$$

in megawatts. The stronger action has a higher expected reduction, by 15 MW, and a standard deviation more than three times larger, because a deep setpoint change pushes more of the fleet into regions where overrides, comfort limits and equipment differences dominate. An optimizer required to deliver 18 MW firm will compare lower quantiles rather than means: under the first action the tenth percentile is $20 - 1.28 \times 3 = 16.2$ MW, and under the second it is $35 - 1.28 \times 10 = 22.2$ MW, so the aggressive action still wins. If instead the requirement were a 90 % probability of delivering at least 16 MW, the first action meets it with a much smaller comfort cost. The point is that the comparison cannot be made on $\mathbb{E}[F]$ alone, and a stronger action is not automatically the preferred one.

12.5.2 Two layers of counterfactual uncertainty

The uncertainty in equation (12.32) has two distinguishable sources. The first is *baseline uncertainty*: the no-intervention load P^{base} is not observed, so its value is uncertain even in hindsight. The second is *response uncertainty*: the controlled load P^{ctrl} is itself stochastic, because overrides, occupancy, equipment state and weather realization vary. Both propagate

into the difference, and because flexibility is a difference the propagation includes a covariance term:

$$\mathrm{Var}(F) = \mathrm{Var}\big(P^{\mathrm{base}}\big) + \mathrm{Var}\big(P^{\mathrm{ctrl}}\big) - 2\ \mathrm{Cov}\big(P^{\mathrm{base}},\ P^{\mathrm{ctrl}}\big). \tag{12.34}$$

Equation (12.34) is the same identity that Chapter 1 used to warn against treating the two terms as independent, and the covariance is the term most often dropped.

It should not be dropped, because the two terms are strongly and positively dependent by construction. They are driven by the same weather, the same calendar, the same fleet and the same availability. If the afternoon turns out hotter than forecast, the baseline load is higher than predicted and the controlled load is also higher than predicted, because the same compressors are working harder in both worlds. $\mathrm{Cov}(P^{\mathrm{base}}, P^{\mathrm{ctrl}}) > 0$ is therefore the normal case and the covariance may be a large fraction of the two variances.

Example 12.7 (The covariance term is not a correction)**.** Let P^{base} have standard deviation 8 MW and P^{ctrl} standard deviation 6 MW, with correlation 0.7, so $\mathrm{Cov} = 0.7 \times 8 \times 6 = 33.6\,\mathrm{MW}^2$. Then equation (12.34) gives

$$\mathrm{Var}(F) = 64 + 36 - 2 \times 33.6 = 32.8\ \mathrm{MW}^2, \qquad \sigma_F = 5.7\,\mathrm{MW}.$$

Assuming independence would have given $\sqrt{64 + 36} = 10.0\,\mathrm{MW}$, overstating the uncertainty of the flexibility by three quarters. With a negative correlation the error runs the other way. Either way the covariance is not a refinement of the answer; it is a term of the same size as the ones retained.

Modeling the difference directly. Example 12.7 is an argument for approach B of Section 12.2.2. A model fitted directly to F produces a predictive distribution for the difference, in which the shared component of the two uncertainties has already cancelled, instead of requiring the forecaster to estimate two marginals and a correlation and then combine them. The quantile regression and distributional methods of Chapter 10 apply unchanged to F as a target. The price is the one already paid in Section 12.2.3: the labels must be trustworthy, which returns the problem to how they were constructed.

12.5.3 Predictive and interventional forecasting

The distinction the chapter has been circling can now be stated in one line. A predictive forecast is

$$p\big(y_{t+h} \mid X\big), \tag{12.35}$$

the distribution of the target given what is observed, which answers "what will happen". An interventional forecast is

$$p\big(y_{t+h} \mid \mathrm{do}(u)\big), \tag{12.36}$$

the distribution of the target when the action is *set* to u, which answers "what will happen if the fleet is commanded to do u". The $\mathrm{do}(\cdot)$ notation of causal inference exists precisely to mark the difference between observing a value of u and forcing one, and it is worth adopting because the two coincide so rarely in operational data.

They differ whenever the historical choice of u carried information about the conditions. Suppose a three-degree setpoint change was used only on extremely hot days. Then $p(Y \mid u = 3°\mathrm{C})$ is the distribution of load on extremely hot days under a strong action; it includes the effect of the heat, because the heat is why the action was chosen. By contrast $p(Y \mid \mathrm{do}(u = 3°\mathrm{C}))$ asks what happens if a three-degree change is applied to the fleet in its *current* state, which may be a mild evening. Conditioning on an observed action reads the action as evidence about the world; intervening sets the action and leaves the world as it is. A model fitted to historical data learns the first and is usually asked for the second.

12.5.4 Scope of causal validity, and historical policy bias

Causal machinery is not required for every question, and claiming otherwise overstates the case. If the dispatch policy is stable, if the actions under consideration are the ones historically used, and if the associations in the data generalize to the conditions of interest, then a predictive model fitted to historical event data is useful and will perform well. Many production forecasting systems operate in exactly this regime, and they are not wrong to.

Causal validity begins to matter under two kinds of question. The first is "what would happen under an action not usually taken", which has no answer in a data set that never took it. The second is "which action should be chosen", because a comparison across actions requires each action's effect to be separated from the conditions under which it was historically applied. Both questions arrive as soon as a forecast is connected to an optimizer, and the direction of travel through Chapter 14 is toward precisely those questions.

Historical policy bias. The characteristic failure is worth naming because it is easy to produce. Suppose the historical policy applied weak actions on mild days and strong actions on severe days, which is the sensible operating practice and therefore the common pattern. The feature "strong control" is then almost perfectly aligned with the feature "severe conditions", and the strongest signal in the data is that strong control coincides with high load. A model fitted without regard to how the actions were assigned may learn that strong control causes large load, rather than that strong control causes a large reduction from an even larger baseline. The resulting response surface can be monotone in the wrong direction, and an optimizer handed it will avoid exactly the action the physics recommends.

12.5.5 Action-space shift and support

Chapter 13 treats distribution shift in the inputs. A counterfactual response model is exposed to a second kind of shift that has nothing to do with the weather: shift in the *action* space. Suppose every event in the training set used a setpoint change of at most 2°C, and the optimizer, searching for a way to meet a large requirement, proposes 4°C. The state X may be entirely familiar, the weather ordinary, the fleet in a well-observed condition, and the model is still extrapolating, because it has never seen the action. A response model must therefore be able to report two separate kinds of familiarity: whether X is familiar, and whether u is familiar.

The precise condition has a name. For the effect of action u under conditions X to be estimable from observational data, the data must contain cases in which that action was applied under those conditions:

$$p(u \mid X) > 0 \quad \text{for the } (X, u) \text{ pairs of interest,} \tag{12.37}$$

which is called *support*, *overlap* or *positivity*. Where equation (12.37) fails, no amount of modeling recovers the effect; the model's output there is an extrapolation of its functional form and nothing more. A concrete instance for a thermostat fleet: if aggressive setpoint changes were never applied when indoor temperatures were already near the comfort limit, because the operator sensibly refused to, then $F(X, u)$ in that corner is unconstrained by evidence. The model will still return a number, often a confident one, and the optimizer should not act on it. The practical requirement is that the response model carry a support indicator alongside its prediction, and that the optimizer treat a prediction outside support as unavailable rather than as merely uncertain.

12.6 Architecture, validation and the worked event

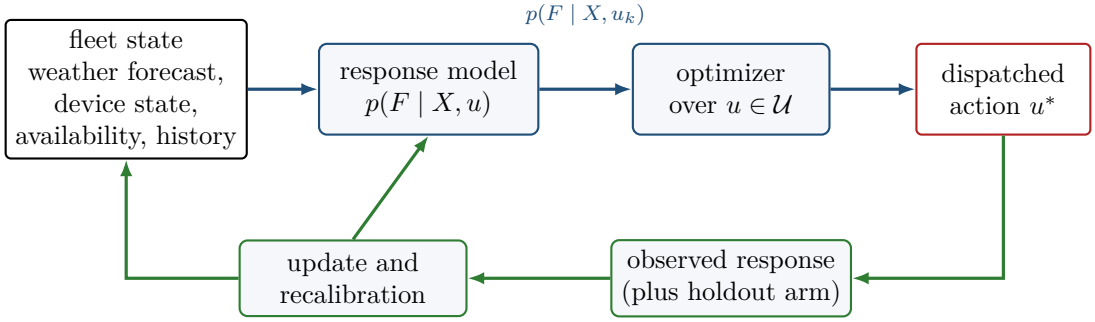
The pieces now assemble into a system. This section gives the loop a production flexibility forecaster runs, the validation strategies available when the counterfactual is unobservable, a

worked event with numbers, and the hierarchy of problems the rest of the book climbs.

12.6.1 A practical architecture

The system is a closed loop rather than a pipeline. Its stages are: the current fleet state, assembled from the weather forecast, device telemetry, availability and the recent control history; a response model that maps that state and a candidate control to a distribution $p(\text{response} \mid \text{state},\ \text{candidate control})$; an optimizer that evaluates candidate controls against the objective of equation (12.30) and selects one; the dispatch of the chosen control to the fleet; the observation of the realized response; and the update and recalibration of the response model from that observation.

Figure 12.7 draws the loop. The upper row runs left to right, from fleet state through the response model to the optimizer and the dispatched action. The lower row runs right to left, carrying the observed response back into the state estimate and into the model update. Two annotations mark where this chapter's content sits: the response model is where equations (12.29) and (12.32) live, and the return path is where the labeling problem of Section 12.2.3 is either solved by a holdout arm or reintroduced by a modeled baseline. The figure describes a learned dynamic response system under closed-loop control, and the loop is what distinguishes it from a forecasting pipeline: the model's own outputs determine the actions that generate its next training data.



The return path carries the labeling problem: a holdout arm measures the counterfactual, a modeled baseline estimates it.

Figure 12.7: The flexibility forecasting and dispatch loop of Section 12.6.1. Blue arrows run forward from state to action; green arrows carry the realized response back into the state estimate and the model. The model's outputs choose the actions that produce its next training data, which is why the labeling and support questions of Sections 12.2.3 and 12.5.5 are structural rather than incidental.

12.6.2 Validation when the counterfactual is unobservable

Section 12.1.3 established that the error of a counterfactual forecast cannot be computed for the events that matter, since $Y(0)$ and $Y(1)$ are never both observed for the same unit at the same time. Validation must therefore proceed indirectly, and the available strategies differ sharply in how much they assume. Table 12.3 lists them with what each uses in place of the missing outcome, what it assumes, and the strength of the evidence it produces. The ordering is by strength, and the pattern is that evidence strengthens as the method moves from reconstructing the counterfactual by assumption to measuring it on a comparable population.

Modeled baseline against randomized control. The top and bottom of Table 12.3 are worth comparing directly. A modeled baseline sets $\hat{Y}(0) = f(X)$ and costs nothing operationally,

Table 12.3: Validation strategies for a counterfactual forecast, ordered by strength of evidence.

Strategy	What stands in for the missing outcome	What it assumes
Simulation	a physics model of the fleet run under both actions	that the simulator is faithful; it tests the forecaster against a model, not against the world
Matched historical days	non-dispatched days with similar weather and state	that matching captured every confounder
Synthetic control	a weighted combination of non-dispatched units that tracks the treated unit before the event	that the pre-event fit implies a valid post-event counterfactual
Holdout population	metered load of devices withheld from an operational dispatch	that the holdout is representative of the dispatched arm
Randomized control group	metered load of a randomly assigned untreated arm	randomization and adequate sample size only
Prospective experiment	a pre-registered randomized event designed to test the model's prediction	randomization, plus that the model's prediction was fixed in advance

but its error is a model error whose size is unknown precisely where it matters, and it is the mechanism of Example 12.2. A randomized control sets $\hat{Y}(0)$ equal to the observed load of the control group; its error is sampling error, which shrinks with the size of the arm and can be quantified, and it requires no assumption about the functional form of anything. Against that, it consumes deliverable capacity, as Section 12.4.4 quantified, and it requires the arm to be representative, which randomization delivers but which a conveniently-chosen holdout does not. Where both are available, the randomized control is the stronger evidence, and the modeled baseline's role is to interpolate to the events for which no control arm was run.

12.6.3 Event-level randomized validation and labels

The concrete form of a randomized validation at event level is a split of a homogeneous population. Take 20 000 similar thermostats and assign 10 000 to treatment and 10 000 to control. The estimator is the difference of the two arms' mean per-device loads,

$$\hat{\tau} = \bar{Y}_{\text{control}} - \bar{Y}_{\text{treat}}, \tag{12.38}$$

and with arms of this size any systematic difference between the two groups in weather, hour, equipment or occupancy has been averaged away by the randomization, so the only error left in equation (12.38) is sampling error. That is the closest thing to a measured counterfactual the problem admits.

The role of fleet scale. An individual household's air-conditioning load is extremely noisy: a door opened, a guest arriving, a cooking event, and the fifteen-minute average moves by more than the effect being measured. The arm *means* in equation (12.38) are not noisy, because the per-device noise divides by the square root of the arm size exactly as in Chapter 11. With 10 000 devices per arm the standard error of each mean is a hundredth of the per-device standard deviation, and a per-device effect of a fraction of a kilowatt is resolved with room to spare. The same scale that makes individual devices unforecastable makes fleet experiments precise, and

this is why an aggregator is in an unusually strong position to run causal experiments: it has the sample size, the randomization mechanism and the measurement already in place.

Experiments as a label source. The output of an experiment is not only a validation statistic. It is a set of clean training labels: for the conditions and action of that event, a measured flexibility untouched by the baseline bias of equation (12.12). A model fitted to such labels can then be generalized across the fleet and across conditions in which no experiment was run, which is the ordinary function of a statistical model. Experimentation and machine learning reinforce each other here: experiments supply unbiased labels in the regions they cover, and the model interpolates between them and reports where it is extrapolating, which in turn identifies the regions the next experiments should target.

> **Principle 12.3.** *A randomized holdout converts a counterfactual into an observation. It is the only method in Table 12.3 whose error is sampling error rather than model error, and at fleet scale its sampling error is small.*

12.6.4 A worked thermostat event

The objects of the chapter can be read off a single operational decision.

Example 12.8 (A 14:00 decision about a 17:00 event)**.** At 14:00 the aggregator must decide what to offer for the evening. The state vector it assembles is: $N_{\text{avail}} = 80\,000$ devices online and program-eligible; a weather forecast $\hat{T}_{17:00} = 37°\text{C}$; a fleet mean duty cycle of 65 %; and a mean indoor-to-setpoint gap of 0.4°C, so the fleet has thermal headroom but is not sitting idle. The candidate action is $\Delta T_{\text{set}} = 2°\text{C}$ held for two hours from 17:00.

The response model returns a trajectory rather than a number. For the first hour,

$$\hat{F}_{0.5}(17{:}00) = 42\,\text{MW}, \qquad \hat{F}_{0.1}(17{:}00) = 34\,\text{MW},$$

a median reduction of 42 MW with a tenth percentile of 34 MW, so a firm commitment of 34 MW would be met nine times in ten. For the second hour, after the fleet has already been curtailed for an hour,

$$\hat{F}_{0.5}(18{:}00) = 30\,\text{MW},$$

a median 12 MW lower than the first hour for the same action, because the thermal headroom of Figure 12.3 has been partly consumed and part of the fleet has reached the top of its comfort band.

The three numbers are the chapter in miniature. The 42 MW is an interventional quantity, conditional on an action that has not yet been taken. The gap between 42 and 34 is the uncertainty of equation (12.34), which the market commitment must respect rather than the median. And the fall from 42 to 30 is the trajectory dependence of equation (12.22): the same fleet under the same action delivers less in the second hour because of what the first hour did to its state. An optimizer given only 42 MW would commit to two hours of it and underdeliver in the second.

12.6.5 Comparison with ordinary load forecasting

Set the two problems side by side. Ordinary load forecasting predicts

$$P_{t+h} = f\big(\text{weather},\ \text{calendar},\ \text{history}\big), \tag{12.39}$$

and is checked against a meter. Flexibility forecasting predicts

$$P_{t+h} = f\big(\text{state},\ \text{weather},\ \text{history},\ \text{control sequence}\big) \quad \text{compared with} \quad P_{t+h}(u = 0), \tag{12.40}$$

and is checked against an estimate or an experiment. The additional arguments in equation (12.40) are not extra features of the same kind as the ones in equation (12.39). The control sequence is chosen rather than observed, which is what makes the problem interventional; the state is latent and has dynamics, which is what makes it a system identification problem; and the comparison term is counterfactual, which is what makes validation hard.

What the problem rewards. The consequence for practice is that flexibility forecasting rewards a different set of habits from generic tabular prediction. It rewards thinking in physical state, so that the model carries indoor temperature and headroom rather than only load lags; in temporal dynamics, so that the effect of an action persists past the interval in which it is applied; in controls, so that the action is an argument and the output is a surface rather than a point; in counterfactual response, so that labels are interrogated before they are fitted; in uncertainty, so that a lower quantile rather than a mean goes to market; and in operational constraints, so that comfort bands, override rights and availability are part of the model. The work is system modeling with statistical tools inside it, and a practitioner who treats it as feature engineering on a wide table will produce a model that scores well and dispatches badly.

12.6.6 The four-level hierarchy and the move to optimization

The problems of this chapter stack into four levels, and the book's progression is a climb through them. Figure 12.8 draws the stack. The boxes from bottom to top are: Level 1, baseline forecasting, the no-intervention load P^{base}; Level 2, response forecasting, the controlled load $P^{\text{ctrl}}(\cdot \mid u)$; Level 3, flexibility, their difference $F = P^{\text{base}} - P^{\text{ctrl}}$; and Level 4, the decision, the chosen action u^*. The right-hand column names the chapter or section where each level is treated and the left-hand column the character of each problem. Reading upward, each level consumes the one below it: the difference at Level 3 cannot be formed without both of the levels under it, and the decision at Level 4 needs the distribution of the difference rather than its point value. Reading downward, each level constrains the ones below it, since the decision determines which actions the response model will be queried about and therefore which regions of the action space need support in the sense of equation (12.37).

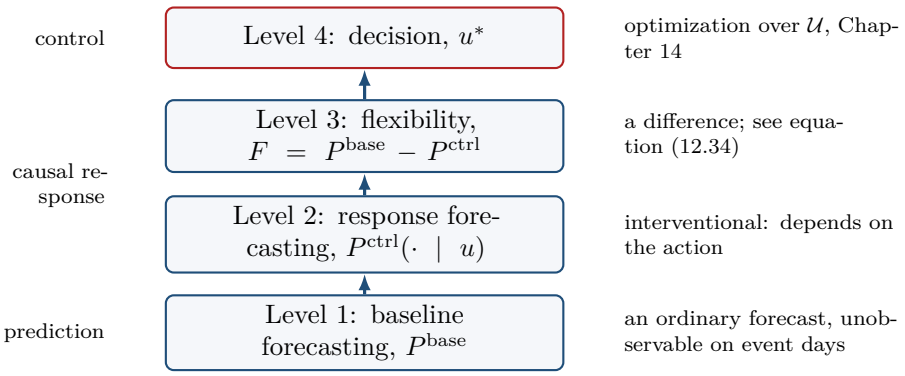


Figure 12.8: The four levels of the flexibility problem. Each level consumes the ones below it; the left-hand labels give the character of the problem at each stage and the right-hand notes the property that makes it distinct. This book moves upward through the stack.

From prediction to optimization. The interface between Level 3 and Level 4 is the distributional object of equation (12.32), evaluated at each of a set of candidate actions $u_1, \ldots, u_K$. The optimizer then chooses the action balancing the response required, the risk of underdelivery against the market commitment, the comfort cost of equation (12.30), the rebound of equation (12.21) and the flexibility left available for the following hours by equation (12.22).

The forecast is no longer a product in its own right; it is a component of a control system, which is the perspective Chapter 14 adopts throughout.

The change in the question. Underneath all of it is one shift. As long as the forecaster is a passive observer, the question is what comes next, and the answer is a distribution conditional on information. As soon as the forecaster's organization takes actions that influence the outcome, the question becomes what comes next if this action is taken, and the answer must be a distribution conditional on an intervention. Every difficulty in this chapter, the unobserved baseline, the biased labels, the confounded history, the missing support, the hard validation, follows from that single change.

12.6.7 Conceptual questions

Four questions on this material recur, and the strong answers are the chapter's four main claims.

Conceptual questions

"How would you forecast thermostat flexibility?"
A strong answer separates the two worlds before describing any model: the no-control baseline and the controlled response are different objects, and flexibility is their counterfactual difference, as in equation (12.9). It then says what the response model is conditioned on, namely the fleet state, the weather forecast, availability, the recent control history and the proposed action, because the response is a function of state and dose and not of time. It raises the labeling question without being prompted, since historical flexibility labels were produced by a baseline algorithm and a biased baseline biases the fitted response, as in Example 12.2. And it ends on validation: randomized or held-out control populations are what make the labels and the calibration credible, because they replace a modeled counterfactual with a measured one.

Conceptual questions

"Why can't you just forecast load and subtract the actual event load?"
Because the load forecast during the event is not a forecast that will ever be checked; it is the unobserved counterfactual of equation (12.6). A strong answer gives the consequence rather than only the definition: a biased baseline produces a biased response estimate through equation (12.12), with Example 12.1's ten megawatts on a thirty-megawatt effect as the scale of the problem. It then adds the point most answers miss, that the subtraction involves two uncertain quantities whose errors are correlated through their shared drivers, so by equation (12.34) the uncertainty of the difference is not the sum of their uncertainties and may be much smaller or much larger.

Conceptual questions

"Why does past dispatch matter to future flexibility?"
Because the intervention changes the physical state. A strong answer traces the mechanism: curtailment lets indoor temperatures drift upward, which consumes the thermal headroom of equation (12.23), which reduces the load available to curtail in the next interval and simultaneously creates the rebound of Section 12.3.4 as the equipment recovers. It then states the general property, that flexibility is state- and trajectory-dependent, $F_{t+h} = f(x_t, u_{t:t+h})$, and gives the magnitude from Example 12.5: spending 40 MW at 17:00 leaves 25 MW at 18:00 where doing nothing would have left about 45 MW. The battery analogy closes it: dispatch is a withdrawal from a stored quantity, not a draw on a renewable one.

Conceptual questions

"Would you train directly on historical event response?"
Possibly, but only after examining how the labels were constructed. A strong answer explains the examination it would do: labels formed from an estimated baseline carry that estimate's bias, and bias correlated with temperature or with event-selection criteria corrupts the fitted relationship rather than merely adding noise. It prefers labels from randomized control groups where they exist, since their error is sampling error. And it adds the support check of equation (12.37): the historical assignment of actions must cover the state-action combinations the optimizer will ask about, because a model trained on setpoint changes of at most two degrees has no evidence about four.

12.7 What to retain

Definition 12.1 (Counterfactual flexibility forecasting)**.** Flexibility is $F = P^{\text{base}} - P^{\text{ctrl}}$, and once the fleet is dispatched the baseline term is never observed, so baseline estimation is part of the response problem rather than a preliminary to it. The general object to forecast is the distribution of the controlled trajectory given the state and the planned control sequence,

$$p(F_{t+1:t+H} \mid X_t,\ u_{t+1:t+H}), \tag{12.41}$$

and because control changes the physical state, today's use of flexibility changes the next hour's availability. Randomized holdouts and controlled experiments are the only sources of credible counterfactual information, because their error is sampling error rather than model error.

Table 12.4 collects the chapter's objects with the subsection that defines each.

12.8 Exercises

Exercise 12.1. An event is settled with a metered controlled load of 75 MW and a baseline estimate of 115 MW, as in Example 12.1. (a) State the recorded flexibility and the true flexibility if the unobserved baseline was 105 MW, and express the label error as a fraction of the truth. (b) Suppose the baseline algorithm has error $\varepsilon_0 \sim \mathcal{N}(0, 6^2)$ in megawatts, independent across events. Using equation (12.12), state the distribution of the recorded label around the truth and the number of events needed for the mean recorded flexibility to be within 1 MW of the mean truth with 95 % confidence. (c) Now let the error instead be $\varepsilon_0 = -0.8\,(T - 32)$ megawatts for $T > 32$°C and zero otherwise. Explain why no number of events fixes the problem, and say which coefficient of a response model fitted on temperature is corrupted and in which direction.

Exercise 12.2. Use the response surface of Table 12.2 and the fitted curve $\mathbb{E}[F] = 46(1 - e^{-\Delta T_{\text{set}}/2})$ MW of Figure 12.6. (a) Compute the predicted reduction at $\Delta T_{\text{set}} = 4$°C and 6°C and the marginal gain per degree over each of those intervals. (b) A requirement of 35 MW must be met. Compare dispatching the whole fleet at 3°C with dispatching it at 1°C and, if that is insufficient, explain what additional information about the fleet would be needed to decide whether a shallow action on a larger population can substitute. (c) Taking $\Pr(\text{override}) = 0.05\,\Delta T_{\text{set}}$ and assuming an overriding device contributes nothing, recompute the expected reduction at each of the four tabulated setpoint changes and state where the maximum now lies.

Exercise 12.3. A fleet of 100 000 eligible devices is dispatched with a 10 % holdout, as in Figure 12.5. The dispatched arm meters a mean of 0.9 kW per device and the holdout arm 1.5 kW per device, with a per-device standard deviation of 1.2 kW in each arm. (a) Compute

Table 12.4: The objects of counterfactual flexibility forecasting.

Object	Definition	Section
$Y(0)$, $Y(1)$	potential outcomes under no action and under the action	12.1.2
$\tau = Y(0) - Y(1)$	treatment effect; flexibility with this sign convention	12.1.2
P^{base}, $P^{\text{ctrl}}(\cdot \mid u)$	baseline and controlled load	12.1.4
$F(t, u)$	flexibility as a function of time *and* action	12.1.4
approach A, approach B	subtract two forecast worlds; forecast the difference	12.2.1, 12.2.2
$\varepsilon_{0,i}$	baseline error, which enters the data as label error	12.2.3
label noise, label bias	baseline error unrelated to, or related to, conditions	12.2.3
similar-day, regression baseline	the two standard estimators of $Y(0)$	12.2.5
pre-event adjustment	modeled baseline shifted by the observed recent offset	12.2.6
$u =$ $[\Delta T_{\text{set}}, \text{start}, \text{duration}, \text{cohort}]$	the action vector; dose-response	12.3.1
$x_{t+1} = g(x_t, u_t, w_t)$	the state-space form; system identification	12.3.3
rebound	post-event consumption above baseline	12.3.4
E_{shed}, E_{rebound}	curtailed and recovered energy	12.3.5
$H = T_{\max} - T_{\text{in}}$	thermal headroom; the state that limits the service	12.3.7
CATE(X)	conditional average treatment effect	12.4.1
confounder	a driver of both the action and the outcome	12.4.2
holdout, randomized control	measured rather than modeled counterfactual	12.4.4
dual control	dispatching so as to deliver and to learn at once	12.4.5
saturation	falling marginal reduction per additional degree	12.4.6
$\Pr(\text{override} \mid u, X)$	behavioral limit on the action	12.4.7
$\text{Var}(F)$ with covariance	uncertainty of a difference of correlated terms	12.5.2
$p(Y \mid \text{do}(u))$	interventional distribution, distinct from $p(Y \mid u)$	12.5.3
$p(u \mid X) > 0$	support, overlap, positivity	12.5.5
the four levels	baseline, response, flexibility, decision	12.6.6

$\hat{\tau}$ from equation (12.38), the implied fleet flexibility delivered by the dispatched arm, and the standard error of $\hat{\tau}$. (b) The utility required 75 MW. State the capacity actually delivered and the shortfall caused by the holdout. (c) Find the smallest holdout that keeps the standard error of $\hat{\tau}$ below 0.02 kW, and state the capacity that holdout costs. Comment on the exploration-and-exploitation trade-off of Section 12.4.4 using your two answers.

Exercise 12.4. Take the worked event of Example 12.8: $\hat{F}_{0.5}(17{:}00) = 42$ MW, $\hat{F}_{0.1}(17{:}00) = 34$ MW and $\hat{F}_{0.5}(18{:}00) = 30$ MW under a two-hour, two-degree action. (a) Assuming the first-hour distribution is Gaussian, infer its standard deviation from the median and the tenth percentile, and state the firm commitment that could be met with 95 % probability. (b) Using the energies of equation (12.21) and the illustrative rebound of Example 12.4, state whether the event primarily sheds or shifts energy, and what additional quantity would have to be forecast to decide. (c) Suppose instead the fleet is dispatched for one hour only. Using the headroom trajectory of Figure 12.3, argue qualitatively whether the flexibility available at 19:00 would exceed, equal or fall short of the 30 MW the two-hour plan expects at 18:00, and name the state variable your argument turns on.

Chapter 13

Forecasting under distribution shift: when tomorrow is not drawn from the past

13.1 The changed data-generating process

Every chapter so far has rested on one assumption, usually without saying so. A model is fitted on historical data, validated on held-out historical data as in Chapter 9, and the validation error is then quoted as an expectation about the future. That inference is sound only if the future is statistically like the past. Real energy systems do not oblige. Tomorrow may be hotter than any day in the training record; the fleet acquires heat pumps and loses conventional air conditioners; a firmware release changes how a thermostat cycles; customers learn what a dispatch event feels like and start overriding it; and the optimizer of Chapter 14 begins proposing control actions the response model has barely seen. The question of this chapter is

What happens to a forecast when the data-generating process changes?

Jargon: distribution shift, covariate shift, concept drift, extrapolation, out-of-distribution, drift, novelty, calibration

Distribution shift is any difference between the joint distribution of inputs and targets that a model was trained on and the one it operates under. *Covariate shift* is the special case in which the distribution of the inputs moves while the input-to-target relationship holds. *Concept drift* is the opposite case: the inputs look as before but the relationship itself has changed. *Extrapolation* is prediction at an input outside the region the training data occupied, as opposed to *interpolation* between observed inputs. An input far from anything in the training set is called *out-of-distribution*, abbreviated OOD. *Drift* is slow, continuous change, such as an override rate creeping upward over months; *novelty* is an abrupt arrival of something unprecedented, such as a firmware release or a record temperature. *Calibration*, introduced in Chapter 10, is the agreement between the stated probability of a forecast interval and the frequency with which outcomes actually fall inside it; under shift it is the first property to fail.

Key idea

A validation score is a statement about one distribution. It transfers to operation only to the extent that operation draws from the same distribution. Outside the region the training data covered, the honest output of a forecasting system is not a sharper number but a wider one.

This section states shift in symbols, gives the example that motivates the whole chapter, and separates the three kinds of shift that have to be diagnosed differently.

13.1.1 Training and operating distributions

Write the training data as draws from a joint distribution over inputs and targets, and operation as draws from another:

$$(X, Y) \sim p_{\text{train}} \qquad \text{during fitting,} \qquad (X, Y) \sim p_{\text{future}} \qquad \text{during operation.} \tag{13.1}$$

If $p_{\text{future}} = p_{\text{train}}$, the ordinary machinery applies: a rolling-origin validation estimates the error the model will make, and a calibrated interval stays calibrated. The model operates *under distribution shift* when $p_{\text{future}} \neq p_{\text{train}}$.

Two things should be said immediately. Shift is not automatic failure. A model can be exposed to a changed input distribution and still predict well, because the structure it learned may be the structure that still holds. What shift does destroy is the *warrant* for the performance claim: the held-out error was computed under p_{train}, and once p_{future} differs, that number is no longer an estimate of anything the system is about to do. The forecast may still be good; the evidence that it is good has expired. Recognizing which of the two situations one is in is the practical problem this chapter addresses.

13.1.2 The example that motivates the chapter

Take a flexibility model for a thermostat fleet fitted on three years of history. Sorting that history by outdoor temperature, almost all of it lies in the band

$$10^{\circ}\text{C} \leq T \leq 35^{\circ}\text{C}, \tag{13.2}$$

with a handful of hours above 35°C and none above 38°C. Tomorrow's weather forecast is 42°C. The model will return a number, and the number will carry whatever interval the training data implied, but the training data contain almost no direct evidence about how this fleet behaves at 42°C.

The uncomfortable part is the coincidence of two facts. Figure 13.1 shows the first: a histogram of training hours by outdoor temperature. The horizontal axis is outdoor temperature in degrees Celsius, the vertical axis is the number of training hours in each 2°C bin, the shaded region is the band of equation (13.2) in which essentially all of the data lie, and the red vertical line at 42°C is tomorrow. The bars fall away steeply above 33°C and reach zero before 39°C, so the red line stands in a region where the bar height is exactly zero. The second fact is operational: 42°C is precisely when the grid needs the fleet most, because that is when the system peak occurs and when a capacity commitment is settled against delivery. The conditions of highest operational importance are the conditions of least statistical familiarity, and that is a structural feature of energy forecasting rather than an accident of this data set.

13.1.3 Three types of shift

The joint distribution factors in two ways, $p(X, Y) = p(X)\,p(Y \mid X) = p(Y)\,p(X \mid Y)$, and each factorization names a kind of shift:

$$\text{covariate shift:} \quad p_{\text{future}}(X) \neq p_{\text{train}}(X), \quad p(Y \mid X) \text{ approximately unchanged;} \tag{13.3}$$

$$\text{concept drift:} \quad p_{\text{future}}(Y \mid X) \neq p_{\text{train}}(Y \mid X); \tag{13.4}$$

$$\text{label (prior) shift:} \quad p_{\text{future}}(Y) \neq p_{\text{train}}(Y), \quad p(X \mid Y) \text{ unchanged.} \tag{13.5}$$

The distinction is not academic: the three call for different responses. Covariate shift leaves the learned relationship intact, so the remedy is to know where the relationship has support and

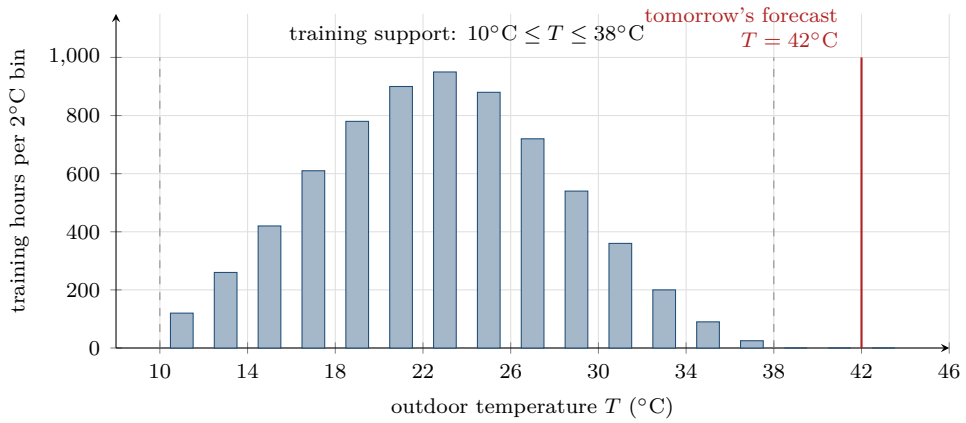


Figure 13.1: Distribution of training hours by outdoor temperature for a three-year thermostat-fleet record, with tomorrow's forecast marked. Bars are counts per 2°C bin; the dashed verticals bound the training support; the red line is the operating point. Counts are illustrative and follow the shape of a temperate-climate record.

to widen the uncertainty outside it. Concept drift invalidates the relationship, so no amount of reweighting the old data helps and new data are required. Label shift matters most in classification problems, where the class frequencies move; for the continuous targets of this book it appears mainly as a change in the marginal distribution of the quantity being forecast, and it is the least important of the three here.

Figure 13.2 sketches the three. Each column is one type. The upper row shows a distribution of the quantity that moves, with the training distribution as a solid blue curve and the operating distribution as a dashed red curve; the lower row shows the conditional relationship, again solid for training and dashed for operation. Column (a) has two separated densities on top and a single line below: the inputs have moved, the relationship has not. Column (b) has one density on top and two lines of different slope below: the same inputs now produce different responses. Column (c) is drawn in the reverse direction, because that is how label shift is specified: the upper panel is the marginal distribution of the target Y, which moves, and the lower panel is the input distribution conditional on the target, $p(X \mid Y)$, which does not. Reading the figure column by column, the diagnostic question is which of the two panels in a column has two curves in it.

13.1.4 Covariate shift: the same physics on new inputs

Suppose the true relationship between outdoor temperature and fleet cooling load is

$$P(T) = 50 + 2\,(T - 22)_+ \,\text{MW}, \qquad (z)_+ = \max(z, 0), \tag{13.6}$$

a base load of 50 MW plus a weather-driven term that switches on at 22°C. The training record covers 20°C to 32°C; operation takes place between 35°C and 40°C. The distribution of the input has moved a long way, so $p(T)$ has certainly shifted. The physics in equation (13.6), however, is unchanged: the same linear dependence above the same breakpoint continues to hold. A model that recovered the functional form will extrapolate correctly, and $p(P \mid T)$ remains valid.

This is the benign case, and it is benign for a reason worth naming: the model's errors at the new inputs are determined by how well it captured structure, not by how much data it saw nearby. The practical difficulty is that the modeler cannot tell from the data alone whether the structure extends. Equation (13.6) is linear for all T by assumption; a real fleet's load flattens when compressors stop cycling and start running continuously. Covariate shift is therefore safe

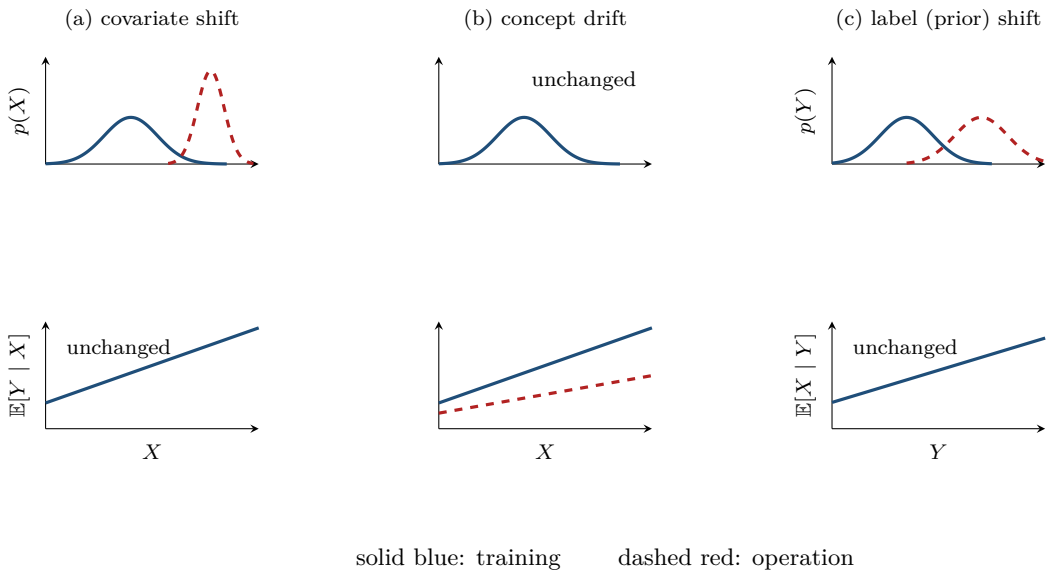


Figure 13.2: The three shifts of equations (13.3)–(13.5). Upper row: the distribution that moves. Lower row: the conditional relationship. Columns (a) and (b) condition the target on the input; column (c) is written in the generative direction, with the target's marginal on top and $p(X \mid Y)$ below. All curves are illustrative sketches: the densities are Gaussian and the conditional means linear.

exactly to the degree that the model class encodes the mechanism rather than the observed curve, which is the subject of Section 13.2.5.

13.1.5 Concept drift: the same inputs with new behavior

Now consider a firmware update pushed to the thermostats. Before the update the fleet's flexibility response to weather, state and a dispatch action u was some function

$$F = f_{\text{old}}(X, u), \qquad \text{and after it} \qquad F = f_{\text{new}}(X, u), \tag{13.7}$$

because the new firmware changes the compressor cycling logic: minimum run times, staging, how a setpoint offset is ramped. Under identical weather, an identical fleet state and an identical command the delivered response is now different. In the notation of equation (13.4), $p(F \mid X, u)$ has changed.

Concept drift is harder than covariate shift in a specific sense. Under covariate shift the historical data remain valid evidence about the relationship; they are simply concentrated in the wrong place, and techniques that reweight them can help. Under concept drift the historical data are evidence about a relationship that no longer exists, so reweighting them cannot help and sufficient new data under the new regime are the only cure. The detection problem is also harder: a covariate shift is visible in the inputs alone, before any outcome is observed, whereas concept drift is invisible until the outcomes arrive and the residuals of Section 13.4.1 deteriorate.

13.2 Sources of shift in a fleet

Shift in an aggregator's forecasting problem is not an unusual event imported from outside. It is generated continuously by the business itself: by enrollment, by customer learning, by the control policy the aggregator chooses, and by the simple fact that weather occasionally leaves the historical envelope. This section catalogues the sources and then examines what different model classes do when an input arrives outside the region they were fitted on.

13.2.1 Fleet composition shift

A fleet that was 70 % conventional air conditioners and 30 % heat pumps becomes, over two years of enrollment growth, 40 % conventional and 60 % heat pumps. Suppose every individual device of each type behaves exactly as it always did, so that no device-level relationship has drifted at all. The aggregate has nonetheless changed, because the aggregate is a mixture and the mixture weights have moved:

$$p_t(X) = \pi_t\, p_{\mathrm{AC}}(X) + (1 - \pi_t)\, p_{\mathrm{HP}}(X), \qquad \pi_t : 0.70 \text{ falling to } 0.40, \tag{13.8}$$

and a heat pump's temperature response, its cycling pattern and its shoulder-season behavior differ from an air conditioner's. Fleet-level features such as mean duty cycle and mean load per device shift for reasons entirely internal to the aggregator's growth.

This is the concrete payoff of the explicit cohort attributes urged in Chapter 11. If the equipment mix is a model input, a change in π_t is a change in a feature value that the model already knows how to use, and the forecast adjusts without retraining. If the mix is not an input, the same change appears as unexplained nonstationarity: the aggregate relationship seems to drift, and the drift is attributed to behavior or weather when its actual cause is the enrollment pipeline. Making composition an input converts a shift problem into an ordinary conditioning problem.

13.2.2 Behavioral drift

Customers are not a fixed physical system. They learn what a dispatch event feels like, and their responses change in consequence. Four mechanisms recur. First, familiarity: after a few events, a household knows what two degrees of setpoint offset feels like in August and decides whether to tolerate it, so the override rate at a given event depth moves. Second, override behavior proper: customers discover the app, the schedule and the thermostat's manual controls, and use them more competently. Third, occupancy: work-from-home patterns change who is in the building during an afternoon event, which changes both the baseline load and the probability that an offset is noticed. Fourth, attrition: after repeated events, some customers opt out, and the ones who opt out are not a random sample, so the remaining population is systematically more tolerant.

All four change $p(Y \mid X)$ rather than $p(X)$: the weather, the state and the command can be identical while the response is not. This is concept drift whose cause is human, and it is the most common form of drift in a demand-response program. It is also usually gradual, which makes it a drift rather than a novelty in the sense of Section 13.5.7, and therefore something a slowly adapting behavioral component can track.

13.2.3 Control-policy shift

Historical dispatch at the aggregator used setpoint offsets of $\Delta T = 1$ to 2°C, because that is what the program rules and the operators' habits produced. A new optimizer, built against the response model, proposes 3°C. The response model has almost no data at 3°C, so the distribution of the action has shifted: $p(u)$ is not what it was.

The point to take from this is that for a response model the relevant distribution is not $p(X)$ but the joint

$$p(X, u), \tag{13.9}$$

over states and actions together. A forecasting system embedded in a control loop therefore has two shift surfaces rather than one: the state space can move, because the weather and the fleet move, and the action space can move, because the controller changed. The second is self-inflicted, which also means it is controllable, and Section 13.5.3 returns to it as a constraint on the optimizer rather than as a property of the weather.

13.2.4 Interpolation and extrapolation

The distinction that organizes the rest of this section is elementary and is routinely lost in practice. Suppose the training record contains observations at outdoor temperatures

$$T \in \{25, 27, 30, 32, 35\}^{\circ}\mathrm{C}. \tag{13.10}$$

Predicting at 31°C is *interpolation*: the input lies between observed inputs, data bracket it on both sides, and a smooth model fitted to the surrounding points is constrained by them. Predicting at 42°C is *extrapolation*: there is nothing on one side, and whatever the model returns is determined by the model's functional form rather than by evidence.

Most machine-learning methods are far more trustworthy at interpolation than at extrapolation, and the reason is that they are fitted by minimizing an empirical loss, which places no constraint at all on regions with no observations. The behavior of a fitted model outside its training support is an artifact of its inductive bias. That artifact differs sharply between model classes, which is why the choice of model class becomes a risk decision when extrapolation is expected.

13.2.5 Extrapolation behavior of trees and linear models

A decision tree partitions the input space into boxes and predicts a constant in each. Suppose the fitted tree contains the rule $T > 35^{\circ}\mathrm{C} \Rightarrow F = 45\,\mathrm{MW}$, learned from the handful of hot hours in the record. An input of 42°C satisfies that rule and the tree returns 45 MW. It would return 45 MW at 50°C and at 100°C. Nothing in the structure of a tree expresses the fact that 42 is beyond everything it has seen; the input simply falls into the hottest leaf, and the leaf's value was estimated from a small number of observations at much lower temperatures. The failure mode is silent: the prediction is a plausible number, produced with no indication that it is a constant extension of the last observed box. Gradient boosting of Chapter 5 inherits this behavior, because a sum of trees is still piecewise constant.

Linear models extrapolate, which is a different danger. A linear fit over the same data might give

$$F(T) = 2T - 20\,\mathrm{MW}, \tag{13.11}$$

so $F(35) = 50\,\mathrm{MW}$ and $F(45) = 70\,\mathrm{MW}$. The model does respond to the new input, which is an improvement on a constant, and the response may well be in the right direction. But the real fleet saturates: above some temperature nearly every compressor is already running continuously, there is no cycling left to curtail, and the achievable response flattens near a physical ceiling of about 55 MW. Equation (13.11) projects straight through that ceiling and predicts 70 MW of flexibility the fleet does not possess. The lesson is that extrapolating is not the same as extrapolating correctly. A model that extends a trend will be right only if the trend is the mechanism, and a physics-informed functional form, such as one that approaches a capacity limit, helps precisely when the mechanism is known in advance.

Figure 13.3 draws the three behaviors on one axis. The horizontal axis is outdoor temperature from 25°C to 45°C, the vertical axis is fleet flexibility in megawatts, and the shaded region on the left is the training support, which ends at 35°C. The green step function is the tree, flat within each leaf and permanently flat at 45 MW beyond the last split. The blue straight line is equation (13.11). The gold curve is the same linear term capped at the physical ceiling, $F(T) = \min(2T - 20,\ 55)\,\mathrm{MW}$. Inside the support all three agree closely, which is why a validation score computed on historical data cannot distinguish them. At the red vertical line marking 42°C they return 45, 64 and 55 MW respectively: a spread of 19 MW produced entirely by inductive bias and not at all by data.

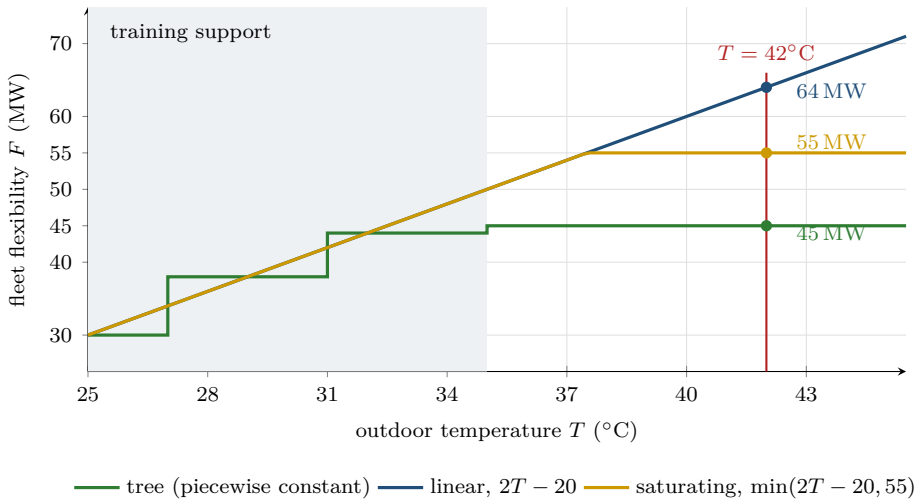


Figure 13.3: Three model classes outside their training support. Curves are, in order, the step function with leaf values 30, 38, 44 and 45 MW at splits 27, 31 and 35°C; the line $F = 2T - 20$ of equation (13.11); and its saturating version $F = \min(2T - 20,\ 55)$. The shaded region is $T \leq 35°\text{C}$, where all three agree. Values are illustrative.

13.2.6 Physical constraints as a bound on extrapolation

The saturating curve in Figure 13.3 was not obtained from data. It was obtained from two facts about the system that hold at every temperature, inside or outside the training support: a thermostat fleet cannot shed more cooling load than it is drawing, and curtailment cannot be negative. Writing $P_{\text{HVAC},t}$ for the fleet's cooling draw at time t,

$$0 \leq F_t \leq P_{\text{HVAC},t}, \tag{13.12}$$

which is a statement about conservation and equipment ratings rather than about any fitted parameter.

A generic regression has no access to equation (13.12) and will violate it cheerfully: a linear model predicts negative flexibility at low temperatures and impossible flexibility at high ones, and a neural network trained without constraints does the same in a less predictable pattern. Imposing the bound, either by construction (predicting a shedable fraction in $[0, 1]$ and multiplying by the forecast draw) or by post-processing (clipping), limits how absurd an extrapolation can get. This is the general principle behind physics-informed modeling as a defense against shift: structure that is true everywhere constrains the model where data are absent, and the more of the model's output range that is pinned by physics, the less of it is left to inductive bias.

> **Principle 13.1.** *Outside the training support, a model's prediction is a property of its functional form. The only reliable way to control it is to build in structure that holds where data do not.*

13.3 Recognizing unfamiliar inputs

A forecasting system that is going to behave differently under shift must first notice that it is under shift. This section builds the detection side, from the simplest range check to the distance measures that catch combinations no single variable flags, and then shows what goes wrong with the probabilistic output of Chapter 10 when the detection is missing.

13.3.1 Out-of-distribution detection and range checks

The usual output of a forecasting service is $\hat{y}_{t+h}$ and perhaps an interval. A system operating under possible shift needs a second, separate output: an answer to the question of whether the input X^* it was just handed looks like the inputs it was trained on. Producing that answer is *out-of-distribution detection*, and it is a property of the input alone, available before any outcome is observed and therefore before any error can be measured.

The cheapest form is a per-variable range check. If the training temperatures spanned $[10, 38]$°C and the current forecast is 42°C, the system raises a flag that reads "outside training support" rather than silently predicting. The same check applies to every input that matters: available device count N_{avail} outside its historical band, fleet duty cycle D above anything observed, a setpoint offset deeper than any in the record, a telemetry completeness below any previous value. Range checks are underrated because they are trivial. They are also the only diagnostic that is certain to be correct, needs no model, costs nothing to compute, and is immediately interpretable in a post-mortem; a system with range checks and nothing else is far better protected than a sophisticated system with none.

13.3.2 Multivariate shift and joint support

Range checks have one structural blind spot. The training record contains hours at $T = 38$°C, and it contains hours at relative humidity RH $= 90\,\%$. It may contain no hour at which both occurred, because in this climate the hottest days are dry and the humid days are mild. Today is $T = 38$°C with RH $= 90\,\%$. Every marginal check passes and the operating point is nevertheless unprecedented.

The combination matters physically, not just statistically. Humid heat loads the latent side of a cooling coil, so a unit that would cycle at 38°C and 40 % humidity runs continuously at 38°C and 90 %; occupant discomfort at a given indoor temperature is higher, so overrides are more likely; and the fleet's thermal recovery after an event is slower. The relationship the model needs is one it never observed.

Figure 13.4 shows the geometry. The horizontal axis is outdoor temperature, the vertical axis is relative humidity, and each blue dot is a training hour; the dashed outline encloses the region the training hours occupy. The horizontal gray band holds the record's humid hours, those above 86 %, and the vertical gray band holds its hot hours, those above 36.5°C; the red cross is today's operating point. The cross lies inside both bands, so each marginal range check passes, and outside the dashed region, so the combination has no precedent. The cloud slopes downward, because hot hours in this climate are dry, and the upper-right corner where the two bands cross is empty. Joint support, not marginal support, is what a model has evidence about.

13.3.3 Distance to the training set

The figure suggests the quantitative version. Define the distance from a candidate input to the nearest training input,

$$d(X^*) = \min_{i=1,\dots,n} \|X^* - X_i\|, \tag{13.13}$$

over the n rows of the training set. A small $d(X^*)$ means that something very like this situation has been seen and that the model's prediction is supported by nearby evidence; a large $d(X^*)$ means it is not.

Equation (13.13) needs two refinements before it is usable. The first is scale. Temperature in degrees Celsius, humidity in percent and device counts in tens of thousands cannot be compared in a raw Euclidean norm, which would be dominated entirely by N_{avail}; each coordinate must be standardized by its training standard deviation. The second is correlation, and it is the more interesting of the two.

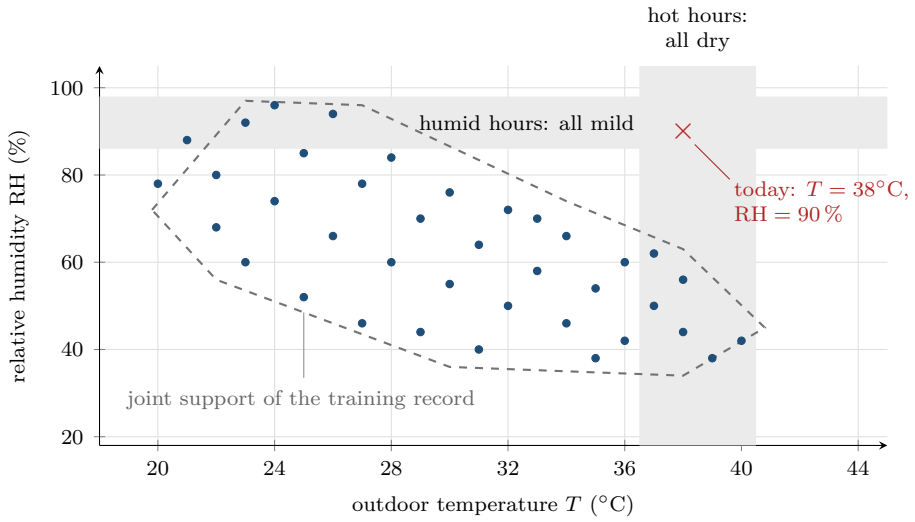


Figure 13.4: Joint support of two inputs. Dots are training hours; the dashed polygon is the region they occupy; the horizontal band is RH $\geq 86\,\%$ and the vertical band is $T \geq 36.5°C$, both of which the record visits. The operating point passes both marginal range checks and lies outside the joint support. Points are illustrative of a climate in which hot hours are dry.

Mahalanobis distance. Standardizing coordinates treats them as independent, which they are not. Write μ and Σ for the mean vector and covariance matrix of the training inputs and define

$$D_M^2(X) = (X - \mu)^\mathsf{T} \, \Sigma^{-1} \, (X - \mu). \tag{13.14}$$

This measures distance in units of the training distribution's own shape. Its effect is exactly what the humidity example requires. High outdoor temperature and high fleet duty cycle normally occur together, so an hour with both is a long way from the mean in a standardized Euclidean sense and a short way in the Mahalanobis sense: the combination is ordinary. High temperature with high humidity, in a climate where the two are negatively correlated, is the reverse. A single number, D_M, therefore distinguishes a familiar extreme from an unfamiliar combination, which no collection of marginal range checks can do.

13.3.4 Uncertainty should grow outside the support

Detection is only useful if the forecast responds to it. The behavior one wants is simple to state. At a familiar operating point, say $T = 30°C$, the model reports

$$F = 40 \pm 3\,\text{MW}, \tag{13.15}$$

a narrow band earned by a great deal of nearby data. At an unfamiliar point, $T = 43°C$, the ideal report is something like $F = 45 \pm 15\,\text{MW}$: a central estimate that still uses whatever structure the model has, and an interval that admits how little evidence stands behind it.

Many uncertainty methods in common use do not behave this way. A quantile regression fitted as in Chapter 10 produces conditional quantiles whose spread is itself extrapolated from the training region, so at 43°C it reports the spread that applied at 35°C. The empirical residual quantiles of a boosted ensemble reflect the residuals that were observed, and no residuals were observed at 43°C. A neural network's softmax or its ensemble variance can be small simply because every member extrapolates the same way. The resulting failure is the dangerous one: the model is *confidently wrong*, reporting a narrow interval centered on a number for which it has no evidence. An explicit out-of-distribution signal is what allows the uncertainty layer to

be widened deliberately, because the model's own uncertainty estimate cannot be relied on to widen itself.

13.3.5 Calibration failure under shift

The quantitative version of that failure is a loss of calibration. A forecaster whose nominal 90 % interval covered the outcome in 90 % of historical cases is calibrated on the historical distribution, and that is the only thing the calibration statement asserts. After a shift, the same intervals may cover only 60 % of outcomes. Four outcomes in ten now fall outside a band advertised to miss one in ten: not a slight miscalibration but a severe overconfidence, and one that propagates directly into every capacity commitment the intervals support.

Two consequences follow. Calibration is not a property a model acquires once; it is a property of a model relative to a distribution, so it has to be monitored continuously rather than certified at deployment. And because coverage is a frequency, measuring it requires accumulating outcomes, which makes it a lagging indicator. The input-side diagnostics of Sections 13.3.1 to 13.3.3 are what provide the leading signal.

13.3.6 Conditional calibration by regime

Aggregate coverage can hide the failure completely. Consider a forecaster whose 90 % intervals covered the outcome in 89 % of all hours of the last quarter. That is close enough to nominal to pass any global check. Split those hours by regime: under normal conditions coverage is 94 %, and on extreme-heat hours it is 61 %. The aggregate is dominated by the normal hours because there are vastly more of them, and the slight overcoverage there offsets the severe undercoverage in the regime where the fleet is dispatched and where a missed commitment is penalized.

Figure 13.5 draws the three numbers. The horizontal axis has one bar per evaluation slice, the vertical axis is the observed coverage of a nominal 90 % interval in percent, and the dashed red line is the nominal level. The first bar, at 89 %, is the aggregate and sits essentially on the line. The second, 94 %, is normal conditions and sits slightly above it. The third, 61 %, is extreme heat and falls far below. The figure's content is the contrast between the first and third bars: the same forecaster, the same intervals, and a conclusion that reverses depending on whether the evaluation conditions on the regime. Conditional calibration must therefore be evaluated and monitored by operating regime, with the regimes chosen to isolate the conditions that matter operationally rather than to balance the sample sizes.

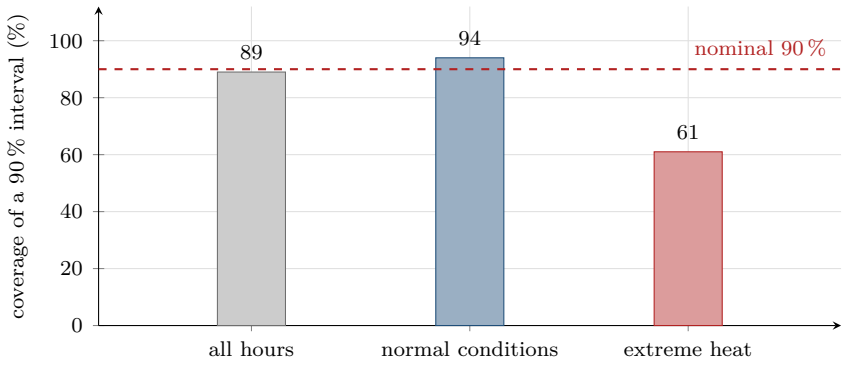


Figure 13.5: Observed coverage of a nominal 90 % prediction interval, aggregated and split by operating regime. The aggregate figure passes a global calibration check while the extreme-heat slice is severely overconfident. Figures are illustrative.

13.4 Monitoring and diagnosis

Detection on the input side and calibration checks on the output side are instruments. This section assembles them into a monitoring practice: what to watch, how to tell a real change in the world from a broken pipeline, and what to do once something has changed. The organizing claim is that the response to a detected change is a diagnosis, not a retrain.

13.4.1 Residual drift and rolling performance

The most direct indicator that something has changed is the behavior of the forecast errors. With $e_t = y_t - \hat{y}_t$ as in Chapter 9, a healthy forecaster has errors whose first two moments are stable in time, so the quantities to track over a rolling window are

$$\mathbb{E}[e_t], \qquad \mathrm{Var}(e_t), \qquad \mathrm{MAE}_t. \tag{13.16}$$

A mean error that moves from 0.5 MW to 8 MW is not noise. A bias of 8 MW sustained over a window means the model is systematically wrong in one direction, and systematic one-directional error is the signature of a changed relationship rather than of ordinary variability.

Table 13.1 records six weeks of rolling mean absolute error for such a forecaster, and Figure 13.6 plots the same six numbers. The table's columns are the week, its mean absolute error in megawatts, and the ratio to the mean of the first four weeks; the figure's horizontal axis is the week index, its vertical axis is mean absolute error in megawatts, and the dashed line is the four-week reference level of 4.3 MW. Weeks one to four sit at 4.2, 4.0, 4.5 and 4.4 MW, a band of ordinary week-to-week variation. Weeks five and six are 7.8 and 8.1 MW. The jump is a factor of nearly two, it persists for a second week rather than reverting, and it is far outside the spread of the first four weeks. Something changed at week five.

Table 13.1: Rolling weekly mean absolute error of a fleet flexibility forecaster. The reference level is the mean of weeks one to four, 4.3 MW. Figures are illustrative.

Week	MAE (MW)	Ratio to reference	Reading
1	4.2	0.98	within normal variation
2	4.0	0.93	within normal variation
3	4.5	1.05	within normal variation
4	4.4	1.02	within normal variation
5	7.8	1.81	step change; triggers diagnosis
6	8.1	1.88	persists, so not a single bad week

What the monitor must not do is conclude what changed. Table 13.2 lists five candidate explanations for the step in Figure 13.6 together with the evidence that discriminates between them. The columns are the candidate cause, the observation that would confirm it, and the appropriate response. The five responses are all different, and two of them are the opposite of retraining. That is the argument of Section 13.4.3 in tabular form.

13.4.2 Monitoring the inputs

Error-based monitoring has one unavoidable defect: it requires outcomes. A forecast issued today for tomorrow evening cannot be scored until tomorrow evening has happened and the metering has settled, which in a demand-response program can be days. Input-based monitoring has no such lag, because the inputs are known at issue time.

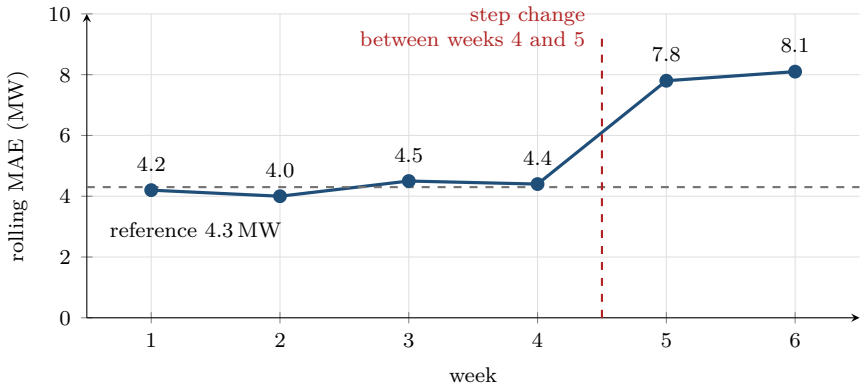


Figure 13.6: The six weekly mean absolute errors of Table 13.1. The dashed horizontal line is the four-week reference level; the dashed vertical line marks where the step occurs.

Table 13.2: Candidate causes of the step change in Figure 13.6 and the evidence that distinguishes them.

Candidate cause	Confirming evidence	Response
Data problem	a feed changed units, a sensor went stale, missingness jumped at the same timestamp	fix the pipeline; do not retrain on corrupted rows
Weather regime	the week's temperatures lie outside the training support, and the error concentrates in the hot hours	widen intervals; collect the new regime's data
Firmware release	the vendor's release date coincides; the error is confined to the updated software version	refit the behavioral component on post-release data
Composition change	cohort fractions moved; error is concentrated in the growing cohort	add or update the composition features
Equipment degradation	the error grows slowly rather than stepping, and tracks device age	re-estimate device capacity parameters

Feature drift. Track the distribution of each model input, not just its presence. The available device count N_{avail} normally runs between 70 000 and 90 000; today it is 45 000. That is a major change in the distribution of a feature that Chapter 11 showed sets the scale of the entire fleet forecast, and it is known before a single outcome arrives. The same applies to the temperature distribution, the duty-cycle distribution, the cohort fractions and the distribution of dispatch depths. A feature monitor is the leading indicator and an error monitor is the lagging one; a system needs both, and the leading one is what buys time to act.

Data-quality shift versus real-world shift. A distribution monitor cannot, by itself, say what it has found. Suppose the fleet's mean indoor temperature $\bar{T}_{\text{in}}$ jumps abruptly by several degrees. Two explanations fit the data equally well: homes really did get hotter, because a heat event began or because a large cohort's setpoints were raised; or an ingestion pipeline changed, and a field arriving in degrees Fahrenheit is being read as degrees Celsius. The monitor sees the same histogram in both cases. Distinguishing them requires evidence the monitor does not contain: the physical plausibility of the magnitude, whether the change is simultaneous across unrelated cohorts, whether it coincides with a deployment, whether the affected field's reported units changed. Separating genuine system drift from pipeline failure is the first question to answer about any detected change, because the two have opposite remedies.

Missingness drift. Monitor the probability that a feature is missing, not only the values of the feature when present. Indoor temperature was available for 95 % of devices and is now available for 60 %. The observed values may look entirely normal, and the forecasting problem has nonetheless changed: the model is now operating on a different, self-selected subset of the fleet, imputation is carrying far more of the load than it was validated for, and the devices whose telemetry dropped out are quite possibly the ones whose behavior is unusual. Pr(feature missing) is a feature of the problem and belongs on the monitoring dashboard beside the feature itself.

13.4.3 Detect, diagnose, respond

The tempting automation is to connect a drift detector directly to a retraining job. It is the wrong pipeline, and Table 13.2 shows why in two of its five rows. Suppose the detected change is caused by a broken temperature feed that reports a constant value. Retraining on that period teaches the model that temperature does not predict load, which is worse than the drift it was meant to fix and survives long after the feed is repaired. Suppose instead the cause is a firmware bug that the vendor will roll back next week. Retraining institutionalizes a transient: the model learns the buggy behavior, and when the rollback lands, the model is wrong again, in the opposite direction.

The correct pipeline has three stages rather than two,

$$\text{detect} \longrightarrow \text{diagnose} \longrightarrow \text{respond},$$

and the response depends entirely on the diagnosis. Monitoring should raise a case for a human to triage, with the evidence attached, and the automation should sit on the detection and evidence-gathering steps, not on the remedy.

13.4.4 Adapting the fit: windows and weights

When the diagnosis is genuine, gradual change, the question becomes which data to fit on. The two standard answers are the ones already met in the validation design of Chapter 9. An *expanding window* uses all history, say 2019 to 2026: more data, better estimated parameters, and a model that averages over behavior that no longer exists. A *rolling window* uses only

the recent past, say 2025 to 2026: fewer data, noisier parameters, and faster adaptation. The trade-off is between variance and staleness, and its resolution depends on one quantity the modeler must estimate, the speed at which the system changes relative to the rate at which informative data arrive.

A continuous version avoids the discrete choice. Weight each training observation by its age,

$$w_i = \exp(-\lambda\, \Delta t_i), \qquad \Delta t_i = t_{\text{now}} - t_i, \tag{13.17}$$

and fit by weighted loss. Recent observations dominate, older ones still contribute rather than being discarded at a window boundary, and the single parameter λ sets the effective memory: the weight halves after $\ln 2/\lambda$, so a half-life of three months corresponds to $\lambda \approx 0.0077$ per day. Equation (13.17) gives gradual adaptation with no discontinuity when a window rolls, and λ can be chosen by the same rolling-origin evaluation that chooses any other hyperparameter.

13.4.5 Stable physics and adaptive behavior

Windows and weights apply one adaptation rate to the whole model, which is wasteful, because the parts of the model do not change at the same speed. Separate them:

$$P = f_{\text{physics}}(\cdot) + f_{\text{behavior}}(\cdot), \tag{13.18}$$

where the first term describes thermal and electrical mechanisms and the second describes what people and control software do. The thermal response of a building to outdoor temperature changes on the timescale of renovations; the probability that a customer overrides an event changes on the timescale of weeks. Fitting both on a three-month rolling window discards physical information that is still perfectly valid; fitting both on all history refuses to track the behavior.

The resulting design is a fixed physical structure with an adaptive correction on top. Concretely, the single-zone thermal model used throughout this book,

$$C\,\frac{\mathrm{d}T_{\text{in}}}{\mathrm{d}t} = \frac{T_{\text{out}} - T_{\text{in}}}{R} - Q_{\text{HVAC}}, \tag{13.19}$$

with thermal capacitance C and envelope resistance R, is stable: the same equation held last year and will hold next year, and its parameters move only when the building does. The override probability $\Pr(\text{override} \mid X, u)$ is not stable: it drifts as customers become familiar with the program, exactly as Section 13.2.2 described. Updating the behavioral component frequently while leaving equation (13.19) alone adapts where adaptation is needed and retains information where it is not. This is the most useful single structural response to drift available to a forecasting system.

> **Principle 13.2.** *Partition the model by the rate at which its parts change, and give each part its own adaptation rate. Refitting a stable mechanism on a short window throws away valid information; refitting drifting behavior on all history refuses to track it.*

13.4.6 Novel devices and transfer learning

Shift also arrives as a new kind of object. A manufacturer introduces an HVAC type the fleet has never contained, the enrollment pipeline records its equipment type as a value no training row carried, and the forecaster is asked for a prediction. A device-specific model is impossible: there is no history. A global model of Chapter 11 is in a better position, because what it learned has the form “devices with these characteristics, under these conditions, behave this way”, and

some of the new type's characteristics are measured: its rated power, its runtime pattern in the first days of telemetry, its observed response to outdoor temperature. The generalization is real but partial, and the right output is a prediction with visibly elevated uncertainty. This is the cold-start problem of Chapter 11 compounded by shift: not only is this device's history absent, the population it belongs to is absent too.

Transfer learning. The standard remedy is to start from the global model f_θ fitted on the whole fleet and fine-tune it on the small data set the new class has produced, rather than fitting the new class from scratch. The general structure comes from the fleet, the adjustment comes from the new cohort, and the amount of data required for the adjustment is far smaller than the amount required to learn the structure. In practice this means freezing or strongly regularizing most of the parameters and allowing a small number of class-specific ones to move, which is the same partial-pooling idea as in Chapter 11, applied across cohorts instead of across devices.

13.4.7 Geographic shift and the four kinds of shift

A model trained on a New York fleet and deployed on a Texas fleet faces every mechanism in this section at once. The climate differs, so the temperature distribution shifts and the operating range extends beyond the training support. The building stock differs in construction, insulation and size, so the thermal parameters of equation (13.19) differ. The equipment differs: more heat pumps, different sizing conventions, different refrigerant systems. Behavior differs, because setpoint habits and tolerance for warm afternoons are regional. And the utility programs differ, with different event rules, incentives and opt-out provisions. The consequence is that $p(Y \mid X)$ can differ even when the hardware is nominally identical, so geographic transfer is a new generalization problem and not a redeployment.

Table 13.3 separates the four kinds of shift a production forecasting system may be asked to handle. The columns are the kind, what moves, and the design question it raises. The table's purpose is to be answered explicitly at design time rather than discovered in operation: a system intended only to survive next summer in the same region needs very different defenses from one intended to be sold into a new territory.

Table 13.3: Four kinds of shift a production forecasting system may face.

Kind	What moves	Design question
Temporal	the same region in a future year: weather, behavior, firmware	how fast does the system drift, and at what cadence must the behavioral component be refitted?
Geographic	a different region: climate, building stock, programs	is anything transferable beyond functional form, and what must be re-estimated locally?
Device	different equipment classes entering the fleet	are equipment attributes model inputs, or is equipment type implicit in the fitted parameters?
Policy	a different control strategy: deeper offsets, new event shapes	does the response model have support over the action set the optimizer may propose?

13.5 Regimes, actions and the optimizer

The shifts catalogued so far were changes in the inputs or in the response. This section treats the two cases in which shift interacts with the physics and with the decision that consumes the forecast: a genuine change of operating regime under extreme conditions, and the tendency of an optimizer to seek out exactly the actions the model knows least about.

13.5.1 Extreme events and regime change

A heat wave is not a large value of one input. It changes several things at once: the weather leaves the historical envelope; the devices saturate, because at 42°C air conditioners approach continuous operation rather than cycling; behavior changes, because occupants are less tolerant and override more; and grid conditions change, because prices spike and the utility may call events on consecutive days, leaving the fleet without thermal recovery. The temperature-to-power relationship estimated at 30°C is a relationship between cycling equipment and weather. At 42°C the equipment is not cycling. The regime has genuinely changed, which is a different statement from "the input is large".

The quantitative version runs through the duty cycle. Let D be the fraction of the fleet's compressors running. In mild weather $D \approx 40\,\%$, in hot weather $D \approx 70\,\%$, and in extreme heat $D \approx 100\,\%$. While $D < 1$, raising the outdoor temperature raises consumption by raising the duty cycle, so $\partial P/\partial T$ is governed by the thermal model. Once $D = 1$, the equipment has no further capacity to take up, so $\partial P/\partial T$ collapses toward zero for consumption and the flexibility available for curtailment changes character: there is a large load running, but no cycling slack, and the comfort consequence of shedding it is much greater. This is a structural nonlinearity, not a coefficient that needs updating, and the only two ways to capture it are data from the saturated regime or a physical model that contains the saturation.

13.5.2 Non-monotone flexibility

The composition of those effects produces a result that catches forecasters trained on mild data. Over the mild-to-hot range, hotter weather increases available flexibility, because more HVAC load is running and there is more to curtail. Over the hot-to-extreme range it decreases: comfort margins are already consumed, so the admissible setpoint offset shrinks; equipment is saturated, so there is no cycling headroom; overrides rise, so the realized response falls below the physical one. Flexibility as a function of temperature therefore rises and then falls.

Figure 13.7 draws the shape. The horizontal axis is outdoor temperature from 20°C to 45°C, the vertical axis is available fleet flexibility in megawatts, the solid gold curve is the illustrative relationship $F(T) = 15 + 55\exp(-((T-33)/6)^2)$ MW, and the dashed blue line is a linear trend fitted on the region $T \le 33$°C and extended beyond it. The three shaded bands label the mild, hot and extreme regimes. The curve peaks near 33°C at about 70 MW and falls to about 21 MW at 42°C; the extrapolated trend reaches about 94 MW at the same temperature. The gap at 42°C is the entire operating question: a model that learned the rising portion and extended it would commit four times the capacity the fleet can deliver. Extrapolating a monotone trend is unsafe whenever the mechanism that produced the trend has a saturation in it, and almost every physical mechanism does.

13.5.3 Action-distribution shift and safe action sets

The response model of Chapter 12 was fitted on dispatch events with offsets in $u \in [0, 2]$°C, because that is what the program has ever done. An optimizer handed that model as a constraint will search over whatever action set it is given, and if the set includes $u = 4$°C it will evaluate the model there. The model will return a number, probably a large one, because within the

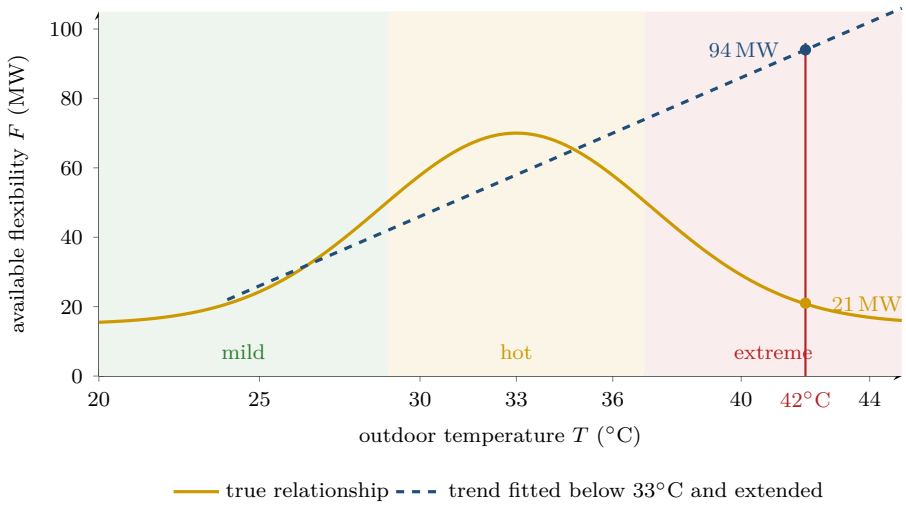


Figure 13.7: Available fleet flexibility against outdoor temperature. The solid curve is the illustrative relationship $F(T) = 15 + 55\exp(-((T-33)/6)^2)$ MW; the dashed line is the least-squares trend of that curve over $T \le 33$°C, approximately $F = 4.0\,T - 74$, extended to 45°C. Bands label the three regimes of Section 13.5.1.

fitted range response grows with depth. At 4°C in reality, comfort limits bind, override rates rise sharply, equipment behavior changes, and the delivered response is nothing like the extension of the fitted trend. This is the control-policy shift of Section 13.2.3 seen from the decision side, and the remedy belongs to the optimization problem rather than to the model: constrain the optimization domain to the region where the model has support.

Safe action sets. Make the constraint explicit. Define, for each state X, the set of actions with adequate historical or physical support,

$$\mathcal{U}_{\text{safe}}(X) = \{\, u : \text{the response model has support at } (X, u) \,\}, \tag{13.20}$$

$$u^* = \arg \max_{u \in \mathcal{U}_{\text{safe}}(X)} \text{objective}(u), \tag{13.21}$$

rather than optimizing over the whole nominal action space. Support can be operationalized with the tools of Section 13.3.3: a bound on the Mahalanobis distance of (X, u) from the training distribution, a minimum count of historical events within a neighborhood of (X, u), or a physical bound such as a maximum admissible comfort excursion. Restricting the feasible set of equation (13.21) to equation (13.20) is a small change to the optimization problem with a large effect: it removes from consideration exactly those actions whose predicted value is an extrapolation artifact.

13.5.4 Optimizer exploitation of model error

The reason the constraint is necessary, rather than merely prudent, is that an optimizer is an adversarial consumer of a model. An analyst who inspects a forecast looks at typical cases and will not notice that the model overestimates the response by a few megawatts at one unusual combination of state and action. An optimizer evaluating thousands of candidate actions per dispatch will find that combination, because finding the input at which the model is most optimistic is precisely what maximizing the model's output does. The phenomenon is known as model exploitation, or the optimizer's curse: optimization is biased toward the regions where the model's error is most favorable, which are typically the regions where the model has least data.

Example 13.1 (An action chosen because the model is wrong there)**.** Let the true fleet response to a large, unusual offset be $F(u) = 30\,\mathrm{MW}$, and let the fitted model predict 50 MW there because it has extrapolated a trend learned at small offsets. For every normal action the model is accurate to within a megawatt. An optimizer maximizing committed capacity compares all candidates, finds that the unusual action scores 50 against the normal actions' 40, and selects it. The fleet delivers 30 MW against a commitment made on the basis of 50. The shortfall is 20 MW, and it was caused not by the model being bad on average but by the optimizer selecting the one point where it was bad. A forecasting model validated as a forecaster can therefore be unsafe as the input to a decision, and the validation that matters is validation over the action set the optimizer will actually search.

13.5.5 Stress testing and the validated envelope

If aggregate validation hides the failure, the remedy is to validate deliberately where failure is expected. Table 13.4 lists the six stress scenarios a fleet forecaster should be tested on; the columns are the scenario, why it is stressful, and the metric that reveals the weakness. The common feature of the six is that each is rare, so each contributes almost nothing to an aggregate score, and each is a condition under which the fleet is likely to be dispatched.

Table 13.4: Stress scenarios for a fleet flexibility forecaster.

Scenario	Why it stresses the model	What to measure
Hottest days	outside temperature support; devices saturated	RMSE and coverage on the top 1 % of hours
Lowest telemetry availability	inputs imputed, population self-selected	error as a function of completeness
Largest dispatch actions	outside action support	realized versus predicted response by offset depth
New cohorts	no history for the segment	error by cohort tenure
Post-event rebound	payback dynamics, not steady state	error in the hours after an event ends
Abnormal composition	mixture weights far from the fitted mix	error against cohort-fraction deciles

Backtesting extreme subsets. The numbers such a test produces are usually stark. A forecaster with an overall root-mean-square error of 5 MW may have 18 MW on the hottest one percent of days. Both numbers are true, and only the second describes the condition under which the capacity commitment is settled. The appropriate way to publish the pair is a model card stating a validated operating envelope,

$$\mathcal{D}_{\text{validated}} = \{X : T \le 38^{\circ}\mathrm{C},\ N_{\text{avail}} > 60\,000,\ \dots\}, \tag{13.22}$$

with the error statistics quoted inside it and an explicit instruction that forecasts outside it are to be treated conservatively. Equation (13.22) is the honest form of a performance claim: a number with the region over which it was measured attached to it.

13.5.6 Confidence gating, fallbacks and ensembles

Equation (13.22) becomes operational when the system branches on it. *Confidence gating* evaluates membership of $\mathcal{D}_{\text{validated}}$ at issue time and routes the forecast accordingly: inside the envelope the normal model's output is published as usual, outside it a conservative path is taken.

The conservative path has four components, which are applied together rather than chosen among: widen the uncertainty interval, reduce the capacity committed against the forecast, fall back to a simpler and more robust model, and raise the case for operator review.

Figure 13.8 draws the flow. The input on the left is the feature vector at issue time; the diamond is the membership test of equation (13.22); the upper branch is the normal path, a single box containing the production model and its published forecast; the lower branch is the conservative path, with the four boxes above. Both branches terminate in the same published forecast object, which is the figure's main point: gating changes how the number and its interval are produced, not the interface the downstream optimizer consumes. The second point is that the test sits before the model rather than after it, so the decision to be cautious is taken on evidence about the input and not on the model's own self-assessment, which Section 13.3.4 showed cannot be trusted outside the support.

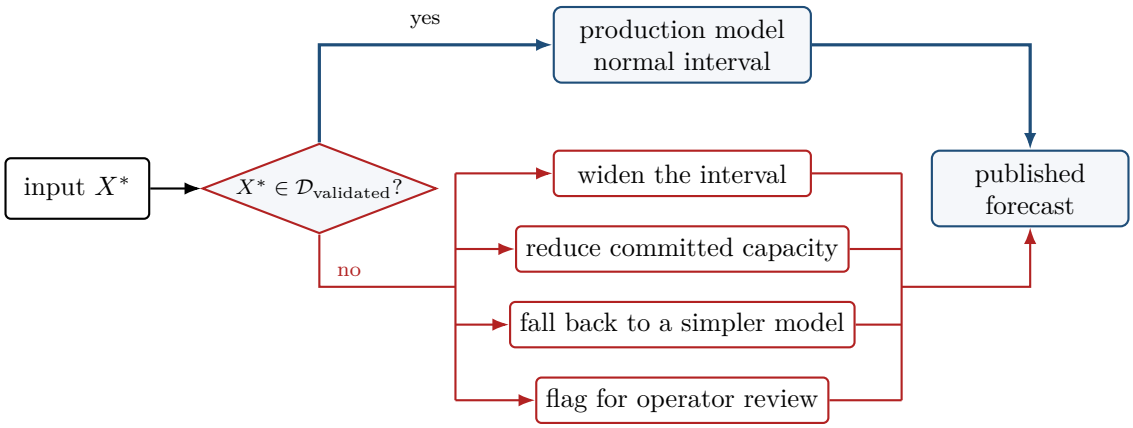


Figure 13.8: Confidence gating. The membership test of equation (13.22) is evaluated on the input before the model runs; the normal path publishes the production forecast, and the conservative path applies all four mitigations. Both paths emit the same kind of object.

Fallback models. The third box in the conservative path deserves its own argument. Suppose the production forecaster is a recurrent network of Chapter 7 that requires a complete sequence of fleet telemetry, and telemetry completeness has fallen to 60 %. The sophisticated model is now operating on inputs it was never validated on. A fallback such as seasonal persistence corrected for weather, of the kind built in Chapter 3, needs almost no telemetry: it needs a historical load shape and a temperature forecast. It is less accurate under normal conditions, by a margin that is easy to measure, and it is far more robust to missing inputs. The design principle is graceful degradation rather than an implicit choice between an excellent forecast and no forecast at all, and the fallback's normal-condition penalty is the premium paid for its availability when the primary model's inputs are gone.

Ensembles as a shift diagnostic. A second use of model diversity is as a detector. Run a gradient-boosted model, a recurrent network and a physics-based model in parallel and compare their point forecasts. Suppose they return 50, 48 and 49 MW: close agreement, three different inductive biases reaching the same answer, which is evidence that the answer is driven by the data rather than by any one model's extrapolation. Now suppose they return 70, 45 and 43 MW. The spread is not a bug in one of them. Three model classes extrapolate differently, as Figure 13.3 showed, so their disagreement grows precisely where the data thin out. Ensemble spread is not a proper quantification of uncertainty, since the members share training data and features and can be jointly wrong, but it is a cheap and informative warning that the current input is in a region where the answer depends on the model rather than on the evidence.

13.5.7 Recalibration, and drift versus novelty

Point accuracy and interval honesty fail separately and are repaired separately. A monitored forecaster may show stable mean absolute error while its nominal 90 % intervals cover only 75 % of outcomes. The center of the distribution is fine; the width is wrong. The appropriate response is to recalibrate the uncertainty layer alone, for example by re-estimating the predictive quantiles $[\hat{q}_{0.05}, \hat{q}_{0.95}]$ from recent residuals, or by multiplying the interval half-width by the factor that restores observed coverage on a recent window. Refitting the point model would not help and might hurt. Point adaptation and uncertainty calibration are two processes, with their own triggers, their own data windows and their own acceptance tests.

Drift and novelty call for different responses. The last distinction of this section is between the two temporal shapes of change. *Drift* is gradual: an override rate that moves from 5 % to 6 % to 7 % over successive quarters. Gradual adaptation handles it; the exponential weighting of equation (13.17) or a periodic refit of the behavioral component will track a trend of that shape without any special machinery, and no individual forecast is badly wrong while it does so. *Novelty* is abrupt and discontinuous: a firmware release, a new device type, a temperature four degrees above the historical record. Gradual adaptation is the wrong tool, because the historical data contain no information about the new condition and averaging them in more slowly does not create any. Novelty calls for explicit caution: the gating of Figure 13.8, widened intervals, reduced commitments, and a deliberate program to collect data under the new condition.

13.6 Operating under shift

The instruments and structural responses are now in place. This section assembles them into the operating posture a forecasting system should adopt, works a single example end to end, connects the posture to the optimization of Chapter 14, and closes with the monitoring design, the deployment practices and the questions an a reviewer is likely to ask about all of it.

13.6.1 What it means for a model to know that it does not know

The phrase is used loosely. Operationally it means that the system emits, alongside the forecast, a set of specific signals, and that a deterioration in any of them reduces the trust placed in the number. Five signals cover most of what is available:

1. the input is far outside the training support, by the range checks of Section 13.3.1 or the distance of equation (13.14);
2. models with different inductive biases disagree, as in Section 13.5.6;
3. recent residuals are worse than the reference level of Table 13.1;
4. interval coverage is deteriorating, as in Figure 13.5;
5. input missingness is abnormal, as in Section 13.4.2.

There is no single universal confidence number that summarizes these. They measure different things, they fail independently, and collapsing them into one scalar destroys the information an operator needs to decide what to do about the warning.

Confidence should have a basis. The consequence for communication is that the question "what is the model confidence?" cannot be answered as posed. The useful reply asks in what sense: the width of the predictive interval, the empirically verified calibration of that interval on recent data, the distance of today's input from the training distribution, the degree of disagreement among models, or the completeness of the input data. These can and do point in opposite directions. The interval can be narrow while the input is far outside support, which is

the configuration of Section 13.3.4 and the most dangerous one. A forecasting service should therefore expose several diagnostics explicitly rather than a single confidence score.

13.6.2 A worked example of conservative adjustment

Put the pieces together on one forecast. Tomorrow's forecast high is 42°C against a historical maximum of 38°C in the training record. Overnight, telemetry completeness has fallen from its usual 95 % to 72 %. The model is run and reports

$$\mathrm{P50}(F) = 70\,\mathrm{MW}, \qquad \mathrm{P10}(F) = 66\,\mathrm{MW}, \tag{13.23}$$

an $[66, 74]$ MW band at the stated levels. That interval should be treated as evidence of a problem rather than as information about the fleet. It is approximately as narrow as the intervals the model publishes on ordinary days, and two separate out-of-distribution conditions hold simultaneously: the weather is four degrees beyond the support, and the missingness has shifted by twenty-three points. A narrow interval under those conditions is not a sharp forecast. It is the signature of an uncertainty layer that has extrapolated its own spread.

The conservative response has five steps. Detect the out-of-distribution condition, by the range check on temperature and the missingness monitor. Widen the interval, by an amount justified by the extreme-subset backtest of Section 13.5.5 rather than chosen arbitrarily: if the hottest one percent of days carry 18 MW of error against an overall 5 MW, the interval should widen by something like that ratio. Reduce the firm capacity offered, so that the commitment is not made at the nominal P10 of 66 MW but at a lower figure; a conservative capacity of 50 MW is a defensible choice given a backtest error of that magnitude. Increase the reserve held behind the commitment. And refresh the forecast as telemetry arrives through the day, since the missingness component of the problem may resolve itself and the weather component will not. The nominal P10 was 66 MW and the offered capacity is 50 MW; the difference is the price of operating outside the validated envelope, and it is the correct direction for unfamiliar conditions to move operational confidence.

13.6.3 Connection to optimization: robust and stochastic thinking

The reduction from 66 to 50 MW can be expressed inside the decision problem rather than applied as a manual haircut. A commitment decision of the kind Chapter 14 formalizes is usually written as a chance constraint: offer a quantity such that the probability of delivering it is at least some level. Under normal conditions the requirement might be

$$\Pr(F \geq 75\,\mathrm{MW}) \geq 0.95. \tag{13.24}$$

Under shift, the object that equation (13.24) is evaluated against is itself less trustworthy, because the distribution of F was calibrated on a distribution that no longer applies. Two adjustments follow from the same observation: tighten the required level, to $\Pr(F \geq 75\,\mathrm{MW}) \geq 0.99$, so that the decision is made further into a tail that may be misplaced; or reduce the quantity committed, which moves the threshold rather than the level. Both buy margin against the possibility that the stated probabilities are wrong.

Robust and stochastic formulations. The distinction behind that adjustment is a standard one. *Stochastic* optimization takes the probability distribution as given and optimizes an expectation or a chance constraint under it, which is the right posture when the distribution is trustworthy. *Robust* optimization does not trust the distribution; it posits a set of plausible outcomes and protects against the worst case within it. Under shift, the honest statement about tomorrow is often of the robust form: flexibility will be somewhere in $[40, 60]$ MW, and the probabilities within that range are poorly known. A decision taken against the worst case of

that set, or against a small number of deliberately conservative scenarios, does not require the probabilities the shifted model cannot supply. Chapter 14 develops both formulations; the point here is that distribution shift is the condition that moves a problem from the first category into the second.

13.6.4 Monitoring design and cohort monitoring

A monitoring system for a fleet forecaster organizes into four categories, listed in Table 13.5 with what each category watches and the lag at which it reports. The categories are ordered by lag: data health is immediate, feature drift is immediate, forecast performance waits for outcomes, and probabilistic reliability waits for enough outcomes to estimate a frequency. A system with only the third and fourth categories learns about every problem late, which is the usual state of a monitoring stack that grew out of a model evaluation script.

Table 13.5: The four categories of a monitoring design for a fleet forecaster.

Category	What it watches	Lag
Data health	missingness rates, ingestion latency, impossible values, stale sensors	immediate, at issue time
Feature drift	temperature distributions, N_{avail}, duty cycle, device mix, dispatch depths	immediate, at issue time
Forecast performance	rolling MAE and Bias, event underdelivery, error by horizon	one settlement cycle
Probabilistic reliability	interval coverage, quantile calibration, by horizon and regime	many outcomes, to estimate a frequency

Monitoring by cohort. Every metric in Table 13.5 must be computed per segment as well as for the fleet, for the reason Chapter 11 gave about attribution. Suppose a firmware release doubles the mean absolute error of cohort B, which holds five percent of the devices. The fleet metric moves by a few percent, well inside its normal week-to-week band, and nothing is flagged. The problem then surfaces on the day cohort B is dispatched and underdelivers. Segment-level monitoring turns the same event into a localized anomaly with a named cohort, a release date and a software version attached to it, which is the difference between a diagnosis available in week one and a surprise in month three.

13.6.5 Retraining cadence, champion-challenger and shadow deployment

The remaining design question is how often to refit. The options are daily, weekly, monthly, or on a drift trigger, and the trade-off is the same variance-against-staleness trade-off as in Section 13.4.4, now applied to the schedule rather than to the window. Frequent retraining adapts quickly and risks instability: the model's parameters move in response to a single unusual week, forecasts become non-reproducible, and transient noise and bad data are absorbed into production. Infrequent retraining is stable, auditable and slow to track real change. The resolution depends on two rates: how quickly the system actually changes, and how much genuinely new information each period brings. Retraining should be tied to the arrival of validated new information, not to the calendar.

Champion-challenger. The deployment pattern that makes retraining safe is to treat a retrained model as a candidate rather than as a replacement. The *champion* is the model in

production; the *challenger* is the retrained candidate. Both are run on the same inputs over a trial period and scored on the same metrics, and the challenger is promoted only if it improves consistently, across horizons and cohorts, rather than in a single aggregate number over a single window. The pattern also supplies the counterfactual needed to tell whether a refit helped at all, which a straight replacement does not.

Shadow deployment. The stronger version runs the new model in forecast-only mode: it produces forecasts, those forecasts are recorded and compared against the champion's and against realized outcomes, and nothing the new model emits controls a device. Shadow mode is worth the operational cost whenever forecast errors have physical consequences, which in a demand-response program they do: a bad forecast can cause an unnecessary event, an uncomfortable home, or an undelivered capacity commitment with a financial penalty attached. Shadow evaluation also supplies exactly the evidence that a champion challenger promotion decision requires, which is why the two practices are usually deployed together.

13.6.6 Conceptual questions

The chapter's arguments recur as conceptual questions, and the four the lesson raised are worth stating with the shape of a good answer.

Conceptual questions

"Tomorrow is hotter than any day in our training set." The answer should begin by refusing the premise that historical error statistics apply, and by naming the situation as out-of-distribution operation rather than as a large input. It then quantifies: how far outside the validated range of equation (13.22) the key inputs are, measured per variable and jointly. It compares several models with different inductive biases and brings in the physical constraints of equation (13.12), so that the answer is bounded by what the fleet can physically do rather than by a fitted trend. It widens the reported uncertainty or reduces the firm commitment, as in Section 13.6.2. It commits to monitoring and reissuing as telemetry arrives through the day. And it closes on the optimizer: the dispatch optimization must be constrained to supported actions by equation (13.20), so that an aggressive action is not selected because the model is optimistic about it. A strong answer reaches the last point unprompted, because that is the step that converts a forecasting caveat into an operational safeguard.

Conceptual questions

"How would you know your model is drifting?" A strong answer separates input monitoring from outcome monitoring. On the input side: feature distributions, missingness rates, fleet composition and the distribution of dispatch actions, all available at issue time. On the outcome side: rolling bias and mean absolute error, event underdelivery, and interval coverage, each sliced by horizon and by cohort rather than aggregated, for the reason Figure 13.5 illustrates. The answer should then make the point of Section 13.4.2 explicitly: before acting on any detected change, establish whether it is a real change in the system or a data-pipeline failure, because retraining on a broken feed is worse than the drift it was meant to fix.

Conceptual questions

"Why isn't retraining every day automatically best?" Because a retraining job absorbs whatever arrived yesterday, including transient anomalies, a one-week firmware bug and corrupted telemetry, and because parameters that move every day make production behavior unstable and forecasts hard to audit or reproduce. The answer should tie retraining to the arrival of validated new information rather than to the calendar, and should name the practices that make a refit safe: champion-challenger comparison and shadow evaluation, so that a candidate model is promoted on consistent evidence across horizons and cohorts rather than on a single window's aggregate score.

Conceptual questions

"How does distribution shift affect probabilistic forecasts?" The key statement is that calibration is conditional on the distribution it was measured on. A nominal 90 % interval verified on historical data may cover far less than 90 % after a shift, as in Section 13.3.5, and models tend to remain overconfident outside their support rather than widening on their own, as in Section 13.3.4. The operational consequences are to monitor conditional coverage by regime instead of aggregate coverage, to pair the probabilistic output with out-of-distribution diagnostics, and to widen the intervals deliberately when those diagnostics fire, since the uncertainty layer will not do it by itself.

13.6.7 A compact mental model

The chapter reduces to one question asked before every forecast is trusted: is tomorrow drawn from the same world that validated this model? If it is, the historical validation is informative and the intervals mean what they say. If it is not, caution is warranted in proportion to how far outside that world tomorrow lies.

Three sub-questions make the test operational, and they correspond to the three factorizations of Section 13.1.3. Has X changed: the weather, the fleet composition, the device state? That is covariate shift, diagnosable from the inputs alone. Has u changed: is the control policy proposing actions the model has not seen? That is action-distribution shift, and it is self-inflicted and therefore controllable. Has $p(Y \mid X, u)$ changed: has firmware, equipment or customer behavior altered the response to identical conditions and identical commands? That is concept drift, and it is visible only in outcomes. A forecasting team that asks those three questions on a fixed cadence has most of the practice of this chapter.

13.7 What to retain

Definition 13.1 (Forecasting under distribution shift)**.** Distribution shift is the condition that the joint distribution of inputs and targets in operation, p_{future}, differs from the one the model was fitted on, p_{train}. In an energy fleet it is produced by extreme weather, changing composition, firmware releases, customer behavior and new control policies. The critical distinction is between interpolation, where predictions are constrained by nearby data, and extrapolation, where they are determined by the model's functional form. The practice is to monitor input drift, missingness, residual performance, calibration, cohort behavior and action-space support; and the rule is that outside the validated domain, the reported uncertainty and the operational conservatism should both increase.

Table 13.6 collects the chapter's objects with the subsection that defines each.

Table 13.6: The objects of forecasting under distribution shift.

Object	Definition	Section
p_{train}, p_{future}	training and operating joint distributions	13.1.1
covariate shift	$p(X)$ moves, $p(Y \mid X)$ holds	13.1.4
concept drift	$p(Y \mid X, u)$ changes	13.1.5
π_t	cohort mixture weight; composition shift	13.2.1
$p(X, u)$	the joint state-action distribution a response model needs	13.2.3
interpolation, extrapolation	inside and outside the training support	13.2.4
$0 \le F_t \le P_{\text{HVAC},t}$	physical bound that holds where data do not	13.2.6
range check	per-variable test against the training support	13.3.1
joint support	the region the training inputs jointly occupy	13.3.2
$d(X^*)$, $D_M^2(X)$	nearest-neighbor and Mahalanobis distance to the training set	13.3.3
conditional coverage	interval coverage computed by operating regime	13.3.6
$\mathbb{E}[e_t]$, $\text{Var}(e_t)$, MAE_t	rolling residual monitors	13.4.1
Pr(missing)	missingness as a monitored quantity	13.4.2
$w_i = \exp(-\lambda \Delta t_i)$	age weighting for gradual adaptation	13.4.4
$f_{\text{physics}} + f_{\text{behavior}}$	partition by rate of change	13.4.5
D	duty cycle; saturation defines the regime	13.5.1
$\mathcal{U}_{\text{safe}}(X)$	actions with model support	13.5.3
model exploitation	optimizers select where the model errs favorably	13.5.4
$\mathcal{D}_{\text{validated}}$	the envelope over which performance was measured	13.5.5
confidence gating	routing on envelope membership at issue time	13.5.6
drift, novelty	gradual change and abrupt unprecedented change	13.5.7
robust formulation	optimizing against a set when probabilities are untrusted	13.6.3
champion-challenger	promote a retrained model only on consistent gains	13.6.5

13.8 Exercises

Exercise 13.1. Take the three extrapolations of Figure 13.3: the tree with leaf value 45 MW above 35°C, the line $F = 2T - 20$, and the saturating form $\min(2T - 20, 55)$. (a) Evaluate all three at $T = 33, 36, 39$ and 42°C and tabulate the pairwise differences. (b) State the temperature above which the three disagree by more than 5 MW. (c) The true relationship is the saturating one. Compute the mean absolute error of each model over the grid $T \in \{25, 27, 30, 32, 35\}$ of equation (13.10) and over the grid $T \in \{39, 41, 43\}$, and explain why a validation score computed on the first grid cannot rank the models on the second.

Exercise 13.2. A fleet's training inputs are temperature and relative humidity with means $\mu = (28°\text{C}, 62\,\%)$, standard deviations $(5°\text{C}, 18\,\%)$ and correlation -0.7. (a) Write the covariance matrix Σ and invert it. (b) Compute the Mahalanobis distance D_M of equation (13.14) for the point $(38°\text{C}, 90\,\%)$ of Figure 13.4 and for the point $(38°\text{C}, 26\,\%)$. (c) Compute the standardized Euclidean distance for both points. (d) Explain which of the two points is the unprecedented one and which of the two distance measures says so.

Exercise 13.3. Use the weekly errors of Table 13.1. (a) Compute the mean and standard deviation of weeks one to four and express weeks five and six as deviations in those standard-deviation units. (b) For each candidate cause in Table 13.2, state one query against the monitoring data of Table 13.5 that would confirm or exclude it. (c) Suppose the diagnosis is a temperature feed that has been reporting a constant 21°C since the start of week five. Explain what a model retrained on weeks one to six would learn about the temperature coefficient, and what would happen to its forecasts after the feed is repaired.

Exercise 13.4. Apply the conservative adjustment of Section 13.6.2. The model reports $\text{P50}(F) = 70$ MW and $\text{P10}(F) = 66$ MW, and the extreme-subset backtest of Section 13.5.5 gives RMSE = 18 MW on the hottest one percent of days against 5 MW overall. (a) Assuming a normal predictive distribution, infer the implied standard deviation from the reported P50 and P10, and compare it with the backtest value. (b) Recompute P10 using the backtest standard deviation. (c) Evaluate $\Pr(F \geq 50\,\text{MW})$ under both standard deviations, and comment on whether a 50 MW firm commitment satisfies equation (13.24) at the 0.95 level and at the 0.99 level.

Chapter 14

From forecast to optimization: chance constraints, expected cost and receding horizons

14.1 Information versus action

Every chapter so far has produced information. Chapters 5 to 8 built models that return $\hat{y}_{t+h}$, Chapter 9 measured how far that number lands from the outcome, Chapter 10 replaced the number with a distribution, Chapter 11 took the distribution to fleet scale, and Chapters 12 and 13 asked whether the quantity being forecast is the counterfactual the operator needs and whether the model is still valid at the conditions it is being asked about. Nothing in any of them dispatched a device. This chapter's question is

$$\textit{Given } p(F_{t+h} \mid \mathcal{I}_t, u)\textit{, what should the operator actually do?}$$

The answer is an optimization problem, and the forecast enters it as data. The chapter develops that problem in the two forms an operator uses, a reliability constraint and an expected cost; extends it over time, where it becomes model predictive control; extends it over uncertainty, where it becomes stochastic or robust optimization; and then returns to the forecasting system to ask what it must emit for the optimizer to be able to use it.

Jargon: chance constraint, expected cost, reserve, model predictive control, receding horizon, scenario, robust optimization, mean-variance

A *chance constraint* requires that a condition hold with at least a stated probability, for example $\Pr(R_{\text{total}} \geq D) \geq 0.95$, rather than requiring it to hold always or on average. An *expected-cost* objective replaces the constraint with a price: every outcome is assigned a cost and the decision minimizes the probability-weighted average. *Reserve* is capacity held back and not committed, carried so that a shortfall in the committed capacity can still be covered. *Model predictive control* (MPC) is the control scheme that, at each decision time, forecasts the near future, optimizes a whole sequence of future actions, implements only the first of them, and then repeats; the repeated re-solve over a window that slides forward is the *receding horizon*. A *scenario* is one complete sampled trajectory of the uncertain quantities over the planning window, internally consistent in time. *Robust optimization* requires the decision to be feasible for every outcome in a declared set, rather than with a stated probability. A *mean-variance* objective trades expected performance against its variance through a single risk coefficient, as in financial portfolio selection.

Key idea

A forecast does not make a decision. The decision is made by an optimizer, and what the optimizer needs from the forecast is not a central estimate but the downside tail, the correlations and the dependence on the planned action. A forecast is valuable exactly to the extent that it changes the decision that follows it.

This section states the dispatch problem, shows what happens when a point forecast is fed into it, and shows that replacing the point forecast with an expectation is not enough either.

14.1.1 Forecasting alone does not make a decision

Suppose the forecasting system reports $\hat{F} = 50\,\text{MW}$ of available flexibility for the 17:00 hour. That statement, taken alone, does not determine a single operational action. It does not say which cohorts to call, how strongly to move their setpoints, how long to hold the event, how much capacity to leave uncommitted as reserve, or how much customer discomfort the operator is willing to spend to obtain the response. A second forecast reporting the same 50 MW with half the uncertainty would support a different action, and so would the same forecast under a different penalty for underdelivery.

The division of labor is clean once stated. Forecasting produces information: a distribution over what the fleet could do. Optimization produces action: a choice of control, subject to constraints, that is best under some objective. The two are different objects with different success criteria, and the rest of this chapter is about the interface between them.

14.1.2 The dispatch problem

The smallest problem that contains the real difficulty is a static, single-hour dispatch over N cohorts. Each cohort i is called at a dispatch level x_i, a scalar encoding how hard it is pushed (the fraction of the cohort enrolled in the event, the size of the setpoint offset, or both), and delivers a response $R_i(x_i)$ in megawatts. The system needs D megawatts. The problem is

$$\min_{x} \; C(x) \qquad \text{subject to} \qquad \sum_{i=1}^{N} R_i(x_i) \;\geq\; D, \tag{14.1}$$

with $x = (x_1, \ldots, x_N)$ the decision vector. The cost $C(x)$ is not a single meter reading. It accumulates customer discomfort, device wear from additional compressor or inverter cycling, any incentive payment made per event or per megawatt, the rebound load the event creates after it ends, and the opportunity cost of using a cohort now rather than keeping it available for a later and possibly more valuable call. Several of those terms will reappear as separate pieces of the objective later in the chapter.

Equation (14.1) is the template, and everything that follows modifies it in one of three ways: the constraint becomes probabilistic, the objective absorbs the risk as a cost, or the whole problem is indexed by time.

14.1.3 Point-forecast dispatch and its failure

Take three cohorts with forecast responses at full dispatch

$$\hat{R}_1 = 30\,\text{MW}, \qquad \hat{R}_2 = 40\,\text{MW}, \qquad \hat{R}_3 = 50\,\text{MW}, \tag{14.2}$$

and a requirement $D = 75\,\text{MW}$. Treat each x_i as binary, call or do not call, and let cost grow with delivered megawatts. The feasible subsets are $\{1, 3\}$ at 80 MW, $\{2, 3\}$ at 90 MW and $\{1, 2, 3\}$ at 120 MW; the pair $\{1, 2\}$ delivers only 70 MW and is infeasible. The cheapest feasible

choice is therefore $x = (1, 0, 1)$, with an expected delivery of 80 MW against a requirement of 75. A deterministic solver returns this answer, and it is correct given its inputs.

The inputs were forecasts. Suppose the event runs and cohort 1 delivers $R_1 = 25\,\text{MW}$ while cohort 3 delivers $R_3 = 42\,\text{MW}$, both within a few megawatts of their forecasts. The total is 67 MW against a requirement of 75: an 8 MW shortfall, with the unused cohort 2 sitting idle beside it. The optimizer solved its problem perfectly and the system failed. The failure is not an error of arithmetic but of formulation: uncertain inputs were treated as certain, so the problem that was solved was not the problem the operator had. Every construction in this chapter exists to repair that one defect.

14.1.4 Expected-value optimization is not enough

The first repair is to acknowledge that the responses are random variables and to write the constraint on their means,

$$\sum_{i=1}^{N} \mathbb{E}[R_i]\, x_i \;\geq\; D. \tag{14.3}$$

This is an improvement, in that it at least names the forecasts as expectations rather than as facts, and it is still inadequate. Consider two cohorts with

$$R_A \sim \mathcal{N}(40, 2^2)\,\text{MW}, \qquad R_B \sim \mathcal{N}(40, 15^2)\,\text{MW}. \tag{14.4}$$

Equation (14.3) cannot distinguish them: both contribute 40 MW to the left-hand side. Operationally they are not interchangeable. Cohort A will deliver between 36 and 44 MW on essentially every occasion; cohort B will deliver below 25 MW about once in six events and above 55 MW about once in six. A commitment sized on means and met by cohort B will fail routinely.

The deficiency is structural rather than numerical. An expectation constraint controls the average of a sum, and reliability is a statement about the lower tail of that sum. Averages and tails are different functionals of the same distribution, and Chapter 10 already established that two distributions can share the first and differ arbitrarily in the second. The next section writes the constraint on the tail directly.

14.2 Reliability as a constraint

The natural way to encode a reliability requirement is as a constraint on a probability. This section defines the chance constraint, shows what it rejects that an expectation constraint accepts, converts it into a safety margin and thereby into a required expected response, and then draws the consequence for how forecasting models should be compared.

14.2.1 The chance constraint

Write the total delivered response as $R_{\text{total}}(x) = \sum_i R_i(x_i)$. A chance-constrained dispatch replaces the deterministic requirement of equation (14.1) with

$$\min_{x} \; C(x) \qquad \text{subject to} \qquad \Pr\!\left(R_{\text{total}}(x) \geq D\right) \;\geq\; 1 - \alpha, \tag{14.5}$$

where α is the accepted probability of shortfall. With $D = 75\,\text{MW}$ and $\alpha = 0.05$ the constraint reads $\Pr(R_{\text{total}} \geq 75) \geq 0.95$: among all dispatches, admit only those with at least a 95 % probability of meeting the target, and among those, take the cheapest.

Two things are worth noticing about the form. First, the constraint is on the distribution of a sum, so it needs the joint distribution of the cohort responses and not only their marginals; Section 14.3.4 makes that dependence explicit. Second, α is an input from outside the forecasting

system. It encodes the operator's tolerance for non-performance, which is set by market penalties, by the availability of reserves and by regulatory obligation, and no amount of forecasting work determines it.

14.2.2 What the constraint rejects

The value of equation (14.5) is visible in a case where it disagrees with the expectation constraint. Consider two candidate dispatches for the same 75 MW requirement:

$$\text{A}: \ \mathbb{E}[R_{\text{total}}] = 80\,\text{MW}, \ \Pr(R_{\text{total}} \geq 75) = 0.65, \qquad \text{B}: \ 85\,\text{MW}, \ 0.97. \tag{14.6}$$

Dispatch A calls fewer cohorts and is cheaper. It also fails to deliver about one event in three. A deterministic optimizer, or one constrained only on the mean, selects A, because 80 exceeds 75 and A is cheaper. The chance-constrained optimizer at $\alpha = 0.05$ declares A infeasible and selects B, paying for the extra cohort in order to buy reliability.

Figure 14.1 draws the two dispatches. The horizontal axis is the total delivered response R_{total} in megawatts and the vertical axis is forecast density; the two curves are the predictive distributions of the two dispatches, the vertical red line is the 75 MW requirement, and the shaded region is the part of dispatch A's distribution that falls short of it. The dashed vertical marks are the fifth percentile of each distribution, the level each dispatch can be expected to exceed nineteen times in twenty. The figure shows the chance constraint as a geometric test: the constraint $\Pr(R_{\text{total}} \geq 75) \geq 0.95$ is satisfied exactly when the dashed mark lies to the right of the red line. For dispatch B it does, at 76.3 MW. For dispatch A it lies far to the left, at 58.7 MW, even though A's mean is only 5 MW below B's. The whole difference between the two is width, and width is invisible to equation (14.3).

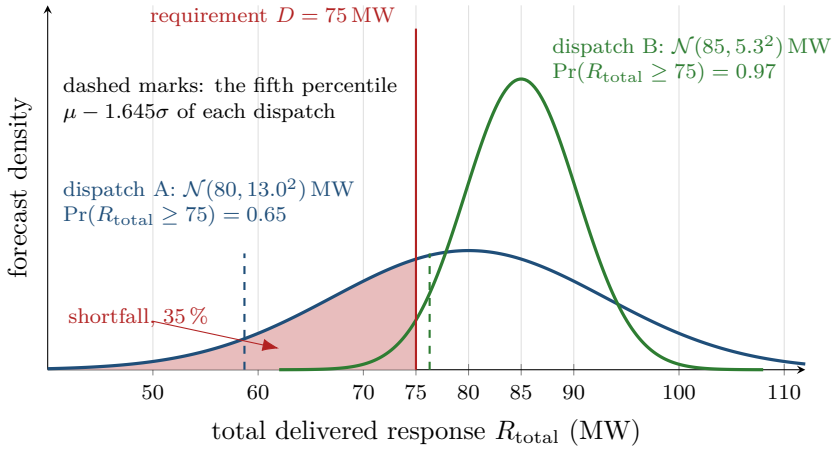


Figure 14.1: The chance constraint as a geometric test. The curves are the normal predictive distributions of the two dispatches of equation (14.6), $\mathcal{N}(80, 12.98^2)$ and $\mathcal{N}(85, 5.32^2)$ in MW, the standard deviations chosen so that $\Phi(5/12.98) = 0.65$ and $\Phi(10/5.32) = 0.97$. The shaded area is dispatch A's 35 % shortfall probability. The dashed marks are the fifth percentiles, $\mu - 1.645\sigma$; the constraint $\Pr(R_{\text{total}} \geq 75) \geq 0.95$ holds exactly when that mark lies right of the red line. Values are illustrative.

14.2.3 The safety margin and the required expected response

For a normal total the chance constraint has a closed form that makes the reserve explicit. If $R_{\text{total}} \sim \mathcal{N}(\mu, \sigma^2)$, then $\Pr(R_{\text{total}} \geq D) \geq 1 - \alpha$ is equivalent to

$$\mu - z_{1-\alpha}\,\sigma \ \geq \ D, \qquad z_{0.95} = 1.645, \tag{14.7}$$

where $z_{1-\alpha}$ is the standard normal quantile. The left-hand side is the fifth percentile of the delivered response, and the quantity $z_{1-\alpha}\sigma$ is a reserve requirement computed from forecast uncertainty rather than from a rule of thumb.

Reading equation (14.7) in the direction of diagnosis: a dispatch with $\mathbb{E}[R_{\text{total}}] = 80\,\text{MW}$ and $\sigma = 5\,\text{MW}$ has a fifth percentile of

$$80 - 1.645 \times 5 = 71.8\,\text{MW}, \tag{14.8}$$

which is below the 75 MW requirement. The dispatch is not safe enough, despite a 5 MW apparent surplus on the mean, because the surplus is smaller than the reserve the uncertainty demands.

Reading it in the direction of design: solving equation (14.7) for μ at $D = 75\,\text{MW}$ and $\sigma = 5\,\text{MW}$ gives

$$\mu \;\geq\; 75 + 1.645 \times 5 = 83.225\,\text{MW}. \tag{14.9}$$

To deliver 75 MW with 95 % confidence the operator must dispatch cohorts whose expected response totals at least 83.2 MW, holding 8.2 MW of implicit reserve. Forecast uncertainty and required reserve are the same quantity seen from two sides.

14.2.4 Better uncertainty unlocks capacity

Equation (14.7) is linear in σ, which turns a forecasting improvement into a number of megawatts. Figure 14.2 plots it. The horizontal axis is the forecast standard deviation σ of the total response in megawatts and the vertical axis is the expected response μ that must be dispatched to satisfy a 95 % chance constraint at $D = 75\,\text{MW}$; the solid line is $\mu = 75 + 1.645\sigma$, the dashed horizontal line is the requirement itself, and the three marked points are the cases of the text. At $\sigma = 10\,\text{MW}$ the operator must dispatch $75 + 16.45 = 91.45\,\text{MW}$ of expected response. At $\sigma = 5\,\text{MW}$ the figure is 83.23 MW, and at $\sigma = 4\,\text{MW}$ it is $75 + 6.58 = 81.58\,\text{MW}$. The vertical arrow between the first and last of those marks the difference, 9.9 MW.

That difference is the operational value of the forecast improvement. Cutting the forecast standard deviation from 10 MW to 4 MW releases almost 10 MW of flexibility that can be safely committed without enrolling a single additional device, without changing any customer's comfort band and without building anything. The same fleet becomes larger as a committable resource because less of it has to be held in reserve against forecast error.

> **Principle 14.1.** *Forecast uncertainty is a reserve requirement. Under a chance constraint the reserve is $z_{1-\alpha}\sigma$, so a reduction in forecast standard deviation converts directly into megawatts of additional safely committable capacity, at no capital cost and with no change to the physical fleet.*

14.2.5 Forecast value is not lower error alone

The reserve formula changes which forecasting model should be preferred. Suppose model A attains RMSE = 6.0 MW on the response and model B attains 6.2 MW. By the point metrics of Chapter 9, A wins. If B's predictive distribution is better calibrated in the lower tail, so that its stated fifth percentile really is exceeded 95 % of the time while A's is exceeded only 88 % of the time, then B is the model that lets the optimizer satisfy $\Pr(R_{\text{total}} \geq D) \geq 0.95$ and A is not. The decision depends on σ and on the honesty of the tail, and root-mean-square error summarizes neither.

This is the concrete form of a claim made in Chapter 9: the model with the best forecasting metric is not necessarily the model that produces the best decisions. The two orderings coincide only when the metric happens to be aligned with the decision, and a symmetric point metric is aligned with an asymmetric reliability requirement only by accident.

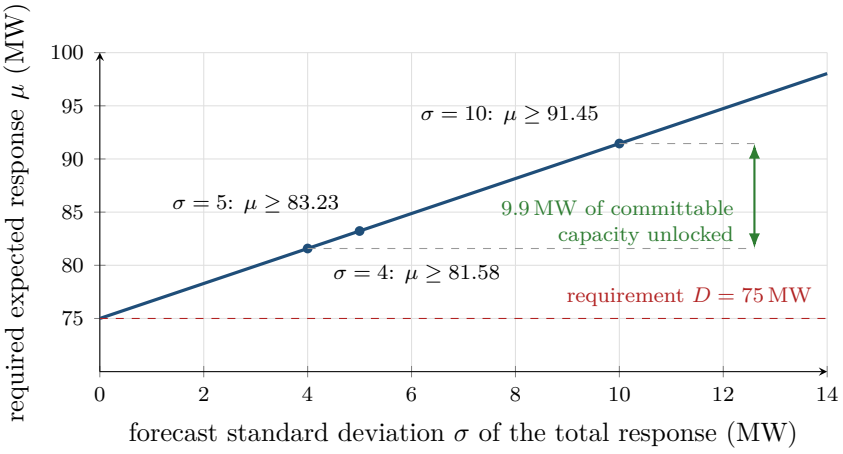


Figure 14.2: Required expected response against forecast uncertainty. The solid line is the closed form $\mu = D + z_{0.95}\sigma = 75 + 1.645\sigma$ from equation (14.7), for a 95 % chance constraint at $D = 75$ MW and a normal total response. The marked points are $\sigma = 4, 5, 10$ MW; the green arrow is the 9.87 MW difference between the requirements at $\sigma = 10$ and $\sigma = 4$.

14.2.6 Decision-focused validation

The remedy is to validate against the decision. The question to ask of a candidate model is not how large its errors were but what would have happened had this forecast driven the optimizer over the evaluation period. That question is answered by replaying history: for each past event, feed the candidate model's forecast to the dispatch optimizer, record the dispatch it would have chosen, and score it against what the fleet actually delivered. The metrics that come out are operational:

- the *target delivery rate*, the fraction of events in which the committed megawatts were delivered;
- the *underdelivery frequency and depth*, how often and by how much the commitment was missed;
- the *comfort-violation rate*, how often customers were pushed past their agreed bands;
- *total cost*, including incentive payments and non-performance penalties;
- *unused flexibility*, the capacity held in reserve and never called, which is the price paid for reliability; and
- *rebound cost*, the load the events created after they ended.

Table 14.1 compares two models on this basis. The columns are a point metric, a bias, the delivery success rate of a dispatch built to a 95 % reliability target, and the average capacity left unused. Model A has the lower root-mean-square error, 5.5 MW against 5.8 MW, and a positive bias of +1.0 MW, meaning it systematically overstates the response; its dispatches meet the target in 88 % of events against the 95 % they were designed for, and it leaves 3 MW unused on average. Model B is unbiased, meets the target in 96 % of events, and leaves 5 MW unused. The two models are not ordered by a single number: B buys seven percentage points of delivery success with 2 MW of additional idle capacity per event. Where reliability is the binding requirement, B is the better model despite losing on root-mean-square error, and the positive bias in A is the mechanism, because a model that overstates the response understates the reserve.

Table 14.1: Decision-focused comparison of two response-forecasting models under a 95 % reliability target; error and bias are in MW. Values are illustrative and follow Section 14.2.6.

Model	RMSE	Bias	Delivery success at a 95 % target	Average unused capacity	Reading
A	5.5	+1.0	88 %	3 MW	lower error, overstates the response, misses the reliability target
B	5.8	0.0	96 %	5 MW	higher error, unbiased, meets the target at the cost of idle capacity

14.3 Cost, quantiles and portfolios

A chance constraint sets reliability by decree at a level $1 - \alpha$ nobody derived. This section replaces the decree with prices, shows that an asymmetric price selects a forecast quantile, and then puts several cohorts together, where the covariance of their errors rather than their individual uncertainties governs the portfolio's reliability.

14.3.1 Asymmetric cost

Failure in the two directions does not cost the same. Suppose each megawatt of shortfall below the target costs \$1000, in penalty and in whatever emergency resource must cover it, while each megawatt delivered above the target costs \$100 in unnecessary incentive payments and customer discomfort. Writing the delivery error as $e = R_{\text{total}} - D$, the cost is piecewise linear and asymmetric,

$$C(e) = \begin{cases} 1000\,|e|, & e < 0 \quad \text{(underdelivery)}, \\ 100\,|e|, & e > 0 \quad \text{(overdelivery)}. \end{cases} \tag{14.10}$$

Figure 14.3 plots it. The horizontal axis is the delivery error e in megawatts, negative to the left where the fleet fell short and positive to the right where it delivered more than needed, and the vertical axis is the cost in dollars; the two straight segments are the two branches of equation (14.10), meeting at zero cost when delivery exactly matches the requirement. The left branch is ten times steeper than the right. The consequence for the optimizer is immediate: because moving one megawatt from the shortfall side to the surplus side saves \$900, the cost-minimizing dispatch is deliberately conservative, aiming above the requirement rather than at it. The asymmetry, not the absolute level of either price, is what sets the degree of conservatism.

14.3.2 Expected-cost optimization

Pricing the outcomes allows the reliability constraint to be dissolved into the objective. The expected-cost dispatch problem is

$$\begin{aligned} &\min_{x} \ \mathbb{E}[C(x, R)]\,, \\ &\text{with} \quad C = C_{\text{dispatch}} + C_{\text{underdelivery}} + C_{\text{comfort}}, \end{aligned} \tag{14.11}$$

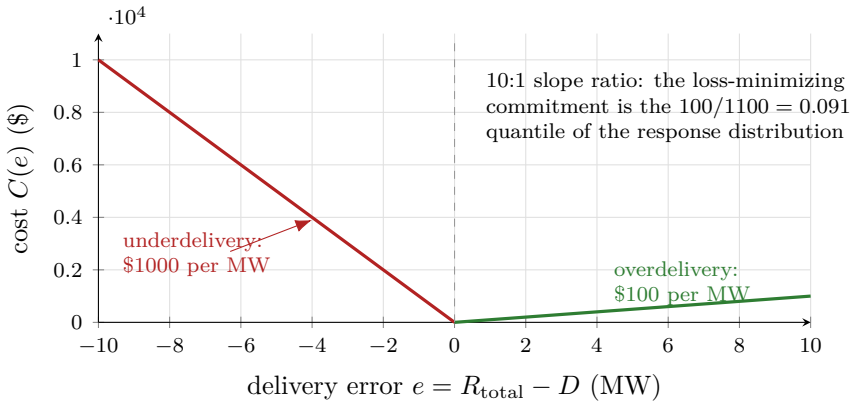


Figure 14.3: The asymmetric cost of equation (14.10): slope \$1000 per MW for $e < 0$ and \$100 per MW for $e > 0$, where $e = R_{\text{total}} - D$. The 10:1 ratio is what drives a conservative dispatch; the quantile it selects is derived in Section 14.3.3. Values are illustrative.

where the expectation is over the uncertain response R and the three terms are, respectively, what it costs to call the cohorts, what it costs to fall short, and what it costs the customers. The formulation balances risk against money without anyone having to choose a confidence level: a 2 % chance of shortfall is accepted when its priced consequence is smaller than the cost of preventing it, and rejected when it is not. Compared with a chance constraint at an arbitrary α, equation (14.11) is the more defensible object, because every quantity in it is in principle measurable.

Example 14.1 (Two dispatch options)**.** Option A dispatches a small set of cohorts for \$500, with a 20 % probability of failing to meet the commitment and a \$5000 penalty if it does. Option B dispatches more cohorts for \$900 and fails with probability 5 %. The expected costs are

$$\mathbb{E}[C_A] = 500 + 0.20 \times 5000 = \$1500, \tag{14.12}$$

$$\mathbb{E}[C_B] = 900 + 0.05 \times 5000 = \$1150. \tag{14.13}$$

Option B costs \$400 more to dispatch and \$350 less in expectation. The cheaper-looking option is the more expensive one, and only a formulation that prices the failure can see that.

14.3.3 From cost asymmetry to the operational quantile

An optimizer that consumes a probabilistic forecast does not always need the whole distribution. Where the decision is a single committable number, the cost structure selects a single quantile, and the quantile can be read straight off the forecast. Suppose the forecast for an hour is

$$\text{P10} = 40\,\text{MW}, \qquad \text{P50} = 50\,\text{MW}, \qquad \text{P90} = 62\,\text{MW}. \tag{14.14}$$

An operator who wants roughly 90 % confidence in delivery commits the P10, 40 MW, because $\Pr(F \geq \text{P10}) \approx 0.90$ by the definition of the quantile. The median, 50 MW, would be delivered only about half the time, and P90 would be delivered about one time in ten.

The choice of level is not a matter of taste once the costs are named. Let the commitment be c, with λ_{under} the cost per megawatt of shortfall and λ_{over} the cost per megawatt of surplus. Differentiating the expected cost with respect to c gives

$$\lambda_{\text{under}}\, F(c) - \lambda_{\text{over}}\,[1 - F(c)] = 0 \quad \Longrightarrow \quad F(c) = \frac{\lambda_{\text{over}}}{\lambda_{\text{under}} + \lambda_{\text{over}}}, \tag{14.15}$$

with F the predictive distribution function of the response. The prices of equation (14.10) give $100/1100 = 0.091$: commit the ninth percentile, which is close to the P10 rule of thumb and is now derived rather than assumed. The general reading of equation (14.15) is a dictionary. Symmetric costs select the median, P50. Costly underdelivery drives the level down toward P10 or below. Costly overcommitment, which arises when calling customers is expensive or when a surplus must be dumped, pushes it up. The cost structure determines which quantile of the forecast is the relevant one, which is why Chapter 10 insisted that the forecast product carry more than its center.

14.3.4 Several cohorts: the covariance decides

With more than one cohort the dispatch is a portfolio. Taking the response to be linear in the dispatch level, so that calling a fraction x_i of cohort i yields $x_i R_i$, the total is $R_{\text{total}} = \sum_i x_i R_i$ with

$$\mathbb{E}[R_{\text{total}}] = \sum_i x_i \mathbb{E}[R_i], \qquad \operatorname{Var}(R_{\text{total}}) = x^{\mathsf{T}} \Sigma\, x, \tag{14.16}$$

where Σ is the covariance matrix of the cohort responses. The mean is a weighted sum of the forecasts and nothing else; the variance depends on the off-diagonal entries, which are exactly the fleet correlation structure of Chapter 11. Substituting equation (14.16) into the margin form (14.7) gives the deterministic equivalent of the chance constraint for a normal total,

$$\sum_i x_i \mathbb{E}[R_i] \;-\; z_{1-\alpha} \sqrt{x^{\mathsf{T}} \Sigma\, x} \;\geq\; D, \tag{14.17}$$

a single inequality the optimizer can carry directly.

How much the off-diagonal entries matter is easiest to see with two cohorts of equal uncertainty. Let each have $\sigma = 5\,\text{MW}$ and let both be called fully. If their errors are independent,

$$\sigma_{\text{total}} = \sqrt{25 + 25} = \sqrt{50} = 7.07\,\text{MW}; \tag{14.18}$$

if they are perfectly correlated, the standard deviations add and $\sigma_{\text{total}} = 10\,\text{MW}$. The two cases have identical forecasts for each cohort and a 41 % difference in portfolio uncertainty, which by equation (14.7) is a 4.8 MW difference in required reserve.

Figure 14.4 draws the whole range. The horizontal axis is the correlation ρ between the two cohorts' response errors, from -1 to $+1$, and the vertical axis is the portfolio standard deviation in megawatts; the curve is the closed form $\sigma_{\text{total}} = 5\sqrt{2 + 2\rho}$ and the three marked points are perfect negative correlation, independence and perfect positive correlation. The curve is steepest near $\rho = -1$ and flattens as ρ approaches $+1$, which carries the practical message: most of the diversification benefit is obtained by moving from strong positive correlation to weak correlation, and the difference between $\rho = 0.6$ and $\rho = 0.9$ matters more than the difference between $\rho = 0$ and $\rho = -0.3$. Diversification helps only to the extent that the errors are not strongly correlated, and adding a second cohort whose errors move with the first adds megawatts without adding reliability.

Geographic diversification. The correlation in Figure 14.4 is physical, and it can be engineered. A thermostat cohort in Texas and a thermostat cohort in New York sit under different weather systems; the forecast error that makes the Texas response come in low is a Texas forecast error, and it says little about New York. Calling both therefore produces a more reliable total capacity than calling twice as much of either, even when the two cohorts' individual forecasts are equally uncertain. The same argument applies across resource classes, since a battery's availability error is nearly uncorrelated with a thermostat fleet's, and across customer segments. Portfolio construction for distributed energy resources is therefore not only a question of how much capacity is enrolled but of how the enrolled capacity's errors covary, which is the same reasoning that governs the construction of a financial portfolio.

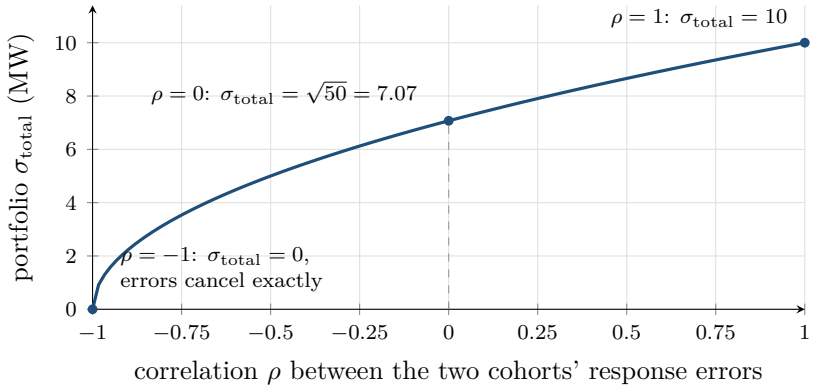


Figure 14.4: Portfolio uncertainty against correlation for two cohorts with $\sigma = 5\,\text{MW}$ each, both called fully. The curve is the closed form $\sigma_{\text{total}} = \sqrt{2\sigma^2(1+\rho)} = 5\sqrt{2+2\rho}$, a special case of equation (14.16). The marked points are equation (14.18) and its perfectly correlated counterpart.

14.3.5 Mean-variance and the comfort trade-off

The portfolio analogy extends to the objective. Expected response plays the role of return and delivery uncertainty the role of risk, and the classical mean-variance objective becomes

$$\max_x \ \Big[\mu(x) - \lambda\,\sigma^2(x)\Big], \qquad \mu(x) = \sum_i x_i \mathbb{E}[R_i], \quad \sigma^2(x) = x^{\mathsf{T}}\Sigma\,x, \tag{14.19}$$

with $\lambda \geq 0$ a risk-aversion coefficient. The formulation is intuitive and tractable, being a quadratic program when the cohort responses are linear in x, and it is not always the right description of the operator's preferences: variance penalizes upside deviation as heavily as downside, whereas an aggregator is indifferent to delivering more than promised and acutely sensitive to delivering less. Where that asymmetry matters, the chance constraint of equation (14.5) or the asymmetric cost of equation (14.10) describes the problem better, and equation (14.19) is best read as a convenient approximation.

Reliability against customer cost. A second trade-off is hidden in the cost term. Take cohort A delivering $20 \pm 2\,\text{MW}$, highly predictable but expensive in customer comfort because its members are already close to their comfort limits, and cohort B delivering $20 \pm 8\,\text{MW}$, much less predictable but cheap in comfort because its members have ample thermal headroom. Calling A yields a reliable 20 MW at high customer cost; calling B yields the same expected megawatts at low customer cost and four times the spread. The optimizer should not be asked to guess which the operator prefers. Writing the objective as

$$\min_x \ C_{\text{comfort}}(x) + \lambda\,C_{\text{risk}}(x) \tag{14.20}$$

makes the exchange rate explicit: λ is the number of dollars of customer cost the operator will pay to remove one unit of delivery risk. Setting λ is a management decision, and the merit of writing it as a coefficient is that it becomes visible, reviewable and auditable instead of being buried in the choice of which cohort a dispatcher happens to call.

14.4 Multi-period dispatch and model predictive control

Real events last hours, and the fleet's state at one hour is produced by what was done to it in the hours before. This section indexes the dispatch problem by time, which turns it into a control problem, and develops the scheme that solves it in operation.

14.4.1 Multi-period optimization

Suppose the requirement spans 17:00 to 20:00. The decision is no longer a single vector but a sequence $u_{17}, u_{18}, u_{19}, u_{20}$ of control actions, and the hours are coupled through the physical state of the fleet,

$$x_{t+1} = g(x_t, u_t, w_t), \tag{14.21}$$

the state-space recursion of the notation chapter: x_t collects indoor temperatures, thermal headroom, battery states of charge and device availability, u_t is the control applied at t, and w_t is the disturbance, chiefly weather and occupant behavior. Because u_{17} changes x_{18}, the flexibility available at 18:00 is a function of what was done at 17:00, and the four hours cannot be optimized independently. The object to optimize is the trajectory $u_{t:t+H}$ over a horizon of H steps, under an objective that sums the per-hour costs and constraints that apply at every hour.

This is the point at which forecasting and control stop being separate subjects. Equation (14.21) is a dynamic model, the optimizer plans over it, and the forecast supplies the distribution of $w_{t:t+H}$. The scheme that assembles those three pieces into an operating procedure is model predictive control.

14.4.2 The model predictive control loop

Model predictive control runs a fixed cycle. At each decision time it estimates the current state, forecasts the disturbances over the horizon, optimizes the whole future control sequence, implements only the first action of that sequence, observes what happened, advances one step and solves again. Figure 14.5 draws the cycle. The six boxes are the six stages in order, the arrows are the flow of control around the loop, and the return arrow from the last box to the first is labeled with the step that makes the horizon recede. The feature that distinguishes the scheme from ordinary planning is the fourth box: a sequence of four actions is computed and only one is implemented, so that the remaining three are recomputed with better information rather than executed as planned.

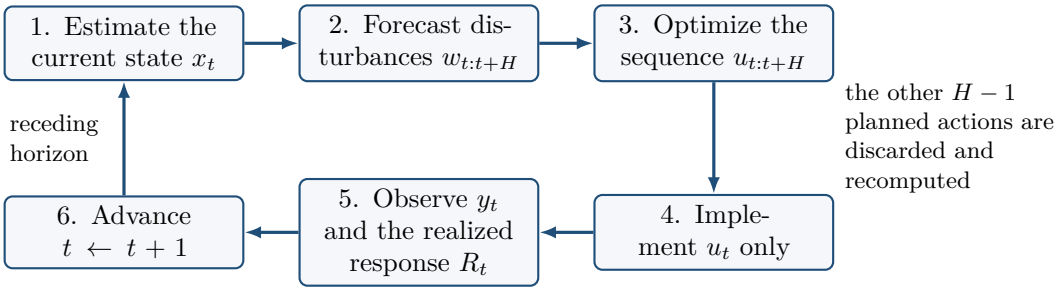


Figure 14.5: The model predictive control loop of Section 14.4.2. Stages 1 to 3 are the planning computation, stage 4 is the only part that reaches the fleet, and stages 5 and 6 close the loop. The forecasting system of this book supplies stage 2 and, through state estimation, stage 1.

Why the scheme suits distributed resources. A plan made at 14:00 for a 17:00 event rests on a weather forecast, an availability estimate and a response model, and all three are revised by 16:00: the afternoon came in hotter than forecast, several thousand devices dropped off the network, and the morning's test dispatch showed the fleet responding more weakly than the model expected. Committing to the 14:00 plan ignores every one of those observations. Re-solving at 16:00 with the revised inputs uses them, and the cost of doing so is one additional optimization, which is cheap. The receding-horizon idea is therefore the operational counterpart of the forecast revision of Chapter 9: a forecast issued closer to the target is better, so a controller should be structured to consume the latest one.

14.4.3 Forecast revision inside the loop

A concrete revision shows what the re-solve changes. At 14:00 the flexibility forecast for the four event hours is

$$\hat{F}_{14:00} = [\,60,\ 55,\ 45,\ 35\,]\,\text{MW} \quad \text{for 17:00 to 20:00,} \tag{14.22}$$

a declining profile, because a fleet held in curtailment drifts toward its comfort limits and because the evening cooling load falls. At 16:00 the fleet is hotter than the 14:00 model assumed, and the revised forecast is

$$\hat{F}_{16:00} = [\,70,\ 60,\ 40,\ 25\,]\,\text{MW}. \tag{14.23}$$

The two profiles disagree in both level and shape. The early hours gain 10 and 5 MW, because hotter buildings have more compressor load to curtail; the late hours lose 5 and 10 MW, because hotter buildings also exhaust their thermal headroom faster once curtailment begins. The total over the block is nearly unchanged, 195 MW against 195 MW, so a planner that looked only at the block total would see no revision at all.

The optimizer's response to equation (14.23) is to save headroom. Under the 14:00 profile the binding hour was 20:00 at 35 MW; under the 16:00 profile the binding hours are 19:00 at 40 MW and 20:00 at 25 MW, so the plan must spend less of the fleet's capacity in the first two hours in order to have anything left for the last two. A re-solve at 16:00 shifts the planned intensity earlier-to-later in exactly that way, and a controller that had committed at 14:00 could not.

14.4.4 Why greedy control fails

The failure mode that multi-period optimization exists to prevent is greed. Suppose the requirement is a flat 40 MW for each of the four hours. A greedy controller satisfies each hour as it arrives, using the cheapest and strongest resources available at that moment. At 17:00 the strongest devices are the ones with the most thermal headroom, and calling them yields 50 MW, comfortably above the requirement. By 19:00 those same devices have reached their comfort limits and dropped out, and the fleet can muster only 20 MW against a 40 MW requirement.

Figure 14.6 contrasts the two strategies. The horizontal axis is the event hour and the vertical axis is delivered response in megawatts; the red horizontal line is the 40 MW requirement in each hour, the blue series is the greedy controller and the green series is a trajectory-aware controller that solves the four hours jointly. Crosses mark the hours in which a strategy falls short. The greedy series starts 10 MW above the requirement and ends 24 MW below it, failing in three of the four hours; the trajectory-aware series delivers exactly 40 MW in each hour and fails in none, by deliberately declining to use capacity at 17:00 that it will need at 19:00. The greedy controller is not worse at any single hour's problem; it is solving the wrong problem, because it treats capacity as if it were free to use now and still available later.

> **Principle 14.2.** *In a multi-period event, flexibility has a time value. Capacity used in the present hour is unavailable in later hours, so the correct objective is the trajectory's total cost, not the sum of each hour's cheapest feasible action. A controller that optimizes hour by hour will exhaust its resource before the event ends.*

14.4.5 The fleet as a virtual battery, and rebound

The time value of flexibility has a familiar analog. A battery obeys

$$\text{SOC}_{t+1} = \text{SOC}_t - \eta\, P_t\, \Delta t, \tag{14.24}$$

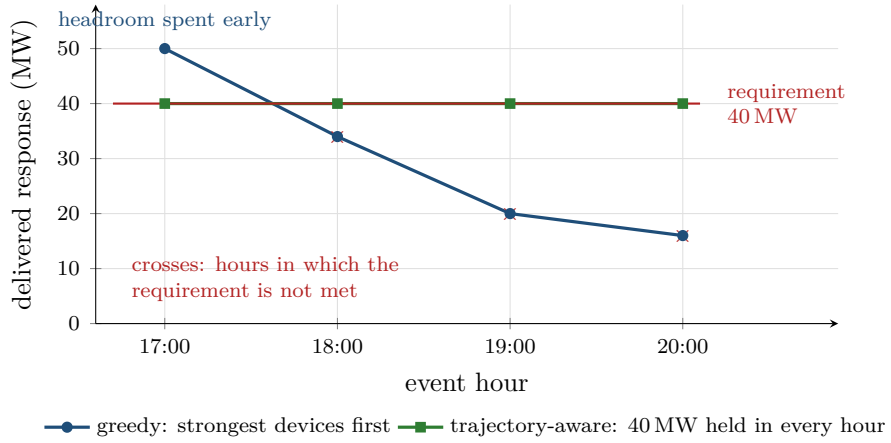


Figure 14.6: Greedy and trajectory-aware dispatch over a four-hour event against a flat 40 MW requirement, following Section 14.4.4. The greedy profile overdelivers at 17:00 and then fails in three hours as thermal headroom is exhausted; the trajectory-aware profile holds 40 MW throughout. Values are illustrative.

so discharging now lowers the state of charge and reduces what can be discharged later. A thermostat fleet obeys a structurally identical relation in its available headroom,

$$F^{\max}_{t+1} = g(F^{\max}_t, u_t, T_t)\,, \tag{14.25}$$

where $F^{\max}_t$ is the maximum response the fleet could deliver at t and T_t is the outdoor temperature. Curtailing now consumes headroom by letting indoor temperatures rise toward their limits, and recovery, the period after an event during which the fleet is allowed to cool back down, restores it. The fleet is therefore a *virtual battery*: a storage-like resource whose state of charge is thermal rather than electrochemical, whose capacity depends on weather, and whose recharge is involuntary.

Rebound belongs in the objective. The recovery that restores headroom is itself a load. A 17:00 control that saves 40 MW may produce an extra 25 MW of consumption at 20:00, as thousands of compressors run simultaneously to return indoor temperatures to setpoint. If 20:00 is an unconstrained hour, the rebound is harmless. If 20:00 is still a peak hour, the net saving over the block is far smaller than the event report suggests, and in the worst case the event has moved the peak rather than reduced it. A multi-period objective can represent this directly by including the rebound as a cost term in the hours after the event,

$$\min_{u_{t:t+H}} \sum_{k=0}^{H} \Big[\, C_{\text{comfort}}(u_{t+k}) + C_{\text{underdelivery}}(u_{t+k}) + C_{\text{rebound}}(u_{t+k}) \,\Big], \tag{14.26}$$

with the horizon H chosen long enough to contain the recovery and not merely the event. Choosing H to end at the last event hour makes the rebound invisible to the optimizer, which is the formulation error that produces peak-shifting dispatches.

14.5 Stochastic and robust formulations

Sections 14.2 and 14.3 treated the response distribution as known. This section treats the two cases that remain: the distribution is known and must be carried through a multi-hour problem, and the distribution is not trustworthy at the conditions in question.

14.5.1 Stochastic optimization over scenarios

When the uncertain quantities are a trajectory rather than a scalar, the expected-cost problem is written over a random element ω collecting the whole uncertain future, temperature path, baseline load, device availability and response variability,

$$\min_{u} \; \mathbb{E}_{\omega}[C(u, \omega)] \,. \tag{14.27}$$

The expectation is almost never available in closed form, so it is approximated by a finite set of scenarios: complete sampled trajectories, each internally consistent in time, drawn from the forecast's joint predictive distribution. Three scenarios for the 17:00 to 20:00 event might be

$$s_1 \text{ (mild)} = [\,55,\; 50,\; 45,\; 40\,] \,\text{MW}, \tag{14.28}$$
$$s_2 \text{ (expected)} = [\,50,\; 45,\; 40,\; 35\,] \,\text{MW}, \tag{14.29}$$
$$s_3 \text{ (difficult)} = [\,40,\; 35,\; 30,\; 25\,] \,\text{MW}. \tag{14.30}$$

The procedure is to evaluate each candidate control sequence against every scenario, aggregate the results with the scenario probabilities, and choose the sequence whose aggregate performance is acceptable. What the scenario representation preserves, and what per-hour quantiles destroy, is the time correlation: within s_3 every hour is low together, because the mechanism that made 17:00 low, a hotter fleet with less headroom, is still acting at 20:00.

Figure 14.7 shows the three scenarios against a flat 40 MW commitment. The horizontal axis is the event hour and the vertical axis is available flexibility in megawatts; the three marked series are equations (14.28) to (14.30), the red horizontal line is the commitment, and each series is annotated at the right with the number of hours in which it meets it. Crosses mark the hours a scenario falls short. Scenario s_1 meets the commitment in all four hours, s_2 in three, failing only at 20:00, and s_3 in one, failing in the last three. Attaching illustrative probabilities of 0.30, 0.50 and 0.20 to the three scenarios gives the per-hour delivery probabilities 1.00, 0.80, 0.80 and 0.30, and the probability of delivering every hour of the event is 0.30, the probability of s_1 alone. The figure makes the asymmetry between the two readings visible: a commitment that looks comfortable in three of four hours is a commitment that fails as a whole seven times in ten.

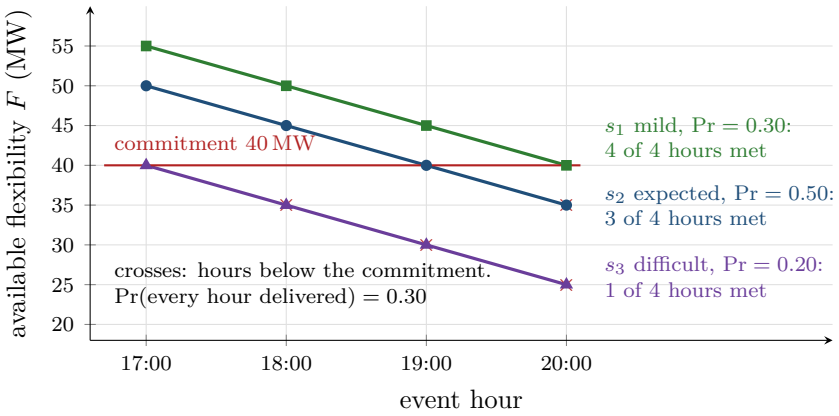


Figure 14.7: Three scenario trajectories, equations (14.28) to (14.30), against a flat 40 MW commitment held across the event. Unlike a per-hour quantile band, each series is a single internally consistent future that the optimizer evaluates as a whole. Scenario probabilities and flexibility values are illustrative.

14.5.2 Per-hour quantiles do not give an event-wide guarantee

The arithmetic behind that last sentence deserves to be separated out, because it is the most common error in multi-hour commitment. Suppose the forecast reports $\mathrm{P10}(F_{17}) = 40\,\mathrm{MW}$ and $\mathrm{P10}(F_{18}) = 40\,\mathrm{MW}$. Each statement, read alone, says that the corresponding hour will deliver at least 40 MW with probability 0.90. Neither statement, nor the pair, says that

$$\Pr(F_{17} \geq 40 \text{ and } F_{18} \geq 40) = 0.90. \tag{14.31}$$

Under independence the joint probability would be $0.90^2 = 0.81$; under perfect correlation it would be 0.90; the truth lies somewhere between, and the marginals contain no information about where. For a four-hour event with four 90 % marginals the independent value is $0.90^4 = 0.656$, so the gap between the per-hour number and the event number is large enough to change the commitment.

The constraint that matches what the operator actually promised is therefore written over the whole event,

$$\Pr(F_t(u) \geq D_t \ \text{ for every } t \in \text{event}) \ \geq \ 1 - \alpha. \tag{14.32}$$

Equation (14.32) is strictly stronger than per-hour constraints at the same level, and it can only be evaluated on an object that carries the joint distribution, which in practice means the scenario set of Section 14.5.1. In Figure 14.7 the shortfalls happen to nest, so the event probability equals the worst per-hour probability; if instead s_2 had failed at 18:00 and s_3 at 20:00, every hour would be met with probability between 0.70 and 0.80 while the event as a whole would still be met only with probability 0.30.

14.5.3 Robust optimization and its relatives

All of the above assumes the forecast distribution is trustworthy. Chapter 13 established the conditions under which it is not: inputs outside the training support, a regime the model has never seen, a fleet whose composition has changed. Where the distribution itself is in doubt, the probability-weighted machinery has no sound basis, and the alternative is to drop probability and require feasibility over a declared set. Robust optimization asks for a dispatch that works for every outcome in an uncertainty set $\mathcal{U}$,

$$\min_{x} \ C(x) \qquad \text{subject to} \qquad \sum_{i} F_i\, x_i \ \geq \ D \quad \text{for all } F \in \mathcal{U}. \tag{14.33}$$

The set $\mathcal{U}$ might be a box, $F_i \in [F_i^{\mathrm{lo}}, F_i^{\mathrm{hi}}]$, or an ellipsoid limiting how far the vector can depart from its forecast. The resulting dispatch is conservative, since it is sized against the worst case in $\mathcal{U}$, and it is safe without requiring the forecast's tail probabilities to be correct, which is precisely the right trade when those probabilities cannot be trusted.

Table 14.2 sets the three formulations side by side. The columns are what each assumes about the uncertainty, what it guarantees, and the conditions under which it is the right choice. Stochastic optimization assumes the predictive distribution is correct and optimizes the average, so it is appropriate for routine operation inside the validated domain, where the calibration checks of Chapter 10 pass. Robust optimization assumes only a range and guarantees feasibility everywhere in it, so it is appropriate for novel conditions and for commitments whose failure is not survivable. Distributionally robust optimization sits between them: it accepts that a distribution has been estimated but allows the true distribution to lie anywhere in a neighborhood of the estimate, and optimizes against the worst distribution in that neighborhood. The conceptual distinction is what matters here; each of the three is a large literature, and an operator needs mainly to know which assumption the production optimizer is making.

Table 14.2: Three ways of handling uncertainty in the dispatch problem.

Formulation	What is assumed	What is guaranteed	When it fits
Stochastic	the predictive distribution $p(F)$ is correct	the best expected cost, or a stated shortfall probability	routine operation inside the validated domain
Robust	only that the outcome lies in a set $\mathcal{U}$	feasibility for every outcome in $\mathcal{U}$	novel conditions, unsurvivable failures
Distributionally robust	the true distribution lies near the estimated one	performance against the worst distribution in that neighborhood	a distribution estimated from limited data

14.5.4 Forecast credibility and the action domain

Which of the three formulations applies is not a fixed choice but a function of the conditions. Chapter 13 defined a validated domain $\mathcal{D}_{\text{val}}$ of feature-and-action pairs on which the model's performance has been measured. Inside it, $(X_t, u) \in \mathcal{D}_{\text{val}}$, the calibrated stochastic treatment of Section 14.5.1 is justified. Outside it, four changes are appropriate at once: widen the safety margins beyond what the model's own σ implies, restrict the actions the optimizer may propose, replace the probabilistic formulation with robust scenarios, and route the resulting dispatch through operator review. The optimization layer must therefore receive a credibility signal from the forecasting layer and act on it, which is a requirement on the interface and not only on the model.

The optimizer exploits modeling error. There is a sharper reason to restrict the action domain, and it is the reason the optimization layer is more dangerous than the forecasting layer. Suppose the fitted response surface $\hat{F}(u)$ contains a small unrealistic bump at a setpoint offset of $u = 3.7°\text{C}$, an artifact of three training events that happened to coincide with unusually hot afternoons. A human reading the surface would not notice it. An optimizer solving $\max_u \hat{F}(u)$ finds it immediately and recommends $u = 3.7°\text{C}$, because its entire job is to locate the maximum of the function it was given. Optimization magnifies modeling error rather than averaging it away, so a response model that is adequate for reporting may be inadequate for control, and the model must be validated over the whole domain of actions the optimizer is allowed to propose, not only over the actions historically taken.

Constraining the optimizer. The direct remedy is to restrict the feasible set to the actions the model has evidence for,

$$u \in \mathcal{U}_{\text{val}} : \quad 0 \le \Delta T \le 2.5°\text{C}, \quad \text{event duration} \le 2\ \text{h}, \tag{14.34}$$

where the bounds are set by where the training record has support rather than by what the devices can physically do. The subscript distinguishes this validated action set from the uncertainty set $\mathcal{U}$ of equation (14.33). If the record contains almost no events with offsets above 2.5°C or durations beyond two hours, then no claim about the response at 4°C for three hours is supported, and the correct action is to exclude those controls until a deliberate experiment supplies the evidence. Widening $\mathcal{U}_{\text{val}}$ is a data-collection project, not a modeling choice.

14.5.5 Physical, comfort and fairness constraints

Alongside the statistical constraints sit the physical ones, which are what makes the dispatch problem a problem rather than an unbounded maximization. The usual set is

$$T_{\text{in}}(t) \le T^{\max}, \quad 0 \le u_t \le u^{\max}, \quad \text{SOC}^{\min} \le \text{SOC}_t \le \text{SOC}^{\max}, \quad \text{SOC}_{\text{dep}} \ge \text{SOC}_{\text{req}}, \tag{14.35}$$

reading in order: indoor temperature must stay below the customer's agreed maximum; control magnitude is bounded by what the device accepts; battery state of charge must stay within its usable band; and an electric vehicle must reach its required state of charge by departure. The division of labor between the two layers is clean. Forecasting predicts what could happen; constraints define what is allowed to happen. A forecast that the fleet could deliver 80 MW by letting indoor temperatures reach 29°C is not an argument for doing so.

Hard and soft constraints. Not every limit is absolute. A *hard* constraint is one the optimizer may never violate, for instance a contractual cap $T_{\text{in}}^{(i)} \le T^{\max}$ on an individual customer's indoor temperature; it enters as an inequality and an infeasible problem is reported as infeasible. A *soft* constraint is one that may be violated at a price, expressed as a penalty in the objective such as

$$C_{\text{comfort}} = \lambda \sum_i \max\!\left(T_{\text{in}}^{(i)} - T^{\text{pref}}, 0\right)^2, \tag{14.36}$$

which permits a small excursion beyond a customer's preferred temperature during a system emergency while making a large or frequent excursion expensive. The quadratic form spreads a necessary violation thinly across many customers rather than concentrating it on a few, which is usually the intended behavior. Deciding which limits are hard and which are soft is a commercial and regulatory decision expressed in the optimization, and getting it wrong in either direction is consequential: a hard constraint where a soft one belongs makes the problem infeasible in exactly the emergencies the program exists for, and a soft constraint where a hard one belongs lets the optimizer sell a customer's comfort for a few megawatts.

Heterogeneity and fairness. Customers are not interchangeable, and the optimizer must be told how they differ. Cohort A might offer high flexibility with a high risk that customers override the event; cohort B moderate flexibility with a low override risk; cohort C low flexibility with almost no operational impact on its members. A good dispatch combines them according to the urgency of the requirement, its duration and the fleet's expected future needs: the override-prone cohort is appropriate for a short, urgent call and inappropriate for a four-hour commitment. Left alone, however, a cost-minimizing optimizer will dispatch whichever customers are cheapest, and because cheapness is a persistent property of a household, it will dispatch the same customers every day. That is a program-design failure with real consequences in attrition and in regulatory exposure. The standard remedies are a hard constraint on dispatch frequency, $n_i \le K$ events per customer over a rolling window, or a cost that rises with use,

$$C_i = C_0 + \lambda\, n_i, \tag{14.37}$$

with n_i the number of times customer i has already been called. The objective of a dispatch optimizer therefore contains more than accuracy and cost; it contains the operator's obligations to its customers, and those obligations have to be written down as constraints or prices to be honored.

14.6 Designing the interface and closing the loop

The last section turns the chapter around and asks what the forecasting system must deliver for an optimizer to use it well, and how the observed outcome returns to the forecast. The answer

reshapes both the training objective and the system architecture.

14.6.1 Forecast value depends on the optimizer

A forecast improvement has value only where the optimizer's decision is sensitive to it. Suppose model A reduces error substantially on mild days, when available flexibility is 20 MW and the requirement is never binding, and model B reduces error slightly in the region near $F \approx 75$ MW, where the chance constraint of equation (14.5) is active and a 2 MW change in the forecast flips the dispatch from one cohort set to another. Model A has the larger reduction in root-mean-square error and close to zero decision value, because the decision was never in doubt in the region it improved. Model B has the smaller metric gain and the larger decision value. The asymmetry is not a curiosity: aggregate error metrics weight every hour equally, and operational value is concentrated in the few hours where the decision is close.

Decision-weighted loss. If the decision value is concentrated, the training objective should be too. The simplest implementation is a weighted loss,

$$L = \sum_t w_t \, (y_t - \hat{y}_t)^2 \,, \tag{14.38}$$

with w_t larger during extreme weather, during events, and when the system is near a capacity limit, and smaller on the mild midnight hours that dominate the record by count and contribute nothing to any decision. The weights are a statement about where accuracy is worth paying for. More sophisticated approaches differentiate through the optimizer itself, training the forecasting model on the realized decision cost rather than on forecast error, so that the model learns whatever representation minimizes regret in the downstream problem. These methods are an active research area; the weighted loss of equation (14.38) captures most of the practical benefit at a small fraction of the engineering cost.

14.6.2 Two architectures

The organizational version of the same point concerns where the two layers meet. In the isolated architecture, a forecasting team produces a number, hands over "we expect 50 MW at 17:00", and an optimization team builds a dispatch from it. Everything the optimizer needs beyond the number, the spread, the correlations, the dependence on the action, the credibility of the model at today's conditions, is discarded at the handover, and the optimizer necessarily reconstructs some crude substitute for it, typically a fixed percentage safety margin applied uniformly.

Figure 14.8 draws the integrated alternative. The boxes along the top row are the forecasting stages, from the information set through the model to the forecast product; the dashed box is the product itself, listing the six items the optimizer consumes; the box at the right is the optimizer with its objective and constraints; the bottom row is what happens physically, the dispatch reaching the fleet and the realized response being measured; and the two upward arrows are the feedback paths, one returning the measurement to the state estimate for the next re-solve and one returning it to model training. What distinguishes this architecture from the isolated one is not the presence of an optimizer but the width of the interface between the boxes and the existence of the return paths.

Table 14.3 lists the contents of the dashed box with the reason each item is there and what the optimizer cannot do without it. The point of the table is that each row corresponds to a formulation developed earlier in this chapter: without the quantiles there is no chance constraint, without the covariance no portfolio variance, without the scenarios no event-wide guarantee, without the rebound forecast no multi-period objective, and without the credibility flag no way to choose between the stochastic and robust treatments.

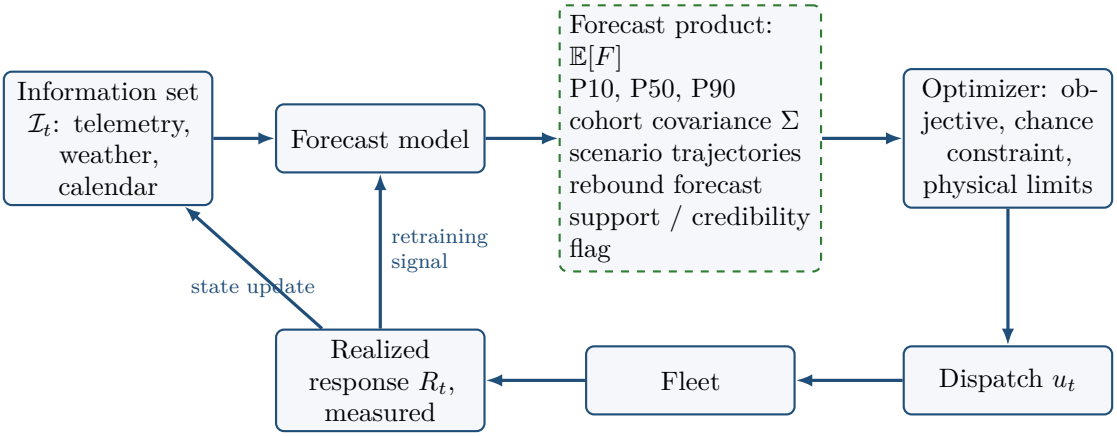


Figure 14.8: The integrated forecast-to-optimizer architecture of Section 14.6.2. The dashed box is the interface: everything the optimizer receives. The two arrows out of the measurement box are the closed loop, one feeding the next re-solve and one feeding model training.

Table 14.3: What the optimizer should receive from the forecasting system, and which formulation each item enables.

Item	What it is	What it enables
Expected response	$\mathbb{E}[F_t(u)]$ per cohort and hour	the mean constraint, equation (14.3)
Quantiles P10, P50, P90	the marginal predictive distribution	the chance constraint and the operational quantile, equations (14.5) and (14.15)
Cohort covariance Σ	how cohort errors move together	portfolio variance, equation (14.16)
Scenario trajectories	joint samples across hours and cohorts	stochastic optimization and the event-wide constraint, equations (14.27) and (14.32)
Rebound forecast	post-event load the dispatch creates	the multi-period objective, equation (14.26)
Customer cost and override risk	comfort cost and expected opt-out per cohort	the comfort-risk trade-off, equation (14.20)
Support and credibility flag	whether (X_t, u) is inside the validated domain	the choice between stochastic and robust, Table 14.2

14.6.3 A risk-aware cohort selection

The interface earns its keep on a concrete problem. Suppose the utility requests 75 MW for 17:00 to 20:00 and three cohorts are available with the forecasts of Table 14.4. The columns are the expected response, the forecast standard deviation and the fifth percentile of each cohort taken alone. A naive dispatch adds the expectations: $A + B = 65$ MW, short of 75, so call part of C as well and the requirement is met on the mean with capacity to spare.

Table 14.4: Three cohorts available for a 75 MW request. Values are illustrative and follow Section 14.6.3.

Cohort	$\mathbb{E}[R_i]$ (MW)	σ_i (MW)	5th pct (MW)	Character
A	35	3	30.1	large and predictable
B	30	10	13.6	large and unreliable; weather-correlated with A
C	25	4	18.4	moderate and predictable; different region

Example 14.2 (The mean is not the deliverable)**.** Take A and C as independent of each other, and $\mathrm{Corr}(R_A, R_B) = 0.6$ through shared weather. Calling all three gives $\mu = 90$ MW and

$$\sigma^2 = 9 + 100 + 16 + 2(0.6)(3)(10) = 161, \qquad \sigma = 12.7\,\text{MW}, \tag{14.39}$$

so by equation (14.7) the fifth percentile is $90 - 1.645 \times 12.7 = 69.1$ MW. The dispatch has 15 MW of apparent surplus on the mean and still fails the 95 % chance constraint at 75 MW. Calling only A and C gives $\mu = 60$ MW and $\sigma = \sqrt{9 + 16} = 5$ MW, a fifth percentile of 51.8 MW. Cohort B therefore contributes $69.1 - 51.8 = 17.3$ MW of firm capacity, not the 30 MW its forecast advertises, and the remaining gap must be closed with reserve: 5.9 MW behind the three-cohort dispatch, or 23.2 MW behind A and C alone. Note also that adding the cohorts' individual fifth percentiles gives $30.1 + 13.6 + 18.4 = 62.1$ MW, which is neither the mean nor the portfolio's fifth percentile; the quantile of a sum is not the sum of the quantiles.

Preserving future flexibility. The selection problem of Example 14.2 acquires another dimension over a multi-hour event. Suppose cohort A can deliver 40 MW but recovers slowly, needing several hours of uncurtailed operation before it can be called again, while cohort B delivers 35 MW and recovers quickly. For a single-hour call A is the better choice. For a four-hour event, using A first may be a mistake: it is spent after one hour, and the remaining three hours must be covered by whatever is left. The quantity the optimizer needs is not $\hat{F}_i$ but the conditional future flexibility $F_i(t + h \mid u_{t:t+h})$, the response cohort i can still deliver at a later hour given everything that will have been done to it in between. That is a forecast of a controlled trajectory, and it is the output that the virtual-battery model of equation (14.25) exists to produce.

14.6.4 Closing the loop

Dispatch is not the end of the cycle. Once the event begins the realized response R_t is measured and compared with what was forecast, and a discrepancy is information about the rest of the event. If the first interval delivers less than forecast, four responses are available and all four are improvements on doing nothing: update the state estimate, since the fleet is evidently in a different thermal state than assumed; revise the forecasts for the remaining hours, since the same discrepancy will likely persist; call additional cohorts to cover the gap; and adjust the intensity of the controls already issued. Each of those is a re-solve of the model predictive control problem of Figure 14.5 with one more observation in hand.

Open loop and closed loop. The contrast is stark. An open-loop controller computes $u_{17:00}$ through $u_{20:00}$ at 14:00 and never revises them. A closed-loop controller observes y_t and updates u_{t+1}. The practical consequence is that forecast uncertainty is far less dangerous in the second case, because an error made at 14:00 is detected at 17:05 and corrected at 17:15 rather than compounding for three hours. Fast feedback substitutes for forecast accuracy, and the two are to some extent interchangeable: a controller that reacts in fifteen minutes can tolerate a forecast that a controller committed four hours in advance cannot.

The value of frequent telemetry. The substitution is worth quantifying because it justifies metering investment. At 14:00 the forecast for the 19:00 hour might carry $\sigma = 10\,\mathrm{MW}$, most of it weather and state uncertainty. At 17:05, five minutes into the event, the realized response of the first interval is observed, and it is a direct measurement of the quantity being forecast under the actual control that was applied; conditioning on it can plausibly cut the remaining hours' uncertainty to $\sigma = 4\,\mathrm{MW}$. By Figure 14.2 that reduction is worth 9.9 MW of committable capacity for the rest of the event. The telemetry that makes the 17:05 observation available at 17:06 therefore has a value that can be written in megawatts, which is a more persuasive argument for interval metering than any appeal to data quality.

14.6.5 Three horizons

A system of this kind runs on three different time constants, and conflating them is a common source of confusion in design discussions. Table 14.5 separates them. The forecast horizon is how far ahead the forecasting model predicts, typically 24 hours, because day-ahead market participation and staffing decisions require it. The optimization horizon is how far ahead the optimizer plans, typically 4 hours, chosen to contain the event and its recovery as equation (14.26) requires, and no longer, because planning beyond the point where the forecast carries information adds computation without adding value. The control interval is how often an action is actually issued, typically 15 minutes, set by telemetry cadence, communication latency and how fast the physical devices can be expected to respond. The three are independent design parameters: a 24-hour forecast, a 4-hour plan and a 15-minute action are mutually consistent, and a system that uses one number for all three is either planning blind or discarding information.

Table 14.5: The three time constants of a forecast-driven dispatch system.

Concept	Typical value	What it is	What sets it
Forecast horizon	24 h	how far ahead $p(F_{t+h} \mid \mathcal{I}_t, u)$ is produced	market gate closure, staffing, procurement
Optimization horizon	4 h	how far ahead the control trajectory is planned	event duration plus recovery, and where the forecast still informs
Control interval	15 min	how often an action is issued and the problem re-solved	telemetry cadence, communication latency, device response time

14.6.6 Conceptual questions

The chapter's arguments recur as conceptual questions, and the five the lesson raised are worth stating with the shape of a strong answer.

Conceptual questions

"How should forecast uncertainty enter the optimizer?" A strong answer begins by rejecting the wrong form: not as a deterministic capacity number, however conservative, because a single number cannot express a reliability requirement. It then names the three legitimate forms, each developed above: optimize expected cost across scenarios, as in equations (14.11) and (14.27); use a conservative quantile selected by the cost asymmetry, as in equation (14.15); or impose a chance constraint, as in equation (14.5). The answer closes on the part candidates most often omit, that the correlations across cohorts and across hours must be preserved, because the chance constraint is a statement about a sum and a multi-hour commitment is a statement about a trajectory.

Conceptual questions

"If Model A has lower RMSE, should the optimizer always use it?" No. The criterion is which forecast produces better decisions, not which has the smaller error. Model B may have lower bias, better calibration in the downside tail, or better accuracy specifically near the conditions at which the dispatch decision changes, any one of which can make it the better model for the optimizer while losing on root-mean-square error, as Table 14.1 illustrates. A strong answer says how to settle the question rather than asserting an answer: replay history through the optimizer and compare downstream decision metrics, delivery rate, underdelivery depth, total cost and unused capacity.

Conceptual questions

"Why isn't expected flexibility enough?" Because two fleets with the same expected response can have very different probabilities of underdelivery, which is the example of equation (14.4): $\mathcal{N}(40, 2^2)$ and $\mathcal{N}(40, 15^2)$ contribute identically to an expectation constraint and behave completely differently in operation. The quantity a commitment depends on is the lower tail, and a strong answer gives the margin form $\mu - z_{1-\alpha}\sigma \geq D$ of equation (14.7) to show that the reserve is a function of the spread, not of the mean.

Conceptual questions

"What if the optimizer proposes controls outside the training data?" That is extrapolation in the action space, and it is more dangerous than extrapolation in the features, because the optimizer is actively searching for the extreme of a function it does not know to be valid, as the 3.7°C artifact of Section 14.5.4 shows. The answer is to restrict the optimizer to a validated action domain, as in equation (14.34), or to attach deliberately conservative uncertainty and hard physical constraints to actions outside it, and then to run controlled experiments to extend the domain rather than letting the optimizer explore it in production. A strong answer connects this to the distribution-shift machinery of Chapter 13.

Conceptual questions

"How do rebound effects change optimization?" They make the problem multi-period. A strong reduction now can raise load or reduce available flexibility later, so an objective that stops at the end of the event optimizes a quantity the operator does not care about. The answer is to forecast the whole controlled trajectory, including the recovery, and to carry the rebound as a cost term and the headroom as a state, as in equations (14.25) and (14.26). The concrete illustration is the 17:00 control that saves 40 MW and creates 25 MW of rebound at 20:00: whether that is a good trade depends entirely on whether 20:00 is constrained, and only a multi-period formulation can ask the question.

14.6.7 The course in one equation

The progression of this book can be written as a single chain. It began with a function returning a number, became a distribution, became a distribution conditional on a planned action, and ends as a decision:

$$\begin{aligned}\underbrace{\hat{y}_{t+h} = f(\mathcal{I}_t)}_{\text{point forecast}} \;\longrightarrow\; \underbrace{p(y_{t+h} \mid \mathcal{I}_t)}_{\text{probabilistic}} \;\longrightarrow\; \underbrace{p(y_{t+h} \mid \mathcal{I}_t, u)}_{\text{action-conditional}} \\ \longrightarrow\; \underbrace{u^* = \arg\min_u \mathbb{E}[C(Y,u) \mid \mathcal{I}_t] \;\; \text{subject to constraints}}_{\text{decision}}\,.\end{aligned} \tag{14.40}$$

Each arrow is a chapter's worth of work, and the last one is this chapter. For the aggregator the final term is concrete:

$$u^* = \arg\min_u \; \mathbb{E}[C_{\text{comfort}} + C_{\text{underdelivery}} + C_{\text{rebound}}] \tag{14.41}$$

subject to $\Pr(F_t(u) \geq D_t \text{ for every event hour } t) \geq 1 - \alpha$ and the physical and customer constraints of equations (14.35) and (14.37). That is risk-aware virtual power plant dispatch, and every symbol in it has been built somewhere in this book.

Why the forecast still matters. The optimizer in equation (14.41) is only as good as the distribution it integrates against. A biased forecast makes the optimizer systematically overcommit or underuse the fleet; a miscalibrated one makes the chance constraint a fiction, satisfied on paper at 95 % and in reality at 70 %; one that ignores correlation understates the portfolio variance and therefore the reserve; and one that is invalid at today's conditions sends a perfect solver to a confidently wrong answer. The solver cannot detect any of these, because it has no access to the truth and no reason to doubt its inputs. Forecasting is therefore not a service that feeds the control system; it is part of the control system, and its failures are control failures.

14.7 What to retain

Definition 14.1 (Forecast-driven dispatch)**.** A forecast supplies $p(F \mid X_t, u)$; an optimizer turns it into an action u^* using the expected response, the downside tail, the correlations across cohorts and hours, the customer cost, the physical constraints, the rebound and the value of flexibility kept for later. The deterministic form $\mathbb{E}[F] \geq D$ is not sufficient; the risk-aware forms are a chance constraint $\Pr(F \geq D) \geq 1 - \alpha$ or an expected cost $\min_u \mathbb{E}[C(u, F)]$. Because a multi-hour event is dynamic, with $x_{t+1} = g(x_t, u_t, w_t)$, the operating form is a receding-horizon re-solve, and a forecast is valuable only insofar as it improves the decision that follows it.

Table 14.6 collects the chapter's objects with the subsection that defines each.

14.8 Exercises

Exercise 14.1. A dispatch has $\mathbb{E}[R_{\text{total}}] = 80\,\text{MW}$ and $\sigma = 7\,\text{MW}$, against a requirement $D = 75\,\text{MW}$. Compute $\Pr(R_{\text{total}} \geq 75)$ under a normal model, and state whether the dispatch satisfies a 95 % chance constraint. Then use equation (14.7) to find the expected response that would be required at this σ, and the value of σ at which the original 80 MW dispatch would just become feasible. Finally, state how many megawatts of committable capacity a reduction from $\sigma = 7\,\text{MW}$ to $\sigma = 5\,\text{MW}$ releases.

Table 14.6: The objects of forecast-driven dispatch.

Object	Definition	Section
$\min_x C(x)$ s.t. $\sum_i R_i(x_i) \geq D$	the deterministic dispatch problem	14.1.2
$\sum_i \mathbb{E}[R_i] x_i \geq D$	the expectation constraint; blind to spread	14.1.4
$\Pr(R_{\text{total}} \geq D) \geq 1 - \alpha$	the chance constraint	14.2.1
$\mu - z_{1-\alpha}\sigma \geq D$	its deterministic equivalent; the reserve is $z_{1-\alpha}\sigma$	14.2.3
decision-focused validation	replaying history through the optimizer and scoring the decisions	14.2.6
$C(e)$ asymmetric	\$1000 per MW short against \$100 per MW surplus	14.3.1
$\min_x \mathbb{E}[C(x, R)]$	the expected-cost objective	14.3.2
$F(c) = \lambda_{\text{over}}/(\lambda_{\text{under}} + \lambda_{\text{over}})$	the quantile an asymmetric cost selects	14.3.3
$\text{Var}(R_{\text{total}}) = x^{\mathsf{T}}\Sigma x$	portfolio variance; the covariance decides reliability	14.3.4
$\max_x[\mu(x) - \lambda\sigma^2(x)]$	the mean-variance objective	14.3.5
$x_{t+1} = g(x_t, u_t, w_t)$	the state recursion that couples the hours	14.4.1
model predictive control	estimate, forecast, optimize, implement one action, observe, repeat	14.4.2
receding horizon	the re-solve over a window that slides forward	14.4.2
$F_{t+1}^{\max} = g(F_t^{\max}, u_t, T_t)$	the fleet as a virtual battery; flexibility has a time value	14.4.5
scenario set	joint trajectories that preserve time correlation	14.5.1
event-wide chance constraint	$\Pr(F_t \geq D_t\ \forall t) \geq 1 - \alpha$; stronger than per-hour	14.5.2
$F \in \mathcal{U}$	the robust uncertainty set	14.5.3
$\mathcal{U}_{\text{val}}$	the validated action domain the optimizer may search	14.5.4
hard and soft constraints	never violated against violated at a priced penalty	14.5.5
$C_i = C_0 + \lambda n_i$	the fairness cost of repeated dispatch	14.5.5
$L = \sum_t w_t(y_t - \hat{y}_t)^2$	the decision-weighted training loss	14.6.1
the forecast product	quantiles, covariance, scenarios, rebound, credibility flag	14.6.2
three horizons	forecast 24 h, optimization 4 h, control 15 min	14.6.5

Exercise 14.2. Using Example 14.1, find the failure probability at which option A's expected cost equals option B's \$1150. Then hold both failure probabilities fixed at 20 % and 5 % and find the penalty at which the two options tie, and state which option is preferred for penalties above and below that value. Comment on what the answer implies about setting a reliability target without knowing the penalty.

Exercise 14.3. Two cohorts each have $\sigma = 5\,\mathrm{MW}$. Compute σ_{total} at $\rho = 0$, 0.5 and 1 using the closed form of Figure 14.4, and convert each into the reserve $1.645\sigma_{\text{total}}$ required by a 95 % chance constraint. Then take three cohorts of $\sigma = 5\,\mathrm{MW}$ with every pairwise correlation equal to 0.3, compute $\mathrm{Var}(R_{\text{total}}) = x^{\mathsf{T}}\Sigma x$ for $x = (1, 1, 1)$ from equation (14.16), and compare the result with the perfectly independent and perfectly correlated cases.

Exercise 14.4. The 14:00 forecast of equation (14.22) is revised to the 16:00 forecast of equation (14.23), and the requirement is 40 MW in each of the four hours. Show that a greedy controller spending available flexibility hour by hour fails under the revised forecast, and construct a trajectory that meets all four hours or demonstrate that none exists. Then add a rebound of +25 MW at 20:00 caused by a strong 17:00 control, as in Section 14.4.5, and state how the feasible set changes if 20:00 is itself a constrained hour.

Chapter 15

Production forecasting systems: pipelines, contracts, monitoring and fallbacks

15.1 From model to system

Every chapter so far has worked on a model. Features were built, a learner was fitted, a validation protocol was chosen, a predictive distribution was calibrated, and the output was handed to an optimizer. All of that assumed that the inputs exist, that they arrive on time, that they mean what they meant during training, and that the software that computes the forecast answers when it is called. In an operating aggregator none of those four assumptions holds continuously. The question of this chapter is

how a forecast is produced continuously, safely and reproducibly when the data are late, missing, changing and occasionally wrong.

The answer is not a better model. It is a system whose components are the data pipeline, the feature layer, the serving layer, the monitoring layer, the fallback logic and the retraining loop, with the model as one component among them.

Jargon: forecast contract, vintage, feature store, training-serving skew, lineage, champion-challenger, shadow mode, canary, graceful degradation

A *forecast contract* is the full specification of what the forecasting system delivers on every cycle: which horizons, which quantiles, and which metadata accompany them. A *vintage* of a forecast (weather or energy) is the version of it issued at a particular time for a particular target; a target time has as many vintages as there were issue times before it. A *feature store* is the single place where a feature's definition lives, so that the same code computes it for training and for serving. *Training-serving skew* is any difference between the feature values a model saw while being fitted and the values it receives in production. *Lineage* is the stored record of everything that produced one forecast: input versions, feature version, model version and output. *Champion-challenger* is the practice of keeping the deployed model (the champion) and a candidate (the challenger) side by side and promoting the challenger only on demonstrated improvement. *Shadow mode* runs the challenger on live data while discarding its output; a *canary* deployment routes a small fraction of real traffic to the challenger before the rest. *Graceful degradation* is the property that when an input or a component fails, the system produces a worse but usable and honestly labeled forecast rather than no forecast.

Key idea

A production forecasting system is judged by the quality of the decisions it supports when conditions are not ideal. Reproducibility, observability, graceful degradation, validated fallbacks and closed-loop monitoring are therefore part of the forecast itself, not operational decoration around it.

This section states the gap between a good model and a reliable system, then fixes the two objects that the rest of the chapter builds on: the contract the system delivers, and the issue time that makes a forecast a well-defined quantity.

15.1.1 The model is one component

Consider a flexibility forecaster that has been built with everything the preceding chapters supply. It is a gradient-boosted model over the features of Chapter 4, validated by rolling origin as in Chapter 9, with calibrated quantiles as in Chapter 10, and its held-out root-mean-square error is RMSE = 5 MW on a fleet whose flexibility runs between 20 and 70 MW. By every measure in this book so far it is a good model.

At 4:55 PM, five minutes before a dispatch event, the following is the state of the system:

- the weather feed is unavailable, so no outdoor temperature forecast for the event window exists;
- 20 % of the thermostats have stopped reporting, so the fleet's state is known for four devices in five;
- the load data are 12 minutes late, so the most recent lag features cannot be computed from observed values;
- the feature job failed, so the feature vector for the current issue time was never written;
- the model service times out, so even a stale feature vector would not produce a number.

Under these conditions the held-out error of 5 MW is irrelevant. It describes the accuracy of a function evaluated on inputs that do not exist. The operational quality of a forecasting system is the conjunction of three things,

$$\text{production quality} = \underbrace{\text{model accuracy}}_{\text{Chapters 3–10}} + \text{data reliability} \;+\; \text{system reliability}, \tag{15.1}$$

and the second and third terms are the subject of this chapter. The list above is returned to in Section 15.6.7, where the same situation is handled by a system that was designed for it.

Principle 15.1. *Offline accuracy bounds how well a forecasting system can perform. Data reliability and system reliability determine how often it performs at all.*

Figure 15.1 draws the full chain the forecast passes through. The boxes are processing stages, read left to right along the top row and then left to right along the bottom row, with the arrowheads showing the direction data move; the three boxes above the top row are the sources, the devices on the left and the two external inputs, the weather feed and the calendar, on the right. The stages are telemetry ingestion, data validation and cleaning, aggregation to fleet and cohort statistics, feature computation, the forecast model, the predictive distribution, the forecast service that other systems call, the optimizer of Chapter 14, the dispatch command, and the observed response that eventually arrives back as telemetry. The figure's point is proportion: the model is one box in ten, and each of the other nine can fail on its own, with the forecast failing as a consequence.

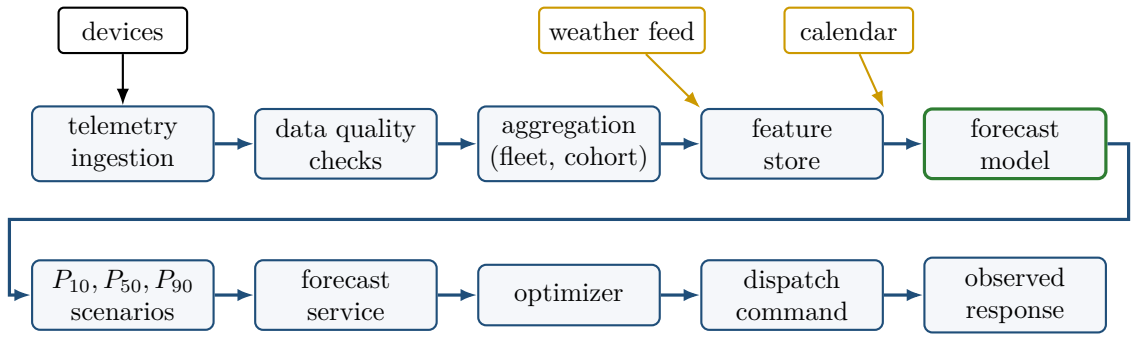


Figure 15.1: The end-to-end production chain as a stage diagram. Top row, left to right: ingestion through the model. The blue path then carries the model output to the bottom row, read left to right: the predictive distribution through the observed response. Gold boxes are external inputs; the green box is the model, which is one stage in ten.

15.1.2 The forecast contract

What the system owes its consumers has to be written down, because an optimizer, a trading desk and a settlement process each depend on a different part of it. A concrete contract for a thermostat aggregator is the following. Every 15 minutes, for every fleet, produce flexibility forecasts at the horizons

$$h \in \{15\ \text{min},\ 30\ \text{min},\ 45\ \text{min},\ \ldots,\ 6\ \text{h}\}, \tag{15.2}$$

and for each horizon deliver a central forecast $\hat{F}_h$ together with the quantiles $P_{10,h}$, $P_{50,h}$ and $P_{90,h}$ of Chapter 10, plus the metadata of Table 15.1. A 6-hour horizon at 15-minute resolution is 24 horizons, so one cycle emits 24 records of four numbers each, and 96 cycles a day emit 2304 records per fleet.

Table 15.1: The forecast contract: what one record of the system's output contains. The first four rows are the forecast, the last five are the metadata without which the forecast cannot be reproduced or trusted.

Field	Meaning
$\hat{F}_h$	central flexibility forecast for horizon h, in MW
$P_{10,h}$	10th percentile: the conservative end used for firm commitments
$P_{50,h}$	median
$P_{90,h}$	90th percentile: the optimistic end
`issue_time`	the time at which the forecast was made, which fixes its information set
`model_version`	the identifier of the fitted model that produced it, for example `7.3`
`data_completeness`	fraction of expected telemetry present at issue time, for example 0.94
`weather_version`	which weather run the inputs came from, for example the `13:00` run
`ood_flag`	whether the input vector lies outside the training distribution, in the sense of Chapter 13

The five metadata fields of Table 15.1 are what turns a number into a reproducible artifact. Without `issue_time` the forecast cannot be scored, because the error of a 15-minute-ahead forecast and a 6-hour-ahead forecast for the same target are not comparable. Without `model_version` a change in behavior cannot be attributed to a deployment. Without `data_completeness` the consumer cannot tell a forecast made from the whole fleet from one made from half of it. Without `weather_version` the forecast cannot be reconstructed, for the reason developed in Section 15.2.7. And without `ood_flag` the optimizer has no signal that the input conditions are ones under which the model has never been validated. A forecasting system delivers records of this shape, not numbers.

15.1.3 Issue time must be explicit

The statement "flexibility at 17:00 is 50 MW" is not a forecast. It is missing the one piece of information that determines what the number means, which is when it was produced. The complete object is

$$\hat{F}_{17:00|14:00} = \mathbb{E}\big[F_{17:00} \mid \mathcal{I}_{14:00}\big], \tag{15.3}$$

read as the forecast of the 17:00 flexibility as issued at 14:00, where $\mathcal{I}_{14:00}$ is the information set of Chapter 1: everything that genuinely existed at 14:00. A second forecast of the same target issued at 16:45 is

$$\hat{F}_{17:00|16:45} = \mathbb{E}\big[F_{17:00} \mid \mathcal{I}_{16:45}\big], \tag{15.4}$$

and because $\mathcal{I}_{14:00} \subset \mathcal{I}_{16:45}$, the second conditions on strictly more: three more hours of telemetry, two more weather runs, the realized afternoon temperature, the current duty cycle of the fleet.

These are two different products, not two attempts at the same number. The 14:00 vintage is the one a day-ahead or hours-ahead market commitment was made on; the 16:45 vintage is the one the real-time dispatch decision uses. Comparing the accuracy of one against the other measures only the value of three hours of information. Evaluating the 14:00 commitment against the 16:45 forecast, rather than against the observed outcome, measures nothing at all. The practice that follows from this is to store every vintage separately and never overwrite one with a later one, which is taken up in Section 15.6.3.

15.2 The data pipeline

This section works through the left half of Figure 15.1: what arrives, what can be wrong with it, what to do when part of it is absent, and why the weather input needs a version number.

15.2.1 Stages and signals

The pipeline that stands behind a fleet flexibility forecast has five stages in series. Devices emit raw telemetry; the telemetry is validated and cleaned; the cleaned records are aggregated into fleet and cohort statistics; the statistics and the external inputs are turned into features; the features are passed to the model. Each stage has its own failure modes, and a failure at any stage propagates to the forecast.

The signals the pipeline carries for a thermostat fleet are

- device load $P_{i,t}$, the power the connected equipment is drawing;
- the setpoint T_{set} currently programmed;
- the indoor temperature T_{in};
- the HVAC state, which is the compressor's on-off status and hence the duty cycle D;

- connectivity, which is whether the device is reporting at all;
- event history, which is when the device was last dispatched and whether it was overridden;
- weather forecasts for the horizons in equation (15.2);
- calendar information, which is the day type, holiday status and program rules.

The first six come from the devices over the network, the seventh from an external provider, the eighth from internal configuration. They have independent failure modes: the weather provider can be unreachable while telemetry is perfect, a firmware release can change the units of the load field while connectivity is unaffected, and a calendar table can go stale in a way that no device-level check would reveal.

15.2.2 Data quality checks

The model must not be the first place where bad data are detected. A model is a function; handed an impossible number it returns a number, and the number looks like a forecast. The checks that stand between the telemetry and the model fall into the categories of Table 15.2, which lists each check, what it rejects, and a concrete example from a thermostat fleet. The table is organized from the cheapest and most local check at the top to the most contextual at the bottom.

Table 15.2: Data quality checks between telemetry and the model. The ranges in the third column are illustrative values for a residential thermostat fleet.

Check	What it rejects	Example
Physical range	values the quantity cannot take at all	$-40°\mathrm{C} < T < 60°\mathrm{C}$; duty cycle $0 \le D \le 1$; SOC $\in [0, 1]$
Operating range	values physically possible but outside normal operation	indoor temperature is typically 18–30°C; a report of 215°C
Timestamp sanity	future timestamps, impossible gaps, clock drift	a reading stamped two hours ahead of the server clock
Staleness	valid values that are old	$T_{\mathrm{in}} = 23°\mathrm{C}$ measured three hours ago
Reporting cadence	devices reporting too often or too rarely	a thermostat on a five-minute schedule sending one report an hour
Unit consistency	a silent change of units	a cohort's reported load falling by a factor of 1000 after a firmware release
Completeness	fewer reporting devices than expected	85 000 of 100 000 enrolled devices reporting

Range checks in detail. The first two rows of Table 15.2 deserve separating, because they fail for different reasons and need different responses. A physical range check rejects what cannot exist: a state of charge of SOC = 1.35 is impossible when SOC is defined on $[0, 1]$, so the record is corrupt and must be dropped or repaired, with no interpretation attached. An operating range check rejects what exists but is implausible: an indoor temperature of 215°C is inside no residence, and the value is almost certainly a sensor fault, a unit confusion or a packet corruption. The distinction matters because the second kind of check has a threshold that can be wrong. A genuine 45°C indoor temperature in an unoccupied house during a heat wave is outside the typical 18–30°C band and is real, and rejecting it silently would hide exactly the condition under which the fleet's flexibility is most valuable. The practice is therefore to reject on physical impossibility, flag and quarantine on implausibility, and count both, so that a rising count of flagged records becomes a monitoring signal in its own right (Section 15.4.2).

15.2.3 Staleness and measurement age

A value can pass every range check and still be useless. Suppose device i reports $T_{\text{in}} = 23°\text{C}$, which is plausible, physically possible, correctly stamped and internally consistent, but the reading is three hours old. The house may now be at 27°C. Nothing about the value is wrong; what is wrong is the implicit assumption that a field in the current state record describes the current state.

The pipeline therefore has to carry an explicit age alongside every measurement,

$$\Delta t_i(t) = t - t_i^{\text{last}}, \tag{15.5}$$

where t_i^{last} is the timestamp of device i's most recent report and t is the issue time. Three uses follow. First, Δt_i is a filter: beyond some threshold the reading is discarded rather than used, and the device is counted as not reporting. Second, Δt_i is a feature: a model given both T_{in} and its age can learn to lean on the measurement when it is fresh and on the fleet aggregate when it is stale, which is more accurate than either a hard cutoff or blind trust. Third, the distribution of Δt_i across the fleet is a health metric: a shift in its median is an ingestion problem that will reach the forecast before any accuracy metric notices, for the reason given in Section 15.4.4.

15.2.4 Missingness and what to do about it

At a given issue time a fleet of 100 000 enrolled thermostats may have 85 000 reporting. The remaining 15 000 are enrolled, probably operating, and invisible. The forecast is required anyway, and there is no single correct treatment. Five designs are defensible, each with a different failure mode:

1. **Forecast only the responders.** Produce flexibility for the 85 000 devices whose state is known and commit nothing for the rest. This is the most conservative choice and it systematically understates the fleet: the missing devices will in general deliver something, so capacity is left on the table every cycle.

2. **Impute the missing states.** Fill each absent T_{in}, D and P_i with a cohort-conditional estimate and run the device-level model on the completed fleet. This recovers the capacity and inherits whatever bias the imputation has, which is the subject of Section 15.2.5.

3. **Extrapolate the fleet response.** Compute the flexibility of the observed 85 000 and scale it by $100\,000/85\,000 = 1.176$. This assumes the missing devices behave like the observed ones, which is the assumption most likely to fail exactly when it matters.

4. **Use historical availability probabilities.** For each device or cohort, estimate from history the probability of being available conditional on the hour, the weather and the recent dispatch, and weight the contribution accordingly, which replaces a point assumption with a distribution over the unobserved part of the fleet.

5. **Fall back to a cohort-level model.** Abandon device-level detail for the cohorts whose coverage has dropped and forecast those cohorts from aggregate features, which is the degraded mode of Section 15.5.2.

The engineering requirement is not to pick the right one in the abstract. It is to pick one deliberately, to state which one is in use in the forecast's metadata, and to know the error characteristics of that choice, because a forecast produced under option 3 and a forecast produced under option 1 are different estimators of the same quantity.

15.2.5 Missingness is a feature

The imputation options above all rest on an assumption that is false in a thermostat fleet. Connectivity is not missing at random. Residential internet service degrades in storms; devices lose power in heat-driven outages; cellular backhaul congests during regional events. In symbols,

$$\Pr(\text{device } i \text{ missing at } t \mid T_t) \;\neq\; \Pr(\text{device } i \text{ missing at } t), \tag{15.6}$$

and the dependence runs through exactly the variable that drives the forecast. The consequence is that imputing missing states from the observed ones biases the result, and biases it most strongly in extreme weather, which is when the fleet is dispatched. Filling the 15 000 unknown homes with the average of the 85 000 known ones assumes the two groups are exchangeable, and equation (15.6) says they are not.

The constructive response is to make the missingness itself an input. Define the observed fraction and its complement,

$$M_t = \frac{N_{\text{enrolled}} - N_{\text{avail}}(t)}{N_{\text{enrolled}}}, \tag{15.7}$$

and include M_t, and its cohort-level counterparts, in the feature vector X_t. A model with M_t available can learn that a high missing fraction co-occurs with storm conditions and with a particular response pattern, so the information in "part of the fleet is dark" is used rather than papered over. This is the production instance of a general rule from Chapter 4: an indicator of why a value is absent often carries more signal than any value substituted for it.

15.2.6 Aggregation adds robustness

Aggregation is usually introduced as a statistical convenience, for the variance reasons of Chapter 11. In a production pipeline it is also a robustness mechanism. A device-level forecast needs every device's record to be present and correct; a fleet-level forecast needs a summary to be approximately right. Summaries that behave well under partial data loss are

$$\begin{aligned} X_t^{\text{fleet}} = [\; & N_{\text{avail}}(t),\ \text{fraction cooling},\ \text{median}(D_{i,t}), \\ & P_{10}(T_{\text{in}}),\ P_{50}(T_{\text{in}}),\ P_{90}(T_{\text{in}}),\ M_t,\ \dots\;], \end{aligned} \tag{15.8}$$

which are a count, a proportion, a median and three quantiles of the indoor temperature distribution. Losing a few hundred devices out of 100 000 moves a median or a quantile of a large sample by a negligible amount, whereas it removes a few hundred terms from a device-level sum outright. Quantiles are chosen over means for the second-order reason given in Chapter 11: they survive both sampling loss and outliers, so one device reporting 215°C before the range check catches it shifts $P_{90}(T_{\text{in}})$ not at all, while it would shift a mean.

15.2.7 Weather vintages and weather versioning

The weather input is the one external signal that both dominates the forecast and changes continuously. A single target time has as many weather predictions as there have been weather runs before it. For a 17:00 target,

$$\hat{T}_{17:00|12:00} = 35^\circ\text{C}, \qquad \hat{T}_{17:00|14:00} = 37^\circ\text{C}, \qquad \hat{T}_{17:00|16:00} = 38^\circ\text{C}, \tag{15.9}$$

three different numbers for one temperature, each correct as of its own issue time. These are the *vintages* of the weather forecast.

Figure 15.2 draws them. The horizontal line is clock time, ticked hourly from 11:00 to 18:00, with the dashed vertical line marking the single target time 17:00. The three horizontal bars above the axis are the three vintages: each starts at its issue time, where a dotted line drops to

the axis, and ends with an arrowhead at the target, and each is labeled with the temperature it predicted. Reading upward, the bars get shorter and the predicted temperature rises from 35 to 38°C. The operational reading is the one on the right-hand side of the figure: the energy forecast must be reissued whenever a new weather vintage arrives, because a 3°C revision in the driver is a large revision in fleet flexibility, and the pipeline must retain all three bars rather than only the last.

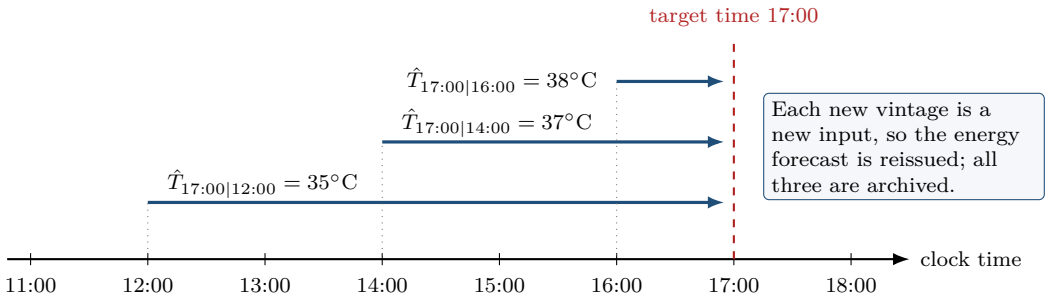


Figure 15.2: Three vintages of the weather forecast for one target time, equation (15.9). The axis is clock time; the dashed red line is the 17:00 target. Each blue bar is one vintage, beginning at its issue time and ending at the target, labeled with the temperature it predicted. Illustrative values.

Why the archive is required. Storing only the observed 17:00 temperature, or only the final vintage, makes offline evaluation unrealistically optimistic. A model evaluated with $\hat{T}_{17:00|16:00} = 38°C$ in the feature slot of a forecast that was actually issued at 12:00 is being given information that did not exist at 12:00. That is the data leakage of Chapter 1, arriving through the weather pipeline rather than through the modeling code, and it is invisible in the modeling code. What must be stored is the vintage keyed by both times, $\hat{T}_{t+h|t}$, so that an evaluation of the 5-hour-ahead forecast uses the weather forecast that was 5 hours ahead. Systems that store only observed weather report skill they will never achieve in operation, and the gap appears on the first day of production as an unexplained accuracy loss.

> **Principle 15.2.** *Every external forecast the system consumes is itself a forecast with an issue time. Archive it by issue time and target time, or the offline evaluation measures a system that cannot exist.*

15.3 Features and serving

This section covers the two stages between the cleaned data and the consumer: the feature layer, where the same definitions must be used for training and for production, and the serving layer, where the model becomes a piece of operational software with latency and throughput requirements.

15.3.1 Feature reproducibility and the feature store

The question that tests a feature layer is asked after the fact: "why did we forecast only 42 MW at 14:00 yesterday?" Answering it requires reconstructing the exact input the model received,

$$X_t\big|_{t=14:00} = g_v(\text{raw data as of } 14:00,\ \hat{T}_{\cdot|13:00},\ \text{calendar}), \tag{15.10}$$

where g_v is version v of the feature logic. Four things must therefore be versioned and retrievable: the raw data as they stood at the issue time, including which devices were reporting; the feature

logic itself; the weather vintage that was used; and the model version that consumed the result. If any one of the four has been overwritten, the forecast cannot be reproduced and the question has no answer.

A *feature store* is the component that makes equation (15.10) well defined. It holds one definition per feature, under a version, and serves the same definition to the training job and to the online path. The features of a thermostat fleet that most need this treatment are the ones with an implicit convention inside them: the lag P_{t-24}, which depends on what "24" counts and on how a missing interval is handled; the rolling duty cycle, which depends on the window length and on whether the window is closed at t or at $t-1$; the degree-day variables of Chapter 4, which depend on the base temperature and on whether the temperature is observed or forecast. When each of these exists once, the offline and online computations agree by construction. When they are written twice, they agree by coincidence.

15.3.2 Training-serving skew

Training-serving skew is what happens when they do not agree. The canonical example is a rolling mean. The training job computes

$$\bar{P}_t^{\text{train}} = \frac{1}{4}\sum_{k=0}^{3} P_{t-k}, \tag{15.11}$$

a four-interval mean, while the production feature job computes

$$\bar{P}_t^{\text{serve}} = \frac{1}{3}\sum_{k=0}^{2} P_{t-k}, \tag{15.12}$$

a three-interval mean, perhaps because the fourth lag is not reliably available in time and somebody shortened the window to make the job finish. Nothing raises an error. Both expressions return a number of the right type in the right units with a plausible magnitude. The model, however, was fitted with a coefficient appropriate to a four-interval average of a rising series, and it is now receiving a three-interval average, which on a rising load is systematically higher. The forecast drifts.

Figure 15.3 draws the two windows. The horizontal axis is time in 15-minute intervals, with the four most recent intervals marked $t-3$ through t and their load values written above them. The upper bracket spans all four intervals and is labeled with equation (15.11); the lower bracket spans only the last three and is labeled with equation (15.12). The interval that the serving window omits is crossed in red. The two averages computed from the values shown differ by 1.83 MW, which is the quantity printed between the brackets. The figure's point is that the difference is a property of the two definitions, present on every cycle, and not an error in either computation taken alone.

Skew of this kind is harder to detect than a crash, because a crash announces itself and a skew does not. The three defenses are the feature store of Section 15.3.1, which removes the possibility; a reconciliation test that recomputes the online feature vector offline from archived raw data and asserts equality to within floating-point tolerance; and the feature-distribution monitoring of Section 15.4.2, which would show the mean of the rolling-average feature shifting on the day the production job changed.

15.3.3 Pipeline leakage

Leakage can enter through the pipeline even when the modeling code is temporally correct. Suppose a daily aggregate is computed over the calendar day, 00:00 to 23:59, and joined onto every record of that day as a feature: the day's mean temperature, the day's peak load, the

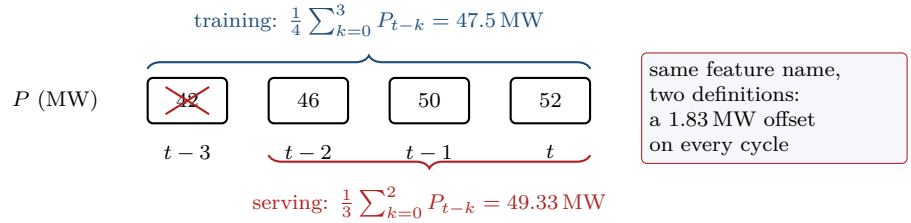


Figure 15.3: Training-serving skew between equations (15.11) and (15.12). The boxes are the four most recent 15-minute loads, in MW. The blue brace is the window used in training, the red brace the window used in production, and the crossed box is the interval the production window omits. Illustrative values.

day's total dispatched energy. A forecast issued at noon then receives a feature that summarizes the afternoon it is trying to predict. The machine-learning code has done nothing wrong: it trained on features and targets, split by time, with no look-ahead in the loss. The look-ahead was in the join.

The requirement is temporal correctness along the whole chain, not only in the model. Operationally this means that every feature carries the latest timestamp of any input that contributed to it, that the feature job asserts this timestamp does not exceed the issue time, and that aggregates used as features are defined over windows closed at or before the issue time, for example "the mean temperature over the 24 hours ending at t" rather than "the mean temperature today". Rolling windows anchored at the issue time are safe by construction; calendar windows are not, and a calendar-day aggregate is the single most common source of leakage in an otherwise correct production pipeline.

15.3.4 Model serving

Once the features exist, the model is reached through a service interface, because the optimizer, the trading system and the operator dashboard are separate processes. A request names the issue time, the fleet and the horizon, and either carries the features or asks the service to fetch them from the feature store:

```
POST /forecast
request:  { issue_time:  14:00, fleet_id:  A, horizon:  180min, features:  {...} }
response:  { p10:  38 MW, p50:  46 MW, p90:  56 MW,
             model_version:  7.3, data_quality:  0.94 }
```

The response carries the three quantiles and the two metadata fields that the consumer needs in order to decide how much to trust them. The shift this interface represents is not technical but organizational: the model is no longer an artifact in a notebook, it is operational software with an interface, a version, a latency budget, an error rate and a rollback path. Everything in Section 15.5.5 follows from treating it that way.

15.3.5 Batch and on-demand, latency, throughput and caching

There are two modes in which the service is called, and a production system usually needs both.

Batch. On a fixed schedule, at 00:00, 00:15, 00:30 and so on, the system produces the full contract of Table 15.1: all horizons, all fleets, written to a store that consumers read. Batch generation is predictable, easy to monitor and easy to backfill.

On-demand. The optimizer of Chapter 14 does not want yesterday's answer to a fixed question; it wants the expected response under a candidate control action. That is a query for $p(F \mid X_t, u)$ at a u that did not exist when the batch ran, and it can only be answered by calling the model at decision time. Any system whose forecast is an input to an optimization over actions needs this path.

Latency. The two modes have requirements that differ by more than an order of magnitude. If the optimizer must evaluate candidate actions inside a 1-second budget, a transformer that takes 20 seconds per evaluation is unusable regardless of its accuracy. If the same model runs once every 15 minutes in the batch path, a 30-second runtime is irrelevant, because 30 seconds out of 900 is slack. The consequence for model selection is direct: the latency requirement is a constraint on the architecture, evaluated alongside accuracy and not after it. A system may legitimately deploy the expensive architecture in the batch path and a cheap surrogate in the on-demand path, provided the two are compared and the difference is known.

Throughput. On-demand evaluation also has a volume problem. With 100 000 devices and 50 candidate actions to score,

$$100\,000 \times 50 = 5\,000\,000 \tag{15.13}$$

device-level evaluations are required for a single optimization pass. Simulating each device individually is out of the question at that count. The four standard reductions are cohort models, which replace 100 000 devices by a few hundred groups; vectorized inference, which evaluates the whole batch as one array operation rather than a loop; surrogate models, which replace an expensive simulator by a cheap function fitted to its output; and caching.

Caching. The structure of the on-demand query makes caching unusually effective. The feature vector X_t is the same for every candidate action; only u varies. If the model factors as

$$\hat{F} = h(\phi(X_t),\, u), \tag{15.14}$$

with ϕ the expensive state-dependent part and h a cheap function of the resulting representation and the action, then $\phi(X_t)$ is computed once per issue time and reused across all 50 actions. The saving is the ratio of the two costs, and designing the model so that this factorization exists is a serving decision made at modeling time.

15.3.6 Versioning and the reproducibility chain

Model version `7.3` is in production today and `7.4` will be next week. Because a forecast is consumed by a decision that is settled weeks later, every emitted record must carry enough version information to be reconstructed after both have been superseded. The minimum is four fields: `model_version`, `feature_version`, the input timestamp, and `weather_version`.

The use of those fields is best seen through the question a trading desk asks: "why did we commit 75 MW for the 17:00 to 18:00 window?" A complete answer is a chain with seven links:

1. the forecast issued at 14:00, retrieved by its identifier;
2. the model version that produced it;
3. the feature vector it consumed, recomputable from archived raw data by the versioned feature logic;
4. the weather vintage inside that feature vector;

5. the probabilistic forecast, not only its center, since the commitment was presumably set from a lower quantile;

6. the optimizer's decision, including the objective and the reserve margin it applied;

7. the actual response the fleet delivered.

Reading the chain forward gives data, then forecast, then decision, then outcome. Each link is a stored object, and the chain is only as strong as its weakest stored link: an unversioned feature job breaks link 3 and with it every link after.

> **Principle 15.3.** *A production forecast is reproducible when its inputs, its feature logic, its model and its downstream decision are all retrievable by version. Reproducibility is a storage design, not an analysis technique.*

15.4 Monitoring

A system that cannot be observed cannot be operated. This section sets out the four layers at which a forecasting system is monitored, what each layer sees that the others do not, why the layers differ in how quickly they react, and how thresholds are set so that an alert means something.

15.4.1 Four layers

Monitoring a forecasting system means answering four separate questions, and a signal that answers one of them says nothing about the other three:

1. **Infrastructure**: is the system running?

2. **Data**: are the inputs healthy?

3. **Prediction**: are the outputs normal?

4. **Outcome**: are the forecasts accurate?

Figure 15.4 draws them as a stack. The four boxes are the layers, infrastructure at the bottom and outcome at the top; the text to the right of each box lists the quantities that layer watches; and the arrow on the far right is the detection delay, which grows from seconds at the bottom to hours or days at the top. The stack is ordered this way because each layer depends on the ones below it: accuracy cannot be assessed if predictions are not produced, predictions cannot be produced if inputs are missing, and inputs cannot arrive if the service is down. The practical consequence is read off the right-hand arrow and developed in Section 15.4.4: the layers that detect fastest are the ones furthest from the thing the system is actually judged on.

15.4.2 Infrastructure and data monitoring

Infrastructure. The bottom layer watches the software: request latency on the forecast API, error rate, process memory and processor use, the depth of the ingestion queue, and the success or failure of scheduled jobs. None of these mentions accuracy, and the reason they matter is that an entirely accurate model can break dispatch through this layer alone. A service whose inference time degrades from 500 milliseconds to 30 seconds has not changed its predictions at all. It has made them arrive after the optimizer's deadline, which makes them unusable, and from the grid's point of view an unusable forecast and a wrong forecast have the same effect.

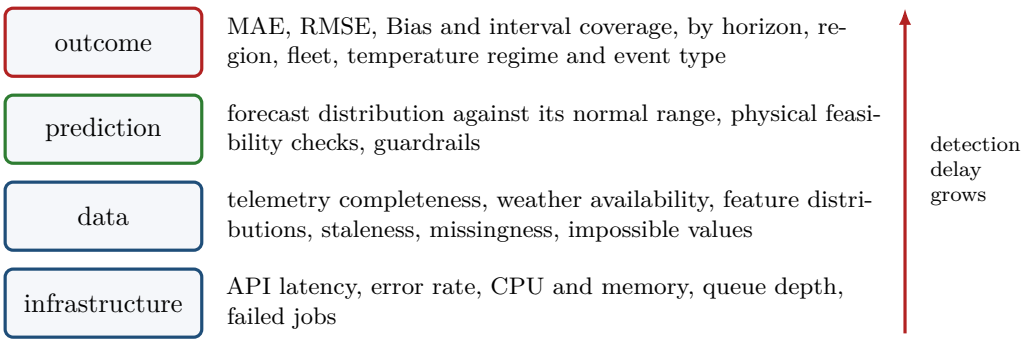


Figure 15.4: The four monitoring layers. Left: the layer. Center: what it watches. Right: detection delay, which is seconds at the infrastructure layer and up to a day at the outcome layer.

Data. The second layer watches the inputs: telemetry completeness against the enrolled count, availability and freshness of the weather feed, the distributions of each feature compared to their training distributions, the staleness statistics of equation (15.5), the missing fraction M_t of equation (15.7), and the counts of records rejected by each check in Table 15.2. This layer is the most valuable one to invest in, because it is predictive rather than retrospective. A feature whose distribution has shifted will produce bad forecasts; the shift is visible now and the bad forecasts will be visible tomorrow. Data monitoring is where the distribution-shift diagnostics of Chapter 13 are installed as standing instruments rather than run as investigations.

15.4.3 Prediction monitoring and physical checks

The third layer watches the output without reference to the outcome, which is possible because the output has a known normal range. If fleet flexibility has run between 20 and 70 MW for a year, a forecast of $\hat{F} = 400$ MW is not a bold prediction; it is a defect. The plausible causes are a units error, a fleet-size bug that has multiplied the device count, a corrupt input that passed the data checks, or a genuine model failure, and all four call for the same immediate action: reject the number before any consumer sees it. A guardrail of this kind is cheap and catches a class of failure that no accuracy metric can catch in time, because the accuracy of a 400 MW forecast is not known until the target time has passed.

Physical feasibility is the sharper version of the same idea, because the bounds come from the system rather than from history. A flexibility forecast must satisfy

$$F_t \geq 0, \qquad F_t \leq P_t^{\text{run}}, \qquad \text{SOC}_{i,t} \in [0, 1], \tag{15.15}$$

where P_t^{run} is the load currently drawn by running HVAC equipment: curtailment cannot reduce consumption below zero, cannot exceed the load that is actually running, and cannot drive a battery outside its state-of-charge limits. These are the component bounds of the physical model, and checking them at the system boundary keeps physically impossible numbers out of the optimizer. An optimizer handed an infeasible forecast will happily build a dispatch plan on it, and the infeasibility will then be discovered by the grid.

15.4.4 Outcome monitoring and the monitoring delay

The top layer computes what the earlier chapters measured. Once the target time has passed, the error

$$e_t = y_t - \hat{y}_t \tag{15.16}$$

is available, and from it the MAE, RMSE and Bias of Chapter 9 and the interval coverage of Chapter 10. What makes production monitoring different from offline evaluation is that these

are computed as continuously updated series, broken down by every dimension on which the system might fail unevenly: by horizon, by geography, by fleet, by temperature regime, and by event type. The validation protocol of Chapter 9, the calibration diagnostics of Chapter 10 and the shift detection of Chapter 13 are not one-off studies. In production they become standing dashboards.

The limitation of this layer is timing. A forecast issued today for y_{t+24} cannot be scored until tomorrow, so outcome monitoring is delayed by the horizon. Feature drift, by contrast, is observable the moment the drifted feature arrives. The two layers therefore answer different questions at different speeds: data monitoring says "the inputs are no longer what the model was fitted on", immediately and without proof of harm; outcome monitoring says "the forecasts are worse", with proof and a delay. A system needs both, and using only the second means every degradation is discovered after it has already cost something.

15.4.5 Alerting

Monitoring produces signals; alerting decides which of them wake somebody. Table 15.3 gives three thresholds of the kind an aggregator sets, with the layer each belongs to and the action each implies. The thresholds are not statistical significance levels. They are operational decision boundaries, chosen so that crossing one changes what somebody does.

Table 15.3: Alert thresholds and the action each implies. Illustrative values for a thermostat fleet whose flexibility runs between 20 and 70 MW.

Signal	Threshold	Action
telemetry completeness	below 80 %	investigate ingestion; prepare the degraded mode of Section 15.5.2
7-day bias $\lvert\text{Bias}_{7d}\rvert$	above 5 MW	the model is systematically off; trigger the retraining pipeline of Section 15.5.5
90 % interval coverage	below 75 %	intervals are too narrow; recalibrate before any further firm commitment is set from P_{10}

The 5 MW bias threshold is chosen against the scale of the forecast and the scale of the decision: on a fleet that commits tens of megawatts, a persistent 5 MW bias is a commercially meaningful systematic error, while a 1 MW bias is within the noise of a 5 MW root-mean-square error. The 75 % coverage threshold is set below the nominal 90 % by enough that sampling variation on a few hundred intervals does not cross it, because an interval whose coverage is genuinely 90 % will occasionally measure 86 % on a short window.

The discipline that makes the table work is to alert only on conditions that are actionable. Operators who receive alerts they cannot act on stop reading alerts, and the next alert they ignore is the one that mattered. A monitoring system with a hundred thresholds and no priority is less useful than one with the three rows of Table 15.3.

15.5 Failure handling and retraining

The failures listed in Section 15.1.1 are not exceptional events to be prevented; they are the ordinary operating conditions of a system with external dependencies. This section specifies what the system does when each of them occurs, and then turns to the one scheduled change the system makes to itself, which is retraining.

15.5.1 The weather fallback hierarchy

When the weather provider cannot be reached, the forecast is still due. There is an ordered sequence of substitutes, each weaker than the last:

1. the most recent successfully retrieved forecast, which is the correct values for the correct target times, merely issued earlier than intended;

2. an alternate provider, which is current but has a different bias and a different resolution from the one the model was fitted on;

3. persistence weather, meaning the assumption that the observed temperature holds over the horizon, which is the weather analog of the persistence baseline of Chapter 3;

4. climatology, meaning the historical average for this calendar day and hour, which carries no information about today at all.

Two things must accompany the substitution. The source actually used is recorded in the forecast's metadata, so that an error analysis weeks later can separate forecasts made on real weather from forecasts made on climatology. And the predictive distribution is widened as the fallback weakens, because each step down the list adds uncertainty to the driver that dominates the forecast. A system that falls back to climatology and emits the same interval width it emits on live weather is making a false statement about its own confidence.

15.5.2 Telemetry loss and graceful degradation

A larger failure is the sudden loss of a large block of telemetry: an ingestion outage, a cellular provider incident, a regional power failure, and 30 % of the fleet stops reporting. A device-level forecast is now impossible for nearly a third of the devices, and the response is not to refuse.

The degraded path falls back to an aggregate or cohort-level model built from what is still available: the remaining telemetry, which still characterizes the cohorts it covers; historical participation rates for the dark cohorts, which say what fraction of them has responded under similar conditions in the past; the weather forecast, which is unaffected by the telemetry outage; and the known fleet size, which bounds the total. These four inputs support a forecast whose central value may be only slightly worse than the nominal one and whose uncertainty is much larger. The firm capacity committed from it is reduced accordingly, because the commitment is set from a lower quantile and the lower quantile has moved down.

This is graceful degradation: the output degrades in accuracy and in confidence, in a way the consumer is told about, instead of disappearing. The alternative, a system that returns no forecast when an input is missing, forces the operator to act with nothing, which is strictly worse than acting with a wide interval that is honestly labeled as wide.

15.5.3 Model fallbacks and why they must be validated

The model itself is a dependency, and the same logic applies to it. A production forecaster keeps a hierarchy of three:

- **primary**: the gradient-boosted or sequence model of Chapters 5 to 8, with the full feature vector;

- **secondary**: a weather regression, needing only the temperature forecast and the fleet size, of the kind used as a baseline in Chapter 3;

- **tertiary**: seasonal persistence, needing only the fleet's own recent history.

Each needs strictly fewer inputs than the one above it, which is what makes the hierarchy useful: the failure that disables the primary model is usually a missing input, and a model that does not need that input is unaffected.

Figure 15.5 draws the selection logic as a decision tree. The root at the top is a new issue time. The first test, drawn as an italic box, asks whether the inputs are complete and fresh and the primary service is responding; the right branch, taken when the answer is yes, leads to the primary model with nominal intervals. The left branch, taken when the answer is no, leads to a second decision on which dependency failed, with three outcomes: a weather failure enters the hierarchy of Section 15.5.1, a telemetry failure enters the aggregate model of Section 15.5.2, and a service failure drops to the secondary and then the tertiary model. All three degraded branches converge on the bottom box, which is the common obligation: widen the intervals, label the forecast as degraded, and reduce the firm commitment. The figure's shape carries the design rule: there is no path from a failed input to no output.

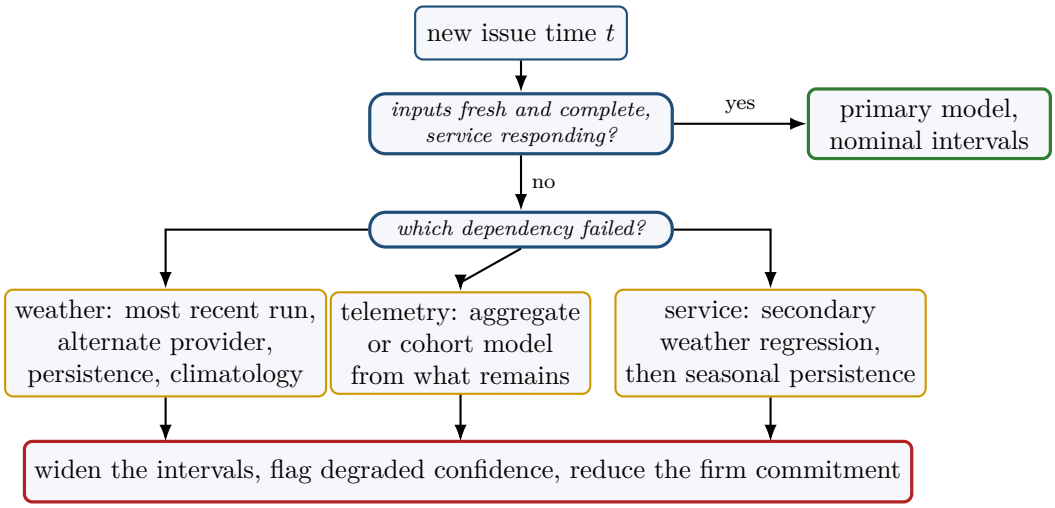


Figure 15.5: The fallback decision tree evaluated at every issue time. The two italic boxes are the decisions, the green box is the primary model, gold boxes are the degraded input paths, and the red box is the obligation every degraded path carries. No branch terminates without a forecast.

Fallbacks must be validated. A fallback that has never been evaluated is not a fallback; it is an untested code path that runs for the first time during an incident. What has to be known in advance, from the same rolling-origin backtests that validated the primary model, is $\mathrm{RMSE}_{\mathrm{fallback}}$, $\mathrm{Bias}_{\mathrm{fallback}}$ and the interval widths the fallback should emit so that its stated coverage is honest. Those numbers have an operational use beyond reporting: if the fallback's root-mean-square error is three times the primary's, then dispatch in fallback mode can automatically require a larger reserve margin, and the degradation in the forecast becomes a quantified change in the commitment rather than an unmeasured risk. A mediocre forecast with known error characteristics is better than no forecast; a mediocre forecast with unknown error characteristics is only marginally better.

15.5.4 Degraded-mode uncertainty

What the fallback should do to the predictive distribution is specific, and it is not what an untrained intuition expects. Consider a nominal forecast and its degraded counterpart for the same target:

$$\text{primary: } (P_{10}, P_{50}, P_{90}) = (45,\ 50,\ 55)\,\mathrm{MW}, \qquad \text{fallback: } (30,\ 48,\ 65)\,\mathrm{MW}. \tag{15.17}$$

The center has barely moved, from 50 to 48 MW, because the fallback model still knows the fleet's size, the weather regime and the time of day, which is most of what determines the central tendency. The width has gone from 10 to 35 MW, a factor of 3.5, because what the fallback has lost is the information that distinguished today from a typical day of this kind.

Figure 15.6 draws the two. The horizontal axis is flexibility in megawatts; each of the two rows is one forecast, drawn as a bar from P_{10} to P_{90} with a filled marker at P_{50} and the three values printed. The upper blue row is the primary forecast, the lower red row the fallback. The two medians sit almost above one another, which is where the forecasts agree; the difference the figure is drawn to show is the length of the bars.

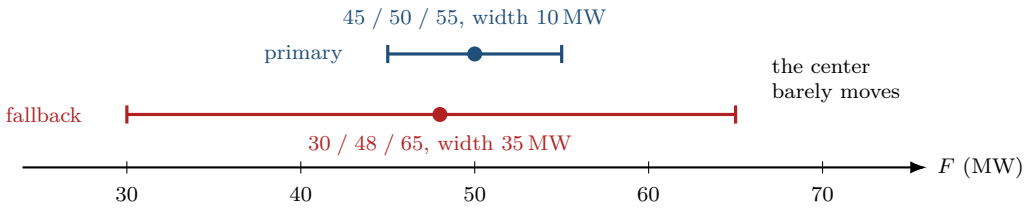


Figure 15.6: Degraded-mode uncertainty, equation (15.17). Each bar runs from P_{10} to P_{90} with the median marked. The primary interval is 10 MW wide, the fallback interval 35 MW; the two medians differ by 2 MW. Illustrative values.

This is the correct behavior, and it is what makes the degraded forecast usable. A system that fell back and kept its interval width would be asserting that it knows as much from four inputs as from forty. A system that fell back and moved its center would be asserting that the loss of telemetry changed the fleet's physics. The honest response to losing information is to say the same thing less precisely, and the optimizer of Chapter 14 can act on that, because it reads the lower quantile and will commit less when the lower quantile drops from 45 to 30 MW.

15.5.5 The retraining pipeline and safe deployment

Retraining is the one change the system makes to itself on purpose, and it is the change most likely to break it. The controlled sequence is new data, validation of those data, construction of the training set, training of a candidate, a time-aware backtest, comparison against the model currently deployed, an approval step, and deployment. Each arrow in that sequence is a place where a check can refuse to proceed.

Figure 15.7 draws the sequence together with the staged rollout that follows it. The top row, left to right, is the training path: new data, validation, training set, challenger, backtest. The bottom row, left to right, is the deployment path: comparison against the champion, shadow mode, canary at 5 %, canary at 20 %, and full deployment at 100 %. The dashed red arrow returning from the right-hand end to the champion box is the rollback path. Each stage of the bottom row exposes the challenger to more real consequence than the one before it, and each has an exit back to the champion.

Champion-challenger. The deployed model is the champion and the candidate is the challenger. Both are evaluated on the same recent rolling backtests, by the protocol of Chapter 9, and the challenger is promoted only on a convincing improvement. "Convincing" has to be defined in advance and it is not a single number: an improvement of 0.1 MW in root-mean-square error that comes with worse coverage, or with worse performance in the hot-afternoon regime where the fleet earns its revenue, is not an improvement worth a deployment.

Shadow mode. Backtests are run on archived data, which means they share the archive's defects. Shadow mode removes that limitation: the challenger runs in parallel with the champion,

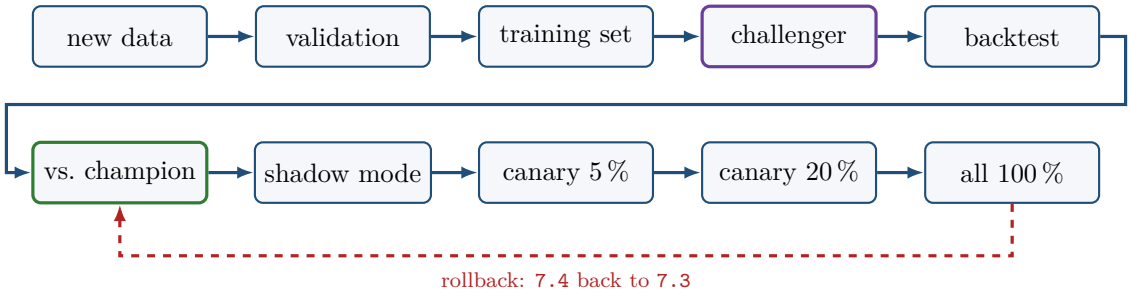


Figure 15.7: The retraining and deployment flow. Top row: the training path. Bottom row: the staged rollout, each stage exposing the challenger to more real traffic. The dashed red path is the rollback, available at every stage.

on live production inputs, at every issue time, and its forecasts are recorded and discarded. The optimizer never sees them. After a few weeks the two models can be compared on exactly the data the production system receives, including the missingness, the staleness and the weather fallbacks of the preceding sections, which no archive reproduces perfectly.

Canary deployment. When the challenger is to be used, it is used gradually: 5 % of portfolios first, then 20 %, then 100 %. The staged exposure bounds the damage of a defect that only appears under real consequence, and it provides a concurrent control group, since at each stage the remaining portfolios are still being forecast by the champion under the same weather.

Rollback. Version `7.4` must be replaceable by `7.3` in minutes, without a retraining run and without a rebuild. This is an unremarkable requirement for operational software and an unusual one for a model, and it is the clearest statement of the chapter's framing: a deployed model is production software, and it needs the same deployment, versioning and reversion machinery as any other component.

15.5.6 The registry, feature versioning and retraining triggers

The model registry. The record of what exists and what is deployed is a registry whose fields are listed in Table 15.4. The registry exists so that the question "why was `7.4` deployed?" has an answer that does not depend on anybody's memory: the row contains the data it was trained on, the hyperparameters, the validation metrics that justified promotion, the date, and the current status.

Table 15.4: Fields of a model registry entry.

Field	Content
version	the identifier carried in every forecast record, for example `7.4`
training dataset	the exact data range and snapshot the fit used
hyperparameters	the configuration of the fit, sufficient to reproduce it
validation metrics	the backtest results that justified promotion, by horizon and regime
deployment date	when it became champion, and when it stopped
status	candidate, shadow, canary, champion or retired

Feature versioning. A version number on the model binary is not sufficient. Suppose the binary is unchanged at `7.3` while the feature engineering changes: the rolling window of

Section 15.3.2 is shortened, a degree-day base temperature is adjusted, a categorical encoding gains a level. The deployed system's behavior changes while its version number does not, and no investigation that looks only at model versions will find the cause. The unit that must be versioned is therefore the package of model, feature definitions and data schema together, so that a single identifier pins all three.

Retraining triggers. Four kinds of trigger are used, and production systems normally combine them:

- **time-based**: retrain every week, whether or not anything has been observed to change, which bounds how stale the fit can be;
- **data-based**: retrain after N new dispatch events, since events are the scarce observations the response model depends on (Section 15.6.1);
- **performance-based**: retrain when a monitored metric crosses a threshold, such as the 5 MW bias of Table 15.3;
- **drift-based**: retrain when the input distribution has moved, by the detectors of Chapter 13, before any accuracy loss has been measured.

The combination matters because the triggers cover different failures. Time-based retraining alone is blind to a sudden change; performance-based retraining alone reacts only after the loss has been incurred and the delay of Section 15.4.4 has elapsed; drift-based retraining alone will fire on harmless input changes.

15.6 Learning from operation and the control loop

The pipeline described so far produces forecasts. This section covers what the system learns from having produced them: the data that dispatch generates, the attribution of errors to their sources, the records that make attribution possible, the ways in which a system that acts on its own forecasts corrupts its own training data, and the closed loop that all of this forms.

15.6.1 The event dataset and logged control actions

A flexibility forecaster has millions of routine observations and very few useful ones. Telemetry accumulates every fifteen minutes from 100 000 devices, but what the model must predict is the fleet's response to a dispatch command, and commands are issued a few dozen times a season. The response is learned almost entirely from events, which makes a dedicated event dataset the most valuable asset the system accumulates. One row per event per cohort, containing

- the pre-event state: indoor temperatures, duty cycles, setpoints, connected load;
- the action u actually taken;
- the weather during the event;
- the availability at dispatch time, N_{avail};
- the baseline estimate or control group used, in the sense of Chapter 12;
- the observed response;
- the rebound after the event ended.

Each of these is cheap to record at the time and impossible to reconstruct later, and the dataset's value grows with every event precisely because events are rare.

Logging the control action. The entry that is most often recorded badly is u. A log that says "an event occurred at 17:00" is nearly useless for fitting a response model, because the response depends on what was done. What must be stored is the setpoint adjustment in degrees, the start time, the duration, which devices were selected, and any change made during the event, such as an early release of part of the fleet or a deepened setpoint at 17:30. Without these, the response data cannot be attributed to an action, and $F = h(X_t, u)$ cannot be estimated as a function of u, which is precisely the function the optimizer of Chapter 14 needs.

15.6.2 Outcome attribution and error decomposition

When an event underdelivers, the first instinct is to call it a model error, and the first obligation is to check whether it was. The questions to ask, in order, are whether the weather forecast was wrong; whether availability fell below what was assumed; whether the response model was wrong given correct inputs; whether the action chosen was poor even though the forecast was right; whether the dispatch command failed to reach part of the fleet; whether customers overrode in unusual numbers; and whether the telemetry underreported a response that actually occurred. Seven causes, with seven different remedies, and only the third is a modeling problem.

A system that cannot distinguish them attributes every failure to the model, and then retrains the model in response to failures the model did not cause, which makes it worse. The structure that organizes the attribution is a decomposition of the total operational error into four parts,

$$\underbrace{e^{\text{total}}}_{\text{delivered minus committed}} \sim \underbrace{e^{\text{input}}}_{\text{weather, availability}} + \underbrace{e^{\text{model}}}_{\text{given correct inputs}} + \underbrace{e^{\text{exec}}}_{\text{dispatch, overrides}} + \underbrace{e^{\text{meas}}}_{\text{telemetry, baseline}}, \tag{15.18}$$

which is a diagnostic structure rather than an identity: the terms interact, and they are not literally additive. Its use is that it assigns every post-mortem finding to a term, and each term has an owner and a different fix. An input error is fixed by a better weather contract or a better availability forecast; a model error by retraining; an execution error by the dispatch path or the customer program; a measurement error by the telemetry pipeline or the baseline methodology of Chapter 12.

15.6.3 Forecast lineage and stored revisions

Attribution requires records. The record for a single forecast is its lineage, and Table 15.5 gives a concrete one. The left column is the field, the right column is its value for one forecast of a 17:00 target issued at 14:00. Read as a whole, the row says which model, fed by which feature version, consuming which weather run, under which telemetry completeness, produced which three quantiles for which target. Every question in Section 15.6.2 is answerable from records of this shape, and none is answerable without them.

Storing revisions. Table 15.5 is one vintage of one target. The 17:00 target has as many rows as there were issue times before it, and for the same target a system might hold

$$\hat{F}_{17|12} = 40\,\text{MW}, \quad \hat{F}_{17|13} = 43\,\text{MW}, \quad \hat{F}_{17|14} = 48\,\text{MW}, \quad \hat{F}_{17|15} = 52\,\text{MW}, \tag{15.19}$$

a sequence that rose by 12 MW over three hours as the weather vintages of equation (15.9) warmed. None of these may be overwritten by its successor. Three uses follow from keeping them all. The revision sequence itself is informative: a forecast that is revised by 12 MW in three hours is one whose 12:00 vintage should have carried a wider interval, and systematic revision in one direction is a bias that a monitor on revisions will detect before an accuracy monitor does. Horizon-resolved accuracy requires the vintages, since the error of the 5-hour

Table 15.5: The lineage record of one forecast. Every field is stored at issue time; none can be reconstructed afterward.

Field	Value
`forecast_id`	`82495`
`issue_time`	`14:00`
`target_time`	`17:00`
`fleet`	`A`
`model_version`	`7.3`
`feature_version`	`12`
`weather_run`	`13:00`
`telemetry_completeness`	94 %
`p10`	38 MW
`p50`	47 MW
`p90`	57 MW

forecast can only be computed from the forecast that was 5 hours ahead. And a post-mortem on a commitment needs the vintage the commitment was made on, not a later and better one, because the question is whether the decision was reasonable given what was known.

15.6.4 Feedback loops, selection bias and exploration

A forecasting system that is used to make decisions influences the data it will later be trained on, and this is a structural property of operation rather than an implementation defect.

The loop. Suppose a cohort is predicted to be highly flexible. The optimizer selects it, so it is dispatched often. Its future telemetry is therefore generated under control: its indoor temperatures, duty cycles and rebound patterns reflect the events it was sent, not its untouched behavior. The next retraining consumes that telemetry. The chain is

$$\text{model} \longrightarrow \text{decision} \longrightarrow \text{data} \longrightarrow \text{new model}, \tag{15.20}$$

and it closes. A model trained on data its own predecessor's decisions generated is not estimating the fleet's natural behavior.

Selection bias. The complement is worse. Devices and cohorts with low predicted response are never dispatched, so no response data are ever collected for them. The system therefore becomes increasingly confident about the devices it uses and remains permanently ignorant about the ones it does not, and the ignorance is self-sustaining: nothing will ever contradict a low prediction for a device that is never called. Production data are not an unbiased sample of the fleet's behavior. They are a sample selected by the optimizer, with the optimizer's own beliefs as the selection rule. This is the reason the controlled experimentation of Chapter 12 remains valuable even after years of operational data have accumulated.

Exploration. The remedy is to dispatch some actions and cohorts deliberately, within safe limits, because the system does not know what they will do. A small fraction of each event's capacity can be allocated to under-used cohorts, or to setpoint depths and durations outside the usual range, at a cost in immediate performance and a gain in the coverage of the response function $F = h(X_t, u)$. This is the production form of the exploration-exploitation trade-off raised in Chapter 12: the system gives up a little expected value now to reduce the uncertainty that limits its decisions later, and the budget for doing so is an explicit operating parameter rather than an accident.

15.6.5 Real-time revision and the state-estimation loop

Forecasting does not stop when the event starts. Suppose an event begins at 17:00 with a predicted response of 45 MW, and at 17:05 the measured response is 35 MW. That observation is information about the current state of the fleet, and the forecast for the remainder of the event must use it:

$$p(F_{18:00} \mid \mathcal{I}_{17:05}) \neq p(F_{18:00} \mid \mathcal{I}_{16:55}), \tag{15.21}$$

and the revision is downward, because the shortfall at 17:05 is evidence either that availability was overestimated or that the fleet's thermal state offered less headroom than assumed. Mid-event revision is online state estimation followed by forecast revision, and a system that cannot do it will hold a commitment it is already failing to meet.

This is the point at which the forecasting service is best understood as one block in a control loop rather than as a predictor. The loop has five stages:

$$\begin{aligned} \text{telemetry} \;\rightarrow\; \text{state estimator } \hat{x}_t \;\rightarrow\; \text{forecast } p(F_{t+h} \mid \hat{x}_t, u) \\ \rightarrow\; \text{optimizer chooses } u \;\rightarrow\; \text{control} \;\rightarrow\; \text{telemetry}, \end{aligned} \tag{15.22}$$

and each stage has a distinct job. The state estimator turns incomplete, noisy, partially stale telemetry into an estimate $\hat{x}_t$ of the fleet's physical state. The forecast propagates that state forward under a candidate action. The optimizer chooses the action. The control is executed. The next telemetry updates the state estimate to $\hat{x}_{t+1}$, and the loop repeats at the next issue time. Seen this way, several things that looked like separate engineering concerns are the same concern: missingness and staleness are properties of the state estimator's inputs, degraded-mode widening is what the estimator's uncertainty does when measurements are removed, and mid-event revision is simply the loop running at its design frequency.

15.6.6 Reliability targets, observability and business metrics

Reliability targets. A forecasting system has engineering requirements that are stated independently of its accuracy, because the accuracy is irrelevant when they are not met. A representative set is service availability of 99.9 %; a forecast delivered within 30 seconds of the issue time; a minimum telemetry completeness of 85 % for the primary model to be used; and a defined threshold below which the system switches to the fallback of Section 15.5.3. These four numbers are the contract the system offers its consumers, and they are testable without reference to any forecast error. Note the relationship between the 85 % target here and the 80 % alert of Table 15.3: the alert fires before the target is breached, which is what makes it useful.

Observability. The dashboards an operator needs are the visible surface of the four monitoring layers: the telemetry arrival rate against its expected value, the success or failure of the most recent forecast generations, the distribution of the predictions currently in force, the recent errors by horizon, the count and identity of missing devices, and the model version currently deployed. The test of observability is whether an operator can determine the health of the system in one screen at 16:55 on the hottest day of the year, which is the only time it will be needed urgently.

Model metrics against business metrics. The last distinction is between what the model is scored on and what the business is scored on. Model metrics are MAE, RMSE, coverage and calibration. Business metrics are the megawatts safely delivered, the number of failed dispatch events, the flexibility that was available but never used, the customer override rate, and the cost of rebound. The two can move independently. A model can improve its root-mean-square error by becoming more accurate on mild days, which are numerous in the validation sample and commercially irrelevant, while becoming no better on the twenty hot afternoons that generate

the revenue. A production system is therefore monitored on both, and when they disagree it is the business metric that decides, with the model metric used to diagnose why.

15.6.7 The 4:55 PM scenario and the system architecture

The chapter closes by returning to the situation of Section 15.1.1 and running it through the system that has now been specified. The time is 16:55. An event is scheduled for 17:00. The primary forecast, before the failures, expected 75 MW. Three things are wrong at once: 25 % of the telemetry is missing, the weather feed is stale, and the model service fails to respond.

A system built as this chapter describes does seven things, in order and automatically. It detects that inputs are stale and incomplete, through the data monitoring of Section 15.4.2 and the age field of equation (15.5). It rejects the primary forecast, because the completeness of 75 % is below the 85 % threshold of Section 15.6.6 and the service is not responding in any case. It activates the fallback path of Figure 15.5, which here means the aggregate cohort model on the remaining telemetry with the most recent weather vintage. It widens the uncertainty, as in Figure 15.6. It notifies the optimizer that the forecast is of degraded confidence, through the metadata of Table 15.1. It reduces the firm commitment, because the commitment is set from a lower quantile that has moved. And it continues attempting recovery, so that if the service returns at 17:02 the next cycle uses the primary model again. What it does not do is raise an exception and stop.

What the two forecasts imply. The numbers make the difference concrete. Suppose the primary forecast had been $P_{50} = 78\,\mathrm{MW}$ with $P_{10} = 70\,\mathrm{MW}$, while the fallback gives $P_{50} = 72\,\mathrm{MW}$ with $P_{10} = 52\,\mathrm{MW}$, against a commitment target of 65 MW. The primary forecast says the target is comfortably safe: its 10th percentile is above it, so the probability of falling short is below one in ten. The fallback says something different. Its median is still above the target, but its 10th percentile is 13 MW below it, so $\Pr(F \geq 65\,\mathrm{MW})$ may be well under what a firm commitment requires. The rational response is to reduce the committed quantity or to call additional resources, and that response is driven entirely by the reliability state of the forecasting system rather than by anything about the fleet. Production reliability changes the grid decision, which is the strongest available statement of why the system matters as much as the model.

Example 15.1 (The cost of a confident degraded forecast)**.** Take the fallback distribution above and treat it as approximately Gaussian. From $P_{50} = 72$ and $P_{10} = 52\,\mathrm{MW}$, the standard deviation is $(72 - 52)/1.2816 \approx 15.6\,\mathrm{MW}$, so $\Pr(F \geq 65) = \Phi\big((72 - 65)/15.6\big) = \Phi(0.45) \approx 0.67$. Committing 65 MW on this forecast fails one time in three. Had the system emitted the fallback median with the primary's interval width, the same calculation with $\sigma = (78 - 70)/1.2816 \approx 6.2\,\mathrm{MW}$ would have reported $\Phi(1.12) \approx 0.87$, and the commitment would have looked almost twice as safe as it was. The width of a degraded forecast is not a presentational detail; it is the quantity the commitment is computed from.

Conceptual questions

"What would you monitor in a production forecasting system?"
A strong answer refuses to name a single metric and instead walks the whole chain in four layers. At the data layer: completeness, latency, staleness, feature drift and weather availability. At the service layer: request latency, failure rate and the deployed model version. At the model layer: the distribution of predictions against its normal range, bias, mean absolute error and calibration broken down by horizon and by operating regime. At the operational layer: underdelivery against commitments, unused available capacity, override rates. The answer ends with the point that distinguishes an engineer from an analyst, which is that the fallback paths are monitored and tested too, since a fallback that has never run is not known to work.

Conceptual questions

"What happens if half the telemetry disappears?"
The answer begins by rejecting the premise that nothing changes. It then describes a sequence: quantify which cohorts are affected, since a uniform 50 % loss and a 50 % loss concentrated in one climate zone are different problems; switch the affected cohorts to aggregate features and historical participation rates; widen the predictive intervals to reflect the lost information; reduce the firm capacity offered, because the commitment is set from a lower quantile that has moved down. The closing point is that all of this must have been validated in advance, so that the error characteristics of the degraded forecast are known numbers rather than hopes.

Conceptual questions

"How would you retrain the model safely?"
A strong answer describes a pipeline rather than a procedure: version the data and the features so the training set is reproducible; train a challenger; evaluate it on time-aware backtests that explicitly include the critical regimes, not only the average day; run it in shadow mode or behind a canary against the current champion; promote it only on both forecast metrics and operational metrics; and keep rollback to the previous version immediate. The answer notes that the model, its feature definitions and its data schema are promoted together as one package, because changing the feature logic under an unchanged model version changes behavior invisibly.

Conceptual questions

"Why store old forecast versions at all?"
Because a forecast is defined by its information set, and that set grows with time. Keeping every vintage makes four things possible: reproducing what the optimizer knew when it committed, which is the only fair basis for judging the decision; evaluating accuracy by horizon, which requires the forecast that was actually that far ahead; analyzing the revision sequence, which reveals bias and under-dispersed intervals earlier than accuracy monitoring does; and diagnosing a failure without the error of comparing a past decision against a later, better-informed forecast.

The architecture. Figure 15.8 assembles the whole chapter. The upper path, left to right, is the forward chain: the weather provider and the devices feed telemetry ingestion, which feeds data quality, which feeds the feature store, which feeds the forecast model, which emits P_{10}, P_{50} and P_{90} to the optimizer, which issues a dispatch command to the devices. The lower path carries the two returns: the observed response enters monitoring, monitoring feeds the training

pipeline, and the training pipeline writes to the model registry, which supplies the deployed model back to the forecast block. The two dashed arrows are those return paths, and they are what make the diagram a system rather than a pipeline. Four properties are distributed across the whole figure rather than located in any one box: versioning, which every record carries; monitoring, which observes every stage; fallbacks, which exist on every input edge; and auditability, which is the ability to reconstruct any past forecast from the stored lineage.

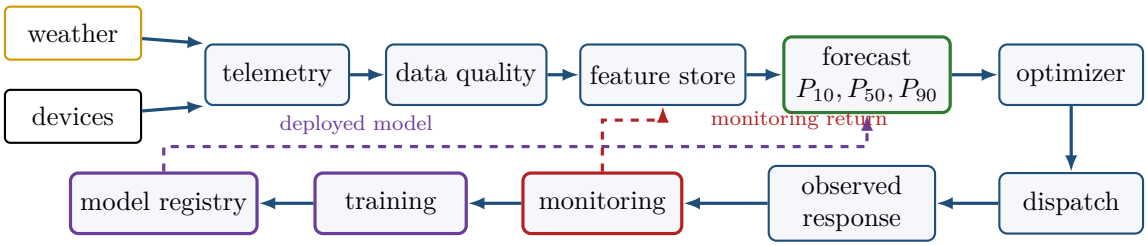


Figure 15.8: The production architecture. Solid blue arrows are the forward chain from inputs to dispatch. The dashed purple arrow is the training return path, which replaces the deployed model; the dashed red arrow is the monitoring return path, which observes the inputs and the outputs. Versioning, monitoring, fallbacks and auditability are properties of the whole figure, not of any box in it.

15.7 What to retain

Definition 15.1 (A production forecasting system)**.** A production forecasting system is the continuous operation of the chain data, features, forecast, decision, feedback, under conditions that include missing telemetry, stale inputs, weather provider failures, software failures, a changing fleet, distribution shift and deployment mistakes. Its five design principles are reproducibility, observability, graceful degradation, validated fallbacks and closed-loop monitoring, and it is successful not when its error is smallest but when it continues to support good decisions while conditions are not ideal.

Table 15.6 collects the chapter's objects with the subsection that defines each.

15.8 Exercises

Exercise 15.1. A weather regression predicts flexibility as $\hat{F} = a + b\,(\hat{T} - 28^{\circ}\mathrm{C})$ with $a = 30\,\mathrm{MW}$ and $b = 3\,\mathrm{MW}/^{\circ}\mathrm{C}$. Using the three vintages of equation (15.9), compute the forecast issued at 12:00, 14:00 and 16:00 for the 17:00 target. The realized flexibility was 58 MW. (a) Give the error of each vintage. (b) Now compute the forecast an offline evaluation would report if it used the observed 38°C in place of the vintage available at 12:00, and give its error. (c) State by how many megawatts such an evaluation would overstate the skill of the 5-hour-ahead forecast, and name the field of Table 15.1 whose absence makes the mistake possible.

Exercise 15.2. Using the four loads in Figure 15.3, $P_{t-3} = 42$, $P_{t-2} = 46$, $P_{t-1} = 50$ and $P_t = 52\,\mathrm{MW}$: (a) verify the two rolling averages of equations (15.11) and (15.12) and their difference. (b) If the fitted model uses this feature with coefficient 0.8, give the resulting forecast offset in megawatts. (c) The forecast's normal range is 20 to 70 MW and the 7-day bias alert fires above 5 MW. State whether the prediction guardrail of Section 15.4.3 would catch this skew, whether the bias alert of Table 15.3 would, and after how many days of a constant offset of this size the second would fire if the offset is the only source of bias.

Table 15.6: The objects of a production forecasting system.

Object	Definition	Section
forecast contract	horizons, quantiles and metadata delivered every cycle	15.1.2
$\hat{F}_{t+h\|t}$	a forecast identified by its issue time	15.1.3
quality check	physical, operating-range, timestamp, cadence and unit tests	15.2.2
$\Delta t_i(t)$	measurement age, used as filter, feature and health metric	15.2.3
M_t	missing fraction of the fleet, included as a feature	15.2.5
weather vintage $\hat{T}_{t+h\|t}$	the weather forecast keyed by issue and target time	15.2.7
feature store	one versioned definition per feature, shared offline and online	15.3.1
training-serving skew	any disagreement between offline and online feature values	15.3.2
pipeline leakage	future information entering through a join, not the model	15.3.3
latency and throughput	the serving constraints that bound architecture choice	15.3.5
four monitoring layers	infrastructure, data, prediction, outcome	15.4.1
guardrail	rejection of a forecast outside its physically feasible range	15.4.3
fallback hierarchy	ordered substitutes needing progressively fewer inputs	15.5.3
graceful degradation	a worse, wider, labeled forecast instead of none	15.5.4
champion-challenger	paired evaluation with promotion on demonstrated gain	15.5.5
shadow and canary	live evaluation without consequence, then staged exposure	15.5.5
model registry	the versioned record of what was trained and deployed	15.5.6
event dataset	one row per event: state, action, weather, response, rebound	15.6.1
error decomposition	input, model, execution and measurement terms	15.6.2
lineage record	everything stored at issue time about one forecast	15.6.3
feedback loop	model, decision, data, new model	15.6.4
reliability targets	availability, deadline, completeness, fallback threshold	15.6.6

Exercise 15.3. Treat the two forecasts of equation (15.17) as Gaussian and recover each standard deviation from $\sigma = (P_{90} - P_{10})/2.563$. (a) Give σ for the primary and the fallback. (b) Compute $\Pr(F \geq 44\,\mathrm{MW})$ under each. (c) Give the largest commitment each distribution supports at 95 % confidence. (d) Repeat part (c) for the 16:55 pair of Example 15.1, where the primary has $P_{50} = 78$ and $P_{10} = 70\,\mathrm{MW}$ and the fallback has $P_{50} = 72$ and $P_{10} = 52\,\mathrm{MW}$, and compare both with the 65 MW target.

Exercise 15.4. An event dispatched against the lineage record of Table 15.5 delivered 34 MW against a committed 45 MW. The post-mortem establishes that the realized temperature was 2°C below the `13:00` weather run, that availability at dispatch was 8 % below the value assumed, that 3 MW of commands were not acknowledged by the devices, and that the override rate was normal. (a) Assign each finding to a term of equation (15.18). (b) State which findings justify retraining the model and which do not. (c) Given monitored values of telemetry completeness 78 %, a 7-day bias of -6 MW and a 90 % interval coverage of 81 %, say which rows of Table 15.3 fire and what each one obliges the operator to do.

Chapter 16

Capstone: eighty thousand thermostats, the hottest day of the year and 75 MW for four hours

16.1 The case and the decision

The fifteen chapters before this one each developed one piece of machinery: a problem statement, the structure of energy series, baselines, features, tree ensembles, autoregressive dynamics, recurrent networks, attention, validation, probabilistic output, fleet aggregation, counterfactual response, distribution shift, the handoff to an optimizer, and the production system that runs the whole thing on a schedule. This chapter uses all of them at once, on a single operating problem of the kind an aggregator meets a few times each summer.

The case is fixed for the rest of the chapter. The aggregator operates 80 000 enrolled residential thermostats. Tomorrow is forecast to be the hottest day of the year. A utility has asked for 75 MW of load reduction from 16:00 to 20:00. The chapter's question is

What forecasting system supports the decision of how to dispatch eighty thousand thermostats so that 75 MW *of load reduction is delivered continuously from 16:00 to 20:00?*

The answer is not a load forecast. It is a chain: state estimation, baseline forecasting, control-response forecasting, uncertainty quantification, optimization, and closed-loop updating during the event. Nine components, each the subject of an earlier chapter, have to be assembled so that the number at the end of the chain is one an operator can act on.

Jargon: firm capacity, reserve, sustained delivery, headroom, lineage

Firm capacity is the quantity a resource can be relied on to deliver at a stated reliability, as opposed to its expected or its maximum output; a fleet whose expected response is 100 MW may have a firm capacity of 75 MW or less. *Reserve* is capacity that is available but deliberately not dispatched, held back to cover forecast error, device dropout and requests that arrive during the event. *Sustained delivery* is the requirement that the committed quantity be met in every interval of a block, not on average over the block and not at its start. *Headroom* is the amount of physical slack a device or cohort still has before a constraint binds: for a thermostat, how far the indoor temperature can still be allowed to rise before the comfort limit is reached. *Lineage* is the end-to-end record that lets an outcome be traced back through the chain that produced it: which forecast was issued, which command was sent, which devices received it, which responded, and what was measured.

Key idea

The system being designed does not map historical load to future load. It maps current fleet state, forecast weather and a *candidate control action* to a distribution over the future controlled response, and hands that distribution to an optimizer. The central object is $p(F_{t+1:t+H} \mid \mathcal{I}_t, u_{t:t+H})$, and the system is successful only if it helps produce reliable decisions.

This section states the decision, fixes the issue time and the information available at it, defines the target as a trajectory rather than a number, and shows with the case's own figures why a request for 75 MW over four hours is not a request for 75 MW.

16.1.1 The decision the forecast serves

The first move is the one Chapter 1 insisted on and Chapter 14 formalized: start from the decision, not from the data. A system designed to answer "what will the fleet's load be tomorrow at 17:00?" will produce a well-validated number that no operator can use, because the operator is not choosing the load. The operator is choosing which cohorts to call, how deep a setpoint change to send, when to start, how long to hold, and how much capacity to keep in reserve. The question the system must answer is therefore

which control strategy gives a sufficiently high probability of delivering 75 MW *continuously from 16:00 to 20:00, without unacceptable customer impact and without a rebound that moves the problem rather than solving it?*

Three features of that sentence drive every design choice that follows. It is about a *control strategy*, so the forecast must be conditional on an action. It asks for a *probability*, so the output must be a distribution, not a point. And it carries two side constraints, customer impact and rebound, so the objective is not delivery alone.

Why the framing is load-shaped and the answer is not. The request arrives in the units of a load forecast, megawatts over a window, which is why the problem is so often mistaken for one. But the quantity requested does not exist independently of the aggregator's own actions. There is no fact of the matter about tomorrow's 75 MW: there is a fleet whose response depends on its thermal state at 16:00, on the weather between 16:00 and 20:00, and on what the aggregator decides to do. The forecast is an input to a decision that changes the thing being forecast, which is the defining feature of the counterfactual target of Chapter 12.

16.1.2 Issue time and the information set

The main decision is taken at 14:00 the day of the event. Fixing that instant fixes everything else, because a forecast may use nothing outside the information set $\mathcal{I}_{14:00}$ available at it. The discipline is the one of Chapter 1: every feature in the model must exist at 14:00, with the values it actually had at 14:00, and a feature computed from data that arrives at 15:00 is leakage no matter how good it makes the backtest look.

Table 16.1 lists what $\mathcal{I}_{14:00}$ contains for this case. The first column groups the contents, the second says what the group holds, and the third says which component of the system consumes it. The groups divide naturally into three kinds: measurements of the fleet's present state, forecasts of its future environment, and records of its past behavior. The table is worth reading before any model is chosen, because it is the complete list of raw material: a design that needs something not in the table is not implementable at 14:00.

The object the system must produce at 14:00 is then

$$p(F_{16:00-20:00} \mid \mathcal{I}_{14:00},\, u_{16:00-20:00}), \tag{16.1}$$

Table 16.1: The information set $\mathcal{I}_{14:00}$ for the case of this chapter, and which component of the system uses each group.

Group	Contents at 14:00	Consumed by
Aggregate load	fleet power at 15-minute resolution up to 14:00	baseline model, lags
Device telemetry	per-device power, reporting status, firmware version	availability, data quality
Thermal state	indoor temperature, setpoint, the gap $T_{\text{in}} - T_{\text{set}}$	response model, headroom
Equipment state	compressor on or off, duty cycle D_t	response model
Availability	devices online and program-eligible, N_{avail}	both models, dispatch limits
Event history	previous dispatches, delivered response, overrides	response model, reliability
Weather observed	outdoor temperature, humidity RH_t up to 14:00	state, bias correction
Weather forecast	$\hat{T}_{t+h\mid 14:00}$ through 22:00	both models, shift detection
Calendar	weekday, holiday, school status	baseline model
Composition	equipment mix, climate zones, cohort shares	fleet aggregation

the distribution of the delivered response over the event window, given what is known at 14:00 and given a candidate control plan u for the window. Equation (16.1) is the whole deliverable of the forecasting system. Everything in the rest of the chapter either constructs it, quantifies it, uses it or revises it.

16.1.3 The target as a trajectory of two load paths

The response is a difference of two load paths. Write $P^{\text{base}}(t)$ for what the fleet would consume over the event window with no action, and $P^{\text{ctrl}}(t \mid u)$ for what it would consume under the control plan u. The delivered flexibility is their difference,

$$F(t, u) = P^{\text{base}}(t) - P^{\text{ctrl}}(t \mid u), \tag{16.2}$$

exactly the counterfactual object of Chapter 12: only one of the two terms is ever observed, because dispatching the fleet destroys the baseline it would have followed.

The target is not F at one instant. It is the whole trajectory over the window,

$$\big[F_{16:00},\ F_{16:15},\ F_{16:30},\ \ldots,\ F_{20:00}\big], \tag{16.3}$$

seventeen quarter-hourly values, or at hourly resolution the four numbers $F_{16}, F_{17}, F_{18}, F_{19}$ of the event's four hours. The reason the trajectory is the target and not its average is physical: flexibility decays over an event. A thermostat held above its setpoint consumes headroom; once the indoor temperature reaches the comfort limit the compressor returns, and the device stops contributing. A fleet under curtailment therefore delivers less in its fourth hour than in its first, and the shape of that decline is the main thing the response model has to get right.

16.1.4 Seventy-five megawatts at 16:00 is not 75 MW for four hours

The case's central difficulty appears as soon as the trajectory is written down. Suppose the fleet could provide 90 MW at the start of the event, and that the response model predicts the

hourly values

$$F_{16} = 90\,\text{MW}, \quad F_{17} = 82\,\text{MW}, \quad F_{18} = 68\,\text{MW}, \quad F_{19} = 52\,\text{MW}. \tag{16.4}$$

Read as a single capacity figure, the fleet looks comfortable: 90 against a request of 75 is a 20 % margin. Read as a trajectory, it fails. The third hour is 7 MW short and the fourth is 23 MW short, so the commitment is met in half the window and missed in half.

Figure 16.1 draws equation (16.4). The horizontal axis is the hour of the event, with ticks at the start of each of the four hours and a final tick at 20:00 where the window closes; the vertical axis is deliverable flexibility in megawatts. The blue curve with circular markers is the predicted trajectory, labeled with its four values. The horizontal red line at 75 MW is the utility's request, held across the whole window. The shaded red region is the area between the request and the trajectory where the trajectory lies below it. Reading the figure: the first two hours clear the line with margin, the curve crosses the line at about 17:30, and the shaded region is the undelivered energy of the last two and a half hours. The figure is the case in one picture, and it is the reason the target of this system is equation (16.3) rather than a single megawatt number.

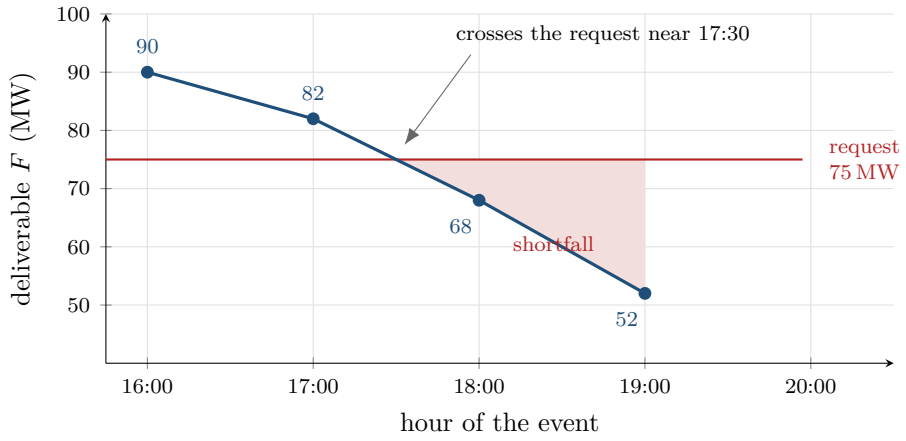


Figure 16.1: The decay of deliverable flexibility over the event, from the hourly values of equation (16.4), against the 75 MW request held across the whole window. The curve is piecewise linear between the four hourly values; the shaded area is the region in which the trajectory lies below the request. Values are illustrative of the case.

The specification the system must therefore satisfy is a statement about the whole window,

$$\Pr\big(F_t(u) \geq 75\,\text{MW} \quad \text{for all } t \in [16{:}00,\ 20{:}00]\big) \quad \text{sufficiently high}, \tag{16.5}$$

and not $F_{16} > 75\,\text{MW}$. The difference between equation (16.5) and its first-hour version is the difference between a commitment the fleet can keep and one it cannot.

Principle 16.1. *A multi-interval commitment is a statement about the whole trajectory. Firm capacity is set by the weakest interval of the block, not by the first one and not by the average.*

16.2 The baseline and response models

Two of the nine components are forecasting models in the ordinary sense, and they forecast the two terms of equation (16.2). This section specifies both, chooses their model families, compares the two architectures that can be built from them, and then deals with the problem that makes this case harder than a load-forecasting case: the historical labels for flexibility are themselves model output.

16.2.1 Component 1: the baseline forecast

The baseline forecast predicts $P^{\text{base}}(t)$, the fleet's consumption over 16:00 to 20:00 under no action. It is an ordinary load forecast, and the whole apparatus of Chapters 2 to 8 applies to it unchanged. Its input vector at issue time is

$$X_t = \big[\, P_t,\ P_{t-1},\ P_{t-24},\ P_{t-168},\ \hat{T}_{t+h},\ \text{RH}_t,\ \text{hour},\ \text{weekday},\ N_{\text{avail}}(t),\ \text{composition} \,\big], \quad (16.6)$$

where P_t is the current fleet load, P_{t-1} the previous interval, P_{t-24} the same hour yesterday, P_{t-168} the same hour last week, $\hat{T}_{t+h}$ the forecast outdoor temperature at the target time, RH_t humidity, and the last two entries the number of eligible devices and the fleet's equipment and climate mix. The lag structure comes from the daily and weekly periodicity established in Chapter 2; the weather terms and the degree variables built from them come from Chapter 4; and the availability and composition terms come from Chapter 11, which showed that a fleet forecast without them confuses a change in behavior with a change in the population.

The model family. The first production model for $P^{\text{base}}(t)$ is gradient boosting, not an LSTM and not a transformer. The reasons are properties of the data rather than preferences. The input of equation (16.6) is structured tabular data with engineered lags already in it, which is the case Chapter 5 identified as the natural home of boosted trees. The weather response is nonlinear with thresholds: above roughly 35°C a large fraction of air conditioners stop cycling and run continuously, and a tree ensemble represents such a break with a split rather than having to approximate it. The interactions that matter, temperature with humidity, temperature with hour of day, availability with duty cycle, are exactly what a depth-limited tree ensemble captures without being told about them. And the practical properties decide the rest: boosted trees train in minutes, can be diagnosed with feature importance and partial dependence, tolerate missing values, and deploy as a single artifact. A sequence model is a candidate to be tested later against a validated incumbent, not a starting point.

16.2.2 The baseline hierarchy

Before the boosted model is trusted, the baseline discipline of Chapter 3 applies: a sophisticated model earns its place only by beating simple ones on held-out data, and the simple ones must be built first. Table 16.2 lists the hierarchy for this case. The first column is the level, the second the forecast rule, and the third what a model has to do better in order to justify itself. Levels 0 to 3 are the required baselines; the candidate and the optional model are what the hierarchy is used to evaluate. The pattern in the table is that each level adds one kind of information: the last value, the daily cycle, the weekly cycle, the weather and the calendar, and finally the nonlinear interactions among all of them.

The hierarchy does more than guard against complexity for its own sake. It gives the production system the fallback chain of Section 16.5.5: when the inputs of the candidate model are degraded, the system does not guess, it drops to a level of Table 16.2 whose inputs are still available and reports that it has done so.

16.2.3 Component 2: the response model

The second model forecasts the effect of the action. It can be written either as the controlled load $P^{\text{ctrl}}(t \mid u)$ or directly as the response $F(t, u)$, and in either form it is conditional on a control plan. Its input vector is physical rather than historical,

$$X_t = \big[\, T_{\text{out}},\ T_{\text{in}},\ T_{\text{set}},\ D_t,\ N_{\text{avail}}(t),\ \text{recent control history} \,\big], \quad (16.7)$$

and its control argument is the decision the operator is making,

$$u = \big[\, \Delta T_{\text{set}},\ \text{start time},\ \text{duration},\ \text{cohort selection} \,\big]. \quad (16.8)$$

Table 16.2: The baseline hierarchy for $P^{\text{base}}(t)$, with the candidate model and the conditional sequence model. Levels 0 to 3 are built first; the candidate must beat all of them on rolling-origin validation.

Level	Forecast rule	What it contributes
0	persistence, $\hat{P}_{t+h} = P_t$	the short-horizon floor; hard to beat below one hour
1	daily seasonal, $\hat{P}_{t+h} = P_{t+h-24}$	the daily cycle, free of any fitting
2	weekly seasonal, $\hat{P}_{t+h} = P_{t+h-168}$	the weekday-weekend distinction
3	weather and calendar regression	the temperature response, linearly
candidate	gradient boosting on equation (16.6)	nonlinear thresholds and interactions
optional	LSTM or temporal fusion transformer	trajectory information not in the features, if validation shows it

Each entry of equation (16.7) is there for a physical reason. Outdoor temperature sets how fast the house gains heat and therefore how quickly headroom is consumed. Indoor temperature and the setpoint gap $T_{\text{in}} - T_{\text{set}}$ say how much headroom exists before the comfort limit binds. The duty cycle says what fraction of the fleet's compressors are running, and only a running compressor can be curtailed. Availability says how many devices will receive the command at all. Recent control history says whether the headroom has already been spent on an earlier event today.

The object the response model produces is the conditional trajectory distribution

$$p(F_{t+1:t+H} \mid X_t,\ u), \tag{16.9}$$

which is the central forecast of the entire system. It is conditional on an action, it is a distribution rather than a number, and it is over a trajectory rather than a single interval. Every one of those three properties is required by the decision of Section 16.1.1, and none of them is a property of an ordinary load forecast.

16.2.4 Two architectures

There are two ways to assemble equation (16.2) from fitted models.

Architecture A: model the two loads separately. Fit a baseline model and a controlled-load model, and subtract:

$$\hat{P}_0 = f_0(X_t), \qquad \hat{P}_u = f_u(X_t, u), \qquad \hat{F} = \hat{P}_0 - \hat{P}_u. \tag{16.10}$$

The appeal is interpretability. The two terms can be inspected and validated separately: the baseline against ordinary load data on non-event days, of which there are many, and the controlled load against event data, of which there are few. A post-mortem can say whether an underdelivery came from a wrong baseline or from a wrong response, which is a distinction the operator cares about.

Architecture B: model the difference directly. Fit one model whose target is the response itself:

$$\hat{F} = f_F(X_t, u). \tag{16.11}$$

The appeal is that the quantity of interest is the quantity fitted, so the model's loss function is aligned with the decision, and the two error terms of equation (16.10) cannot both be large and cancel in a way that flatters the validation.

Figure 16.2 draws both. The left column holds the inputs: the state and feature block built from Table 16.1, the weather forecast, and the candidate control plan u. The two blue boxes in the middle are the baseline model f_0 and the response model f_u of equation (16.10); the circular node is the subtraction, and its output feeds the green box on the right, the probabilistic flexibility of equation (16.9) with its scenarios. The dashed purple arrow is architecture B, a direct path from the inputs to the same output that bypasses the two intermediate loads. The figure shows what the choice costs: architecture B deletes the two middle boxes, and with them the ability to attribute an error to one of the two terms. Note also that the control plan u enters only the response model in architecture A, which is the formal statement that the baseline is what happens when nothing is done.

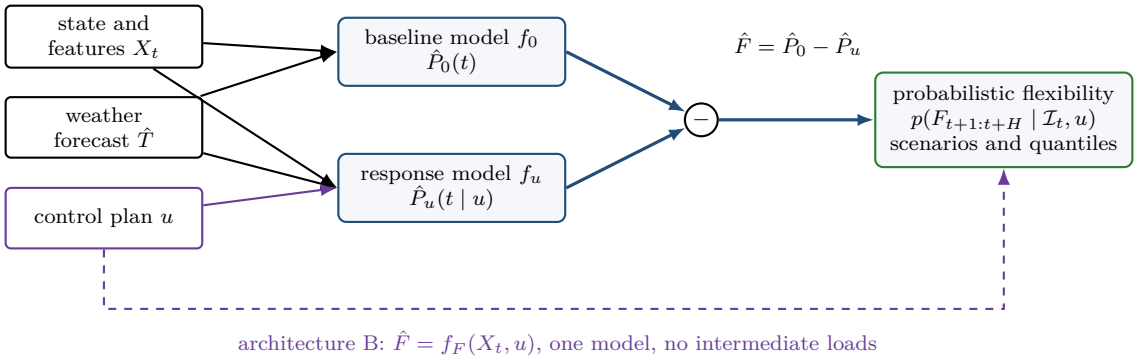


Figure 16.2: The two-model architecture. Architecture A, equation (16.10): a baseline model and a response model are fitted separately and subtracted at the circular node. Architecture B, equation (16.11), is the dashed path: one model maps the same inputs directly to the response. Both terminate in the same probabilistic output.

Both architectures should be built and compared on the same rolling-origin event backtest, by the protocol of Chapter 9. The comparison is not purely empirical, however, because of how the training labels for either architecture were constructed.

16.2.5 Historical labels are not truth

Both architectures need historical examples of F to fit against, and no such examples exist in the data. What the aggregator observed in each past event is the controlled load P^{ctrl}, the load that actually occurred while the fleet was dispatched. The baseline that would have occurred was never observed, so every historical flexibility label was produced as

$$F_t^{\text{label}} = \hat{P}_{0,t} - P_t^{\text{observed}}, \tag{16.12}$$

with the first term the output of whatever baseline method the program used at the time. The label therefore carries that method's error.

The consequence is a specific and serious one. A model trained on equation (16.12) learns the sum of the true response and the historical baseline bias, and it will reproduce that bias on every future forecast. If the program's old baseline method was a same-day-average rule that systematically overstated what a hot afternoon would have consumed, then every label overstates flexibility, the fitted model overstates flexibility, and the validation, which scores predictions against the same biased labels, confirms that it is accurate. The error is invisible from inside the data set.

The practical consequence is a question to ask before any modeling begins. Presented with a historical event data set, the first thing to establish is how the historical event baselines were generated, because the answer determines whether the flexibility labels can be trusted at all.

A program whose baselines came from a weather-adjusted similar-day rule has labels with a known and estimable bias; a program that cannot say how its baselines were produced has labels of unknown quality, and fitting a model to them transfers that unknown quality into every future commitment. The same question returns as an objection in Section 16.6.1.

16.2.6 Experimental evidence and randomized holdouts

The remedy that removes the problem rather than bounding it is experimental. On each dispatch, hold back a random subset of the fleet. With 80 000 eligible devices, dispatch 70 000 and hold out 10 000 chosen at random, and the held-out group's measured load, scaled to the dispatched group's size, is a contemporaneous estimate of the baseline under the same weather, the same calendar and the same fleet state. Randomization is what makes the two groups comparable: no matching rule, no weather normalization and no same-day adjustment is required, because the assignment itself guarantees that the groups differ only in treatment.

The cost is real and small. A 10 000-device holdout gives up roughly an eighth of the deliverable response on each event, which on the case's numbers is of order 10 MW. What it buys is unbiased labels for every event, which is the only basis on which a response model can be trusted on the day its forecast is used to commit capacity. The question to ask of any historical dispatch data set, before fitting anything, is therefore whether randomized control or holdout cohorts were part of the program design.

> **Principle 16.2.** *Flexibility labels are model output, not measurement. Either a randomized holdout makes the baseline observable, or the baseline method's bias is inside every label, every fitted model and every validation score.*

16.3 Fleet representation and uncertainty

The two models of the previous section were written for a single feature vector, but the case has eighty thousand devices. This section chooses the level at which the fleet is represented, defines the cohorts, shows why the uncertainty of the fleet total does not shrink the way a count of devices suggests, and fixes the form in which the distribution is reported.

16.3.1 Component 3: three levels and the compromise

Eighty thousand separate models are not the answer, and neither is one model of the fleet total. The three candidate levels, developed in Chapter 11, are device, cohort and fleet.

Device level. One forecast per thermostat. This is the only level that supports highly individualized dispatch, in which the operator chooses which particular devices to call based on their state and their override history. It costs eighty thousand predictions per interval, and its target is so noisy, because one household's response depends on whether somebody happened to open a door, that device-level error metrics look poor however good the model is.

Cohort level. One forecast per group of devices that share predictive structure. Dispatch is then issued per cohort, which is in fact how most programs operate, and the target is far less noisy because the idiosyncratic part has already averaged out within the group.

Fleet level. One forecast of the total. This is the smoothest target and the simplest production path, and it cannot say which devices produced the number, so it cannot be used to select them.

The compromise this case adopts has three parts: a single global response model, trained on all device-level and cohort-level observations so that every parameter is estimated from the whole fleet's data; device and cohort features supplied to that model, so it can distinguish a heat pump in a hot-dry zone from a conventional air conditioner in a humid one; and aggregation of its output to the fleet total, which is the level at which the 75 MW commitment is made. The arrangement separates the level at which the model is trained from the level at which the forecast is used, and those are independent choices.

16.3.2 Cohorts

A cohort earns its existence only if its members would be expected to respond similarly to the same weather and the same command. Table 16.3 lists the six dimensions along which this fleet is partitioned, with the reason each one splits the response. The first column is the dimension, the second is what it changes physically or behaviorally, and the third says whether it is known at enrollment or has to be estimated from telemetry. Three of the six are physical and three behavioral, and a partition that uses only the physical ones will group reliable participants with customers who override every event.

Table 16.3: Cohort dimensions for the thermostat fleet, with what each one changes and where it comes from.

Dimension	Why it splits the response	Source
Climate region	sets the weather the cohort experiences and the shared forecast error	enrollment
HVAC type	heat pump and conventional units differ in capacity and recovery	enrollment
Recent duty cycle	only running compressors can be curtailed	telemetry
Indoor-setpoint gap $T_{\text{in}} - T_{\text{set}}$	how much headroom remains before the comfort limit	telemetry
Historical event response	the delivered megawatts per device in past events	event history
Override tendency	the fraction of events the customer cancels	event history

With cohorts $k = 1, \dots, K$ defined, the fleet response is the sum of the cohort responses,

$$F_{\text{fleet}}(t) = \sum_{k=1}^{K} F_k(t), \tag{16.13}$$

and the dispatch decision is a choice of which F_k to activate and when. Equation (16.13) is what makes the cohort the natural unit of this case: it is simultaneously the unit the model can forecast with acceptable noise, the unit the dispatch system can address, and the unit a post-mortem can attribute an error to.

16.3.3 Individual errors do not cancel

A count of eighty thousand devices invites the conclusion that individual errors average away. They do not, and the reason is the second term of the variance of a sum,

$$\operatorname{Var}\Big(\sum_{i} F_i\Big) = \sum_{i} \operatorname{Var}(F_i) \;+\; 2\sum_{i<j} \operatorname{Cov}(F_i, F_j). \tag{16.14}$$

The first term grows like the number of devices and, divided by an expectation that also grows like the number of devices, produces the $1/\sqrt{N}$ improvement of Chapter 11. The second term grows like the square of the number of devices, so any positive pairwise covariance eventually dominates however small it is.

This fleet has positive covariance by construction. A single weather forecast feeds every device in a metropolitan area, so a 2°C error at 18:00 is an error in every thermostat's prediction at once. Extreme heat changes behavior collectively: above a threshold temperature, air conditioners stop cycling fleet-wide and the fleet leaves the regime the model was fitted on, all at the same moment. A firmware release changes the control logic of a software version, which is to say of a large block of devices simultaneously. A holiday or a regional event shifts occupancy patterns across the whole fleet. A telemetry outage removes visibility of many devices at once, and the devices it removes are not a random sample. Each of these puts a positive term into equation (16.14), and on the hottest day of the year several of them are active together, which is exactly when the commitment is being made. The design consequence is that the fleet distribution must be constructed with its correlation structure retained, and never by adding per-device variances.

16.3.4 Component 4: probabilistic output and scenarios

The output of the response model is not $\hat{F} = 85\,\text{MW}$. For the first hour of the event it is a set of quantiles,

$$\text{P10} = 68\,\text{MW}, \qquad \text{P50} = 84\,\text{MW}, \qquad \text{P90} = 96\,\text{MW}, \tag{16.15}$$

produced by the quantile regression or conformal methods of Chapter 10. For the four-hour window the output must carry the dependence across hours as well, and the practical form is a set of trajectories. Three of them for this case are

$$s_A = [88, 84, 78, 71], \quad s_B = [82, 77, 70, 62], \quad s_C = [92, 88, 85, 80] \quad \text{MW}, \tag{16.16}$$

each a plausible internally consistent path through the four hours, with decay built into each path rather than imposed afterward.

Figure 16.3 plots them. The horizontal axis is the hour of the event and the vertical axis is deliverable flexibility in megawatts. The three marked curves are the trajectories of equation (16.16) and the horizontal red line is the 75 MW request, held across the window. Reading the figure: all three scenarios clear the request in the first hour, which is why a first-hour forecast would have reported no problem; scenario s_C clears it in all four hours; scenario s_A falls short only in the last hour; and scenario s_B falls short in the last two. Counted over equally weighted scenarios, one path of three sustains the commitment, which is the kind of calculation the probability in Section 16.5.6 reports. The quantiles of equation (16.15) are the first-hour marginals of the same distribution these paths are drawn from, and the three paths shown are ordinary draws rather than the P10, P50 and P90 trajectories, which is why their first-hour spread is narrower than the marginal interval.

16.3.5 Independent hourly quantiles are not enough

Suppose the system reports, hour by hour, that $\Pr(F_t \geq 75\,\text{MW}) = 0.95$. That is four marginal statements, and the commitment is a joint one. If the four hours were independent the probability of delivering in all of them would be

$$0.95^4 = 0.8145 \approx 0.81, \tag{16.17}$$

so four comfortable-looking hourly figures become a four-in-five commitment. And the hours are not independent. They share the weather forecast, the baseline error, the availability estimate

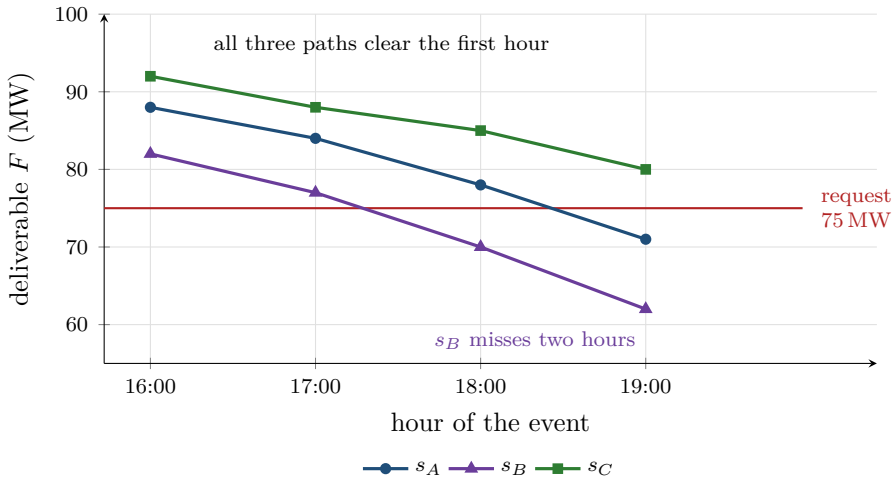


Figure 16.3: Three scenario trajectories from equation (16.16) over the four hours of the event, against the 75 MW request held across the window. Each trajectory declines because curtailment consumes thermal headroom; scenario s_C sustains the request, s_A misses the last hour and s_B the last two. Values are illustrative of the case.

and the thermal state of the same devices, so an hour that comes in low is followed by an hour that comes in low for the same reason. Positive correlation moves the joint probability back up toward 0.95, and perfect correlation would recover it exactly; the point is that neither 0.81 nor 0.95 can be obtained from the marginals, because the marginals do not contain the dependence.

This is why equation (16.16) is reported as trajectories and not as twelve quantile numbers. A scenario set carries the dependence in its construction: each path is drawn with correlated weather, baseline and availability errors, so the fraction of paths that sustain the commitment is a direct estimate of the quantity in equation (16.5). The optimizer then evaluates each candidate action on each path, which is the only formulation in which a sustained-delivery constraint is computable.

Principle 16.3. *Per-hour exceedance probabilities do not determine the risk of a multi-hour commitment. The deliverable is a joint distribution over the trajectory, carried as scenarios.*

16.4 The hottest day and the optimizer

Two facts about the case have been set aside until now: tomorrow is hotter than anything in the training data, and the forecast exists in order to choose an action. This section handles both. It first treats the extrapolation explicitly, then states the optimization problem, defines the decision variable, and develops the three ideas that make the decision dynamic rather than a single allocation: reserve, headroom and rebound.

16.4.1 Component 5: forecasting outside the validated domain

The maximum outdoor temperature in the training data is 38°C. Tomorrow's forecast high is 42°C. Every model in the system is therefore being asked for a prediction four degrees outside the range on which it was fitted and validated, at exactly the moment its output is used to commit capacity.

Chapter 13 named the situation: the input is outside the support of the training distribution, and no validation statistic computed on historical data says anything about accuracy there. A

boosted tree ensemble will extrapolate as a constant, returning the value it learned at the edge of its data, which understates a response that keeps growing with temperature. A linear or neural model will extrapolate along its fitted trend, which may overstate it. Neither behavior is a prediction in the sense the validation report certifies, and the difference between the two is not resolvable from the data.

The required engineering response is not to suppress the problem but to make it a field of the output. The system flags the forecast as out of distribution, states the reason, namely that the forecast temperature exceeds the training maximum, and passes the flag downstream so that it reaches the optimizer rather than stopping at a dashboard.

Operating under the shift. Five actions follow from the flag, and they are actions rather than caveats. First, compare models that extrapolate differently: if the boosted ensemble, a simple weather regression and a physics-informed limit disagree materially at 42°C, the spread among them is itself a measure of the extrapolation risk, and if they agree the risk is smaller. Second, bring in physical limits: a thermostat cannot curtail more than its connected compressor load, and summing nameplate capacities over available devices gives a bound no statistical model can exceed. Third, look at cohort behavior in the hottest historical events available, even if there are only a handful, because the direction in which behavior moved as those events got hotter is informative even when the magnitude is not. Fourth, enlarge the uncertainty: widen the quantiles of equation (16.15) beyond what the residual distribution implies, because the residual distribution was measured inside the validated domain. Fifth, reduce the firm commitment, which is the only one of the five that changes the outcome rather than the description of it.

Constrain the actions, not only the forecast. The shift applies to the control variable as well as to the weather. If historical dispatches rarely used a setpoint offset above 2°C, then a candidate plan with $\Delta T_{\text{set}} = 4$°C is an extrapolation in the action dimension, and the response model has no evidence about it. The model will return a number, typically by scaling what it learned at 2°C, and there is no reason to believe the response is linear in offset: a deeper offset may deliver more per device, or may trigger overrides and deliver less. The optimizer should therefore be restricted to the region of action space where the response has been observed, and a plan outside it treated as a research question rather than an option.

16.4.2 Component 6: the optimization problem

The forecast now meets the decision. Let $D_t = 75\,\text{MW}$ be the required delivery in every interval of the window. Three formulations are available, and they are not equivalent.

The weakest is the expectation constraint,

$$\mathbb{E}\big[F_t(u)\big] \geq D_t \qquad \text{for each } t, \tag{16.18}$$

which requires the average outcome to meet the target. It is weak because it is satisfied by a plan that delivers 75 MW half the time and fails half the time, and because it is satisfied interval by interval without any statement about the block.

The chance-constrained form is the one the case needs,

$$\min_{u \in \mathcal{U}} \ \mathbb{E}\big[C(u)\big] \qquad \text{subject to} \qquad \Pr\big(F_t(u) \geq D_t \ \ \forall t \in [16, 20]\big) \geq 1 - \alpha, \tag{16.19}$$

with $1 - \alpha = 0.95$ a typical requirement. The quantifier inside the probability is the whole point: it is one probability of one joint event, which is the sustained-delivery specification of equation (16.5) and the reason the scenarios of equation (16.16) were required.

The third formulation prices the failure instead of forbidding it,

$$\min_{u \in \mathcal{U}} \ \mathbb{E}\big[C_{\text{customer}}(u) + C_{\text{under}}(u) + C_{\text{rebound}}(u)\big], \tag{16.20}$$

where the three terms are the comfort cost imposed on customers, the penalty for delivering less than D_t, and the cost of the post-event rebound of Section 16.4.7. Equation (16.20) is the more honest statement when the penalty for underdelivery is known in money, because it lets the optimizer trade a small probability of a small shortfall against a large and certain comfort cost, which the hard constraint of equation (16.19) cannot. In practice both are solved and compared: the chance constraint says what reliability costs, and the expected-cost form says whether that reliability is worth buying.

16.4.3 What the decision variable is

The variable u in equations (16.19) and (16.20) is not a single number. At minimum it is the four-part object of equation (16.8): which cohorts to call, how deep a setpoint change to send, when to start, and for how long. For a four-hour event it is better written as a sequence of interval decisions u_t, because the useful plans are staged rather than simultaneous. A concrete staged plan for this case is

16:00: call cohort A; 16:30: add cohort B;
18:00: release part of A to recover; hold cohort C in reserve throughout. (16.21)

Equation (16.21) is a control trajectory, and it is what turns the problem from an allocation into the dynamic problem of Chapter 14. Each element of it changes the state the later elements act on, which is why the response model had to be conditional on recent control history in equation (16.7).

16.4.4 Preserving reserve

The fleet is expected to be able to provide about 100 MW at 16:00 and the target is 75 MW. The dispatch that follows from that arithmetic is not to call everything.

A sound plan dispatches to a target of about 80 MW of expected response and holds roughly 20 MW in reserve. The reserve covers four distinct things. It covers forecast error, because the expected 100 MW has a distribution around it whose lower tail reaches well below 75 MW. It covers device dropout, the devices that are counted as available at 14:00 and do not respond at 16:00 through connectivity failures, overrides or a command that never arrived. It covers decay, the decline of Figure 16.1, which is certain to happen and uncertain in magnitude. And it covers requests that arrive during the event, since a utility that asked for 75 MW at 14:00 may ask for more at 18:00. A fleet dispatched to its expected capacity has no answer to any of the four.

Expected capacity is not firm capacity. The distinction is the one in the jargon box, and it is the single most common error in reading a forecast. An expected response of 100 MW with a wide distribution and a declining trajectory may support a firm commitment of only 75 MW, or of less. The offer follows from the required reliability and the whole trajectory, not from the central estimate of the first interval.

16.4.5 Thermal headroom as a state of charge

The reserve argument becomes precise once headroom is treated as a state variable. Define $\mathrm{HR}_{k,t}$ as the thermal headroom of cohort k at time t: how much further the indoor temperature

of its members can be allowed to rise before the comfort limit binds. Curtailment consumes it and recovery replenishes it,

$$\mathrm{HR}_{k,t+1} = g(\mathrm{HR}_{k,t},\ u_{k,t},\ T_t), \tag{16.22}$$

with the outdoor temperature T_t entering because a hotter day both consumes headroom faster during curtailment and replenishes it more slowly during recovery. The analogy that makes the structure familiar is the battery of Chapter 14, whose state of charge evolves as

$$\mathrm{SOC}_{t+1} = \mathrm{SOC}_t - \eta\, P_t\, \Delta t. \tag{16.23}$$

Equations (16.22) and (16.23) are the same kind of object: a stock that an action draws down, that refills when the action stops, and whose level bounds what the resource can deliver next. A thermostat fleet is a thermal battery whose state of charge is comfort, and the dispatch problem is the problem of managing that stock over four hours.

16.4.6 Sequencing cohorts

The headroom state immediately rules out the intuitive dispatch rule. Suppose cohort A can deliver 40 MW immediately but recovers slowly, and cohort B can deliver 30 MW and recovers quickly. Ranking cohorts by immediate response calls A first. Over a four-hour event that is the wrong order: A's headroom, once spent, is not available again within the window, whereas B's is. Using B earlier and preserving A gives the plan a second resource to call in the third and fourth hours, which is precisely when Figure 16.1 shows the commitment being missed.

Figure 16.4 shows the consequence for the fleet's headroom. The horizontal axis is clock time across the event, and the vertical axis is aggregate remaining headroom as a percentage of what the fleet had at 16:00. The red curve is the strategy that calls the highest-response cohort first and holds it; the blue curve is the staged strategy of equation (16.21), which uses the fast-recovering cohort first, adds the slow one later and releases part of it at 18:00. The dashed horizontal line is the headroom below which the fleet can no longer hold 75 MW. Reading the figure: both strategies start at 100 % and both decline, because curtailment always consumes headroom; the staged strategy declines more slowly, shows a visible recovery after the 18:00 release, and stays above the dashed line to the end of the window, while the highest-response-first strategy reaches the floor at 19:00 and spends the last hour of the event below it. At 20:00 the staged strategy still holds 59 % of its initial headroom against the other's 34 %. The two curves represent the same total energy curtailed, allocated differently in time.

The dispatch problem is therefore a dynamic control problem over a depleting stock, not a knapsack over fixed capacities. The optimizer must look ahead over the whole window, because the value of calling a cohort at 16:00 includes the cost of not having it at 19:00.

16.4.7 Rebound

The event ends at 20:00 and the problem does not. Eighty thousand homes that have been held above setpoint for four hours on a 42°C day all begin cooling at once, and the fleet's load jumps above what it would have been with no event at all. A rebound of +30 MW over 20:00 to 22:00 is a realistic figure for a dispatch of this size.

Three consequences follow. The forecast window must extend past the event: the response model is asked for $P^{\mathrm{ctrl}}(t \mid u)$ through 22:00, not through 20:00, and the rebound is part of its output rather than a separate study. The optimizer must carry the rebound term of equation (16.20), because a plan that maximizes delivery during the window by driving every cohort to its comfort limit also maximizes the rebound after it. And the operational conclusion has to be stated plainly: if the grid is still constrained at 20:30, a dispatch that delivers 75 MW between 16:00 and 20:00 and returns 30 MW of excess at 20:15 has moved the problem rather than solved it. Staging the release of cohorts, which Figure 16.4 introduced for headroom reasons, is also the instrument that spreads the rebound.

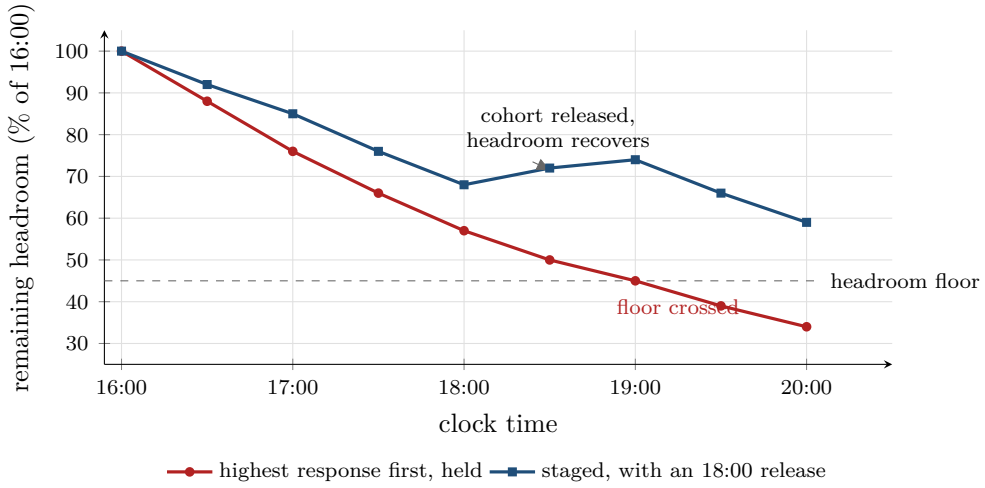


Figure 16.4: Aggregate fleet headroom over the event under two sequencing strategies, as a percentage of the headroom available at 16:00. The dashed line is the level below which 75 MW can no longer be held. The staged strategy of equation (16.21) uses the fast-recovering cohort first and releases part of the slow one at 18:00, which produces the visible recovery. Both curves are illustrative and are drawn from stated coordinates, not from a fitted model.

16.5 Closing the loop and validating the design

The 14:00 forecast is not the last forecast. This section follows the case forward through the event, treats the observed response as the most informative measurement the system will ever get, and then specifies the two things that make the design a system rather than a model: the validation plan and the behavior under failure.

16.5.1 Component 7: closed-loop updating

The decision at 14:00 commits to a plan, and the plan is revised as information arrives. Three revision points matter for this case.

At 15:00 the weather forecast has been re-issued and an hour of telemetry has accumulated. The information set has grown but contains nothing qualitatively new, so the revision is a quantitative adjustment: the expected capacity moves by a few megawatts, the quantiles tighten slightly, and the staged plan is re-solved against them.

At 15:45 the pre-event state is as well observed as it will ever be before the fact. Indoor temperatures, duty cycles and availability are current, and the forecast horizon to the start of the event is fifteen minutes rather than two hours. This is the last opportunity to change the composition of the first dispatch rather than to react to it.

At 16:05 the situation changes in kind. The first five minutes of actual response have been measured. The information set now contains a direct observation of the quantity being forecast, under the actual weather, with the actual fleet, under the actual command, and $\mathcal{I}_{16:05}$ is far more informative about the rest of the event than $\mathcal{I}_{14:00}$ ever was. The forecast to revise is

$$p(F_{16:15-20:00} \mid \mathcal{I}_{16:05},\ u), \tag{16.24}$$

and the revision is not a refinement. It can change the plan.

Figure 16.5 draws the sequence. The horizontal line is clock time from 14:00 to 16:05, with the four decision points marked. The boxes above the line say what information each point adds; the boxes below say what action is taken. Reading left to right: the 14:00 point produces the initial plan from forecasts alone; 15:00 and 15:45 refine it against better forecasts and a

better-observed state; and 16:05 replaces forecast response with measured response, which is the first point at which the system learns whether its model of the fleet was right. The asymmetry in the figure is deliberate: the three left-hand points differ in degree, the right-hand one differs in kind.

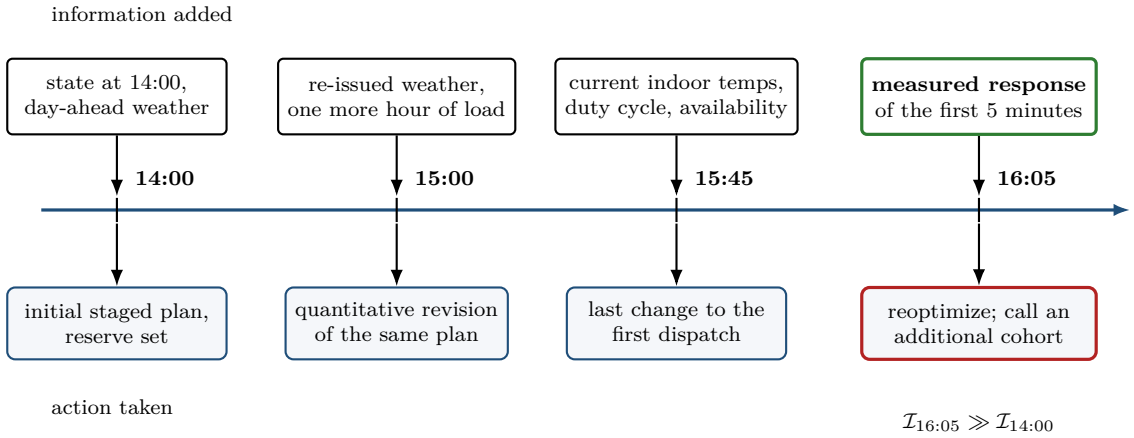


Figure 16.5: The closed loop from the main decision to the first revision inside the event. Above the time line, the information each point adds; below it, the action taken. The 16:05 point is marked differently because it is the first time the system observes the quantity it has been forecasting.

16.5.2 The first five minutes

Suppose the 14:00 forecast predicted an initial response of 80 MW and the measurement at 16:05 is 67 MW. The gap is 13 MW, about one sixth of the prediction, and it appears five minutes into a four-hour commitment.

Five explanations are available, and they are not equally actionable. Fewer devices responded than were counted as available, which is an availability error and is checkable immediately from command acknowledgments. The baseline was wrong, so the fleet is consuming less than the counterfactual assumed and the measured difference understates the physical response. Indoor temperatures at the start of the event were higher or lower than the state estimate believed, so the headroom assumption was wrong. Communication failures prevented commands from reaching part of the fleet, which looks like an availability error but has a different fix. And customers overrode the event at a higher rate than the cohort's history implied.

Whatever the cause, the response is the same in form: reoptimize against equation (16.24) and, if the revised trajectory no longer sustains 75 MW, dispatch another cohort from the reserve that Section 16.4.4 insisted on holding. This is the moment the reserve earns its cost. A plan that dispatched the full expected 100 MW at 16:00 has nothing left to call at 16:05, and its only remaining option is to deepen the setpoint offset on devices that are already curtailed, which consumes headroom faster and makes the fourth hour worse.

16.5.3 State estimation and the resemblance to predictive control

The step performed at 16:05 has a standard name. Given a measurement y_t of the fleet's response, the system updates its belief about the unobserved state,

$$p(x_t \mid y_{1:t}) \;\longrightarrow\; \text{revised } p(F_{t+1:t+H} \mid \mathcal{I}_t,\; u), \tag{16.25}$$

where x_t collects the quantities that are not measured directly: the distribution of indoor temperatures, the remaining headroom per cohort, the true availability, the fraction of devices

that have overridden. The measured response is informative about all of them at once, which is why one observation at 16:05 is worth more than an hour of pre-event telemetry.

The loop that results is

$$\text{forecast} \to \text{dispatch} \to \text{observed response} \to \text{state update} \to \text{reforecast} \to \text{redispatch},$$

which is model predictive control with a statistical model in place of a physical one. The resemblance is not a metaphor: the structure is the same, with a forecast over a receding horizon, an optimization over that horizon, execution of the first step only, and re-solution after the next measurement. Chapter 14 developed the optimization step; what this case adds is that the model inside the loop is learned and its uncertainty is part of the loop's state.

16.5.4 Component 8: the validation plan

Nothing above is trustworthy without a validation plan that tests the whole chain rather than its first link. The protocol is the rolling-origin backtest of Chapter 9, run across historical events: train on everything before an event, forecast that event as it would have been forecast at 14:00 on the day, score the result, and advance. Table 16.4 lists what is scored. The first column is the dimension, the second the metric, and the third the question a failure on that row answers. The table is ordered from the easiest dimension to the most important: baseline accuracy is cheap to measure and least decisive, and the decision outcome on the last row is the only row that says whether the system did its job.

Table 16.4: The validation plan: what is scored in a rolling backtest over historical events, and what each row tells the designer.

Dimension	Metric	What a failure tells the designer
Baseline accuracy	MAE, RMSE, Bias of $\hat{P}_0$ on non-event days	the counterfactual term of equation (16.2) is wrong
Response accuracy	event-level error of $\hat{F}$ against holdout-based labels	the conditional model of the action is wrong
Calibration	empirical coverage of P10, P50, P90	the stated probabilities are not the observed frequencies
Sustained delivery	error in the trajectory shape; is the decay captured?	the system will commit to blocks it cannot hold
Rebound	error in the 20:00 to 22:00 load	the optimizer is pricing rebound from a wrong forecast
Decision outcome	would the optimizer, given this forecast, have met the target?	the only row that scores the system rather than the model

Aggregate scores hide the cases that matter. Every row of Table 16.4 must also be reported on the slices where the system is actually used, because an average over all events is dominated by mild days and short dispatches. Six slices matter here: the hottest 5 % of days, which is the regime of this case; short events against long ones, since decay only appears in the long ones; the one-hour horizon against the four-hour horizon; high availability against degraded availability; fresh headroom against headroom already spent earlier in the day; and by geography and cohort, since a fleet-level score can be good while one region is badly wrong. A model that is well calibrated on average and overconfident on the hottest 5 % of days is a model that is overconfident exactly when it is consulted.

16.5.5 Component 9: failure handling in production

The production discipline of Chapter 15 supplies the last component, and the case gives it a concrete test. At 15:55, five minutes before the event, three things go wrong at once: telemetry completeness has fallen to 65 %, the weather feed is stale, and the primary response model times out.

The forecast cannot be skipped, because a dispatch is about to happen, and it must not be fabricated. The system therefore has a tiered fallback, each tier using fewer and more reliable inputs than the one above it, and each tier declaring what it is.

Figure 16.6 draws the arrangement. The leftmost box is the data-quality gate, which measures completeness, staleness and model health. The three stacked boxes are the tiers: the primary global boosted or temporal-fusion response model conditioned on the full feature set; the fallback cohort-level regression, which uses the latest reliable weather and cohort aggregates and does not need per-device telemetry; and the emergency fallback, which looks up the measured response of historically similar events and applies a conservative capacity figure. The arrows from the gate select a tier, and every tier emits both a forecast and a degradation flag into the box on the right, which is the optimizer. The flag is the load-bearing part of the figure: the optimizer that receives a tier-two forecast is told that the forecast is degraded, and it responds by increasing the reserve and reducing the firm commitment. A system that silently imputed the missing telemetry would have produced a number of the same shape with none of that information attached.

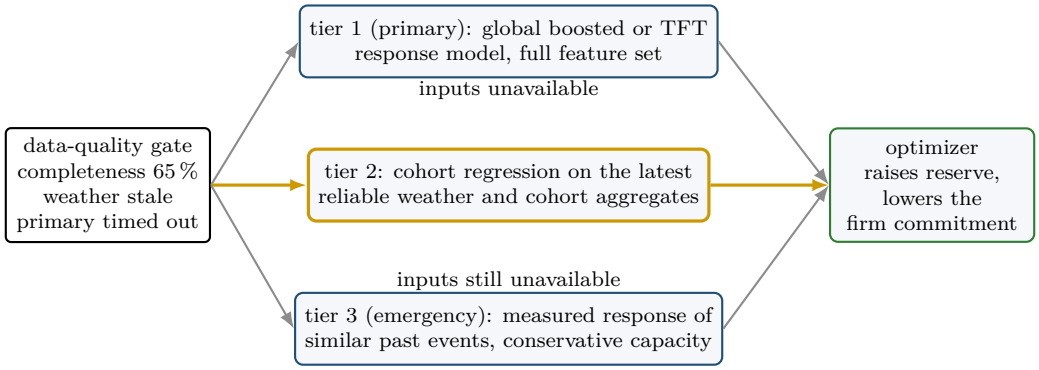


Figure 16.6: The three-tier fallback at 15:55. The gate selects the highest tier whose inputs are available; the gold path is the one taken in this case. The flag that accompanies the forecast is what lets the optimizer respond to degradation instead of treating a tier-two number as a tier-one number.

16.5.6 The forecast output contract and what it implies

Everything the system knows has to be expressed in the message it sends downstream, and the message is a contract rather than a number. Table 16.5 is the contract for this case as issued at 14:00. The first column is the field, the second its value, and the third why the field exists. The fields fall into four groups: what is being forecast and when, the central estimate and the quantiles around it, the probability of the event the operator actually cares about, and the health indicators that say how much of the rest to believe.

The immediate conclusion. The decisive row of Table 16.5 is the sustained-delivery probability of 72 %. Against a requirement of 95 % it says plainly that 75 MW is not firm capacity for

Table 16.5: The forecast output contract for the case, issued at 14:00. Values are those of the case; the last three rows are the fields that a point forecast would omit.

Field	Value	Why it is in the contract
Issue time	14:00	fixes the information set; anything later is leakage
Event window	16:00 to 20:00	the block the commitment covers
Median capacity	84 MW	the central estimate of the first hour
P10	69 MW	the conservative commitment level
P90	98 MW	the upside, relevant to surplus offers
Pr(sustain 75 MW)	72 %	the number the commitment decision uses
Out-of-distribution flag	HIGH	forecast temperature exceeds the training maximum
Telemetry completeness	94 %	how much of the state estimate is measured
Model and forecast identifiers	version, run id	lineage, so the outcome can be traced back

this fleet on this day under this plan. Five responses are available and they are not exclusive. Reserve additional cohorts, which raises the probability by making the reserve of Section 16.4.4 larger. Reduce the offered capacity, which raises the probability by lowering the threshold in equation (16.19). Dispatch more devices, which raises the expected response but spends headroom faster and may lower the probability in the fourth hour. Modify the timing, staging the dispatch as in equation (16.21) so that the decay is flattened. Or wait for telemetry, if the commitment deadline allows, since the 15:45 information set is materially better than the 14:00 one. What the forecast must not do is report 84 MW and let the operator infer that the request is covered.

16.5.7 What to start with, and what would change it

The design above says what the system computes. The remaining question is what model families to put in it, and the answer is deliberately conservative.

- **Baseline load $P^{\text{base}}(t)$:** gradient boosting on equation (16.6), after the hierarchy of Table 16.2 has established what it must beat.
- **Flexibility response:** gradient boosting at cohort or fleet level, conditioned on the control variable of equation (16.8).
- **Uncertainty:** quantile boosting for the marginals, or calibrated residual intervals and conformal prediction from Chapter 10, with the correlated draws that produce equation (16.16).
- **Temporal structure of the response:** explicit event-duration and recent-control features first, and a sequence model only if those features demonstrably fail.

The list does not start with a transformer, and the reason is not conservatism for its own sake.

Why gradient boosting first. This problem has strong structured predictors and few of them: outdoor temperature and its forecast, the recent load lags, the duty cycle, the number of available devices, the indoor-setpoint gap, the control intensity, and the record of recent events.

The relationship between those predictors and the response is nonlinear with interactions, which is what a boosted ensemble of shallow trees represents efficiently, as Chapter 5 established. The non-statistical reasons are equally decisive in a system that will be operated by people at 15:55: boosted models are quick to train, their errors can be attributed to features, their failure modes are understood, and they deploy as one artifact with no inference-time state. A sequence model must demonstrate incremental value against this incumbent on the rolling event backtest of Table 16.4; it does not get the benefit of the doubt.

What would justify a sequence model. A specific empirical finding would. Suppose residual analysis shows that two cohorts with identical current indoor temperature, identical outdoor temperature and identical current load respond differently, and that the difference tracks their preceding six hours of thermal history: one was cooled hard all morning and has cold thermal mass, the other has been drifting up since noon. That is trajectory information that no fixed set of lag and duty-cycle features is capturing, and it is precisely what a recurrent hidden state of Chapter 7 or the attention mechanism of Chapter 8 is built to carry. A temporal fusion transformer is attractive in this case for a further structural reason: it combines historical observed signals, static cohort attributes, known future inputs such as the weather forecast and the planned control schedule, and multi-horizon outputs, which is exactly the shape of equation (16.9). The argument for it is the residual diagnostic, not the architecture.

16.6 The pushback round and the end-to-end design

A design of this kind is examined by being argued with. This section collects the sixteen objections the case attracts, grouped by what they are about, and then assembles the answers into the two summaries the chapter has been building toward: the architecture as an engineering diagram and the same loop as a short piece of mathematics.

16.6.1 Sixteen questions

The questions below are the ones a skeptical colleague actually asks. Each is paired with the shape of a strong answer rather than with a slogan.

Model choice and forecasting level. The first three objections all say that the design is more complicated than it needs to be.

Conceptual questions

"Why not aggregate everything?"
A strong answer concedes the statistical point first: aggregation averages out device-level noise, and for the total capacity figure it genuinely helps, which is the $1/\sqrt{N}$ argument of Chapter 11. It then gives the reason aggregation is not sufficient here. Reliability, control cost and remaining headroom differ by cohort, and the decision of Section 16.4.3 is a choice of which cohorts to call in what order, which an aggregate number cannot inform. The resolution is to forecast at the level dispatch is issued at, cohort or global-device, and reconcile upward to the fleet total, with the resolution chosen to match the dispatch system rather than the reporting system.

Conceptual questions

"Why do you need an LSTM?"
The honest answer is that it may not be needed. A strong answer starts with engineered state and lag features plus a tree model, as in Section 16.5.7, and adds a recurrent model only after backtesting shows residual structure that depends on trajectories the features do not encode. Stating the specific diagnostic that would trigger the change, the six-hour thermal history example, is what distinguishes a considered answer from a preference.

Conceptual questions

"Why not just use yesterday?"
A strong answer agrees that yesterday is a baseline and insists that it be built, as level 1 of Table 16.2. It then gives the two reasons it is insufficient for this case: tomorrow's weather differs substantially from yesterday's, which is the whole point of a record hot day, and flexibility is not a property of the calendar but of the fleet's current thermal state, its availability and the action taken, none of which yesterday reproduces. The same answer concedes the converse obligation: the sophisticated model must demonstrate incremental value over this baseline, not merely exist.

The baseline and the data behind it. The next four go after the data the design is fitted on.

Conceptual questions

"Your baseline is unobservable once you dispatch."
The observation is correct and should be conceded immediately. A strong answer states that baseline estimation is part of the uncertainty problem rather than ground truth, that equation (16.12) means flexibility labels inherit the baseline method's bias, and that the first thing to establish before trusting any of them is how the historical event baselines were generated. It then gives the two remedies: randomized or held-out control groups, which make a contemporaneous baseline observable, and explicit propagation of baseline uncertainty into the flexibility distribution so that the optimizer sees it.

Conceptual questions

"What if half your telemetry disappears?"
A strong answer refuses silent imputation. The system enters a declared degraded mode: it switches to cohort and aggregate features that survive missing devices, uses the validated fallback of Figure 16.6, widens the uncertainty, and reduces the firm capacity it will support. It also treats the missingness itself as a signal rather than a nuisance, because a telemetry outage correlates with the conditions that cause it and with the devices it hides, so a sudden drop in completeness is information about the fleet and not only about the pipeline.

Conceptual questions

"How do you know your 90 % interval is 90 %?"
Only by measuring it. A strong answer describes empirical coverage over rolling backtests, as in Chapter 9: count how often the outcome fell inside the stated interval, and do it not once but by regime, since aggregate calibration hides local overconfidence. The regimes to report are the slices of Section 16.5.4, extreme heat, long events, each horizon and degraded availability, and the answer should say explicitly that a well-calibrated average with an overconfident hot-day slice is the failure mode this case would actually suffer.

Conceptual questions

"What if your model is biased by only 3 MW?"
A strong answer refuses to treat 3 MW as small without asking two questions: in which direction, and used for what. A persistent +3 MW bias in forecast flexibility means systematic overcommitment by 3 MW on every event, which is an accumulating settlement and reputational exposure rather than a rounding error. The same magnitude in the opposite direction merely leaves revenue on the table. And a directional bias matters most near a threshold: a bias that pushes the estimate across the 75 MW commitment line changes the decision, where the same bias in the middle of a comfortable range changes nothing.

Operating outside the validated domain. Two objections are about the record temperature and about trust.

Conceptual questions

"Tomorrow is hotter than anything in training."
A strong answer agrees and says what it does about it, following Section 16.4.1: flag the forecast as outside the validated domain, compare models that extrapolate differently and bring in physical limits, increase the stated uncertainty beyond what in-domain residuals imply, restrict the control action to offsets the response has actually been observed at, and operate conservatively until the first measured response at 16:05 supplies evidence from the new regime.

Conceptual questions

"Why should the optimizer trust your forecast?"
It should not trust a number because a model produced it. A strong answer says that trust is conditional and must be earned field by field: the forecast ships with validated uncertainty, data-quality indicators, a model version and out-of-distribution diagnostics, which is the contract of Table 16.5. The optimizer uses the forecast where calibration has been demonstrated, and falls back to conservative logic where it has not. A forecast that cannot say how much to believe it is not usable in a decision system.

Which number to optimize on, and how much to dispatch. Three objections are about the choice between the center of the distribution and its tail, and about how much of the fleet to call.

Conceptual questions

"Would you optimize on P50?"
Not necessarily, and the choice is not a convention. A strong answer derives it from the operational loss: if underdelivery is expensive relative to the comfort cost of calling extra devices, the right object is a lower quantile or the joint chance constraint of equation (16.19), not the median. If the costs are symmetric the median is defensible. The quantile to optimize on comes out of the loss function, and a system that always uses P50 has not asked what the loss function is.

Conceptual questions

"Your model says 100 MW. Why offer only 75?"
Because expected capacity is not firm capacity. A strong answer gives the four components of the reserve from Section 16.4.4, forecast uncertainty, correlated response, device dropout and decay over the block, and then points at the trajectory: the offer is set by the weakest interval of the window at the required reliability, which Figure 16.1 shows is the fourth hour and not the first.

Conceptual questions

"Why not call every thermostat?"
Because it maximizes the wrong quantity. A strong answer notes that calling everything does maximize the immediate response, and then lists what it costs: customer impact across the whole enrolled base, headroom consumed everywhere at once so that nothing is left for the fourth hour, a higher override rate, no reserve for the 16:05 revision, and a larger rebound after 20:00. The alternative is an optimized subset with reserve retained, which is what equation (16.19) produces when the comfort and rebound costs are in the objective.

Decay, lineage and the downstream contract. The last four are about operating the system over time.

Conceptual questions

"How would you handle response decay?"
By forecasting the controlled trajectory explicitly rather than a capacity figure. A strong answer gives the target of equation (16.3), names the features that let a model reproduce the decline, recent dispatch history and the thermal state of the cohort, and states the physical principle underneath it: current curtailment consumes future flexibility, so the trajectory is a depletion curve and not a noisy constant.

Conceptual questions

"How do you separate forecast error from dispatch failure?"
With lineage. A strong answer describes the full record kept for every event: the forecast that was issued and its version, the control instruction that was chosen, the command as it was delivered to each cohort, the acknowledgments and participation per device, and the measured response. With that chain, a 13 MW gap can be attributed to devices that never received the command rather than to a model that predicted them wrongly, and the two have completely different fixes. Without it, every operational failure is scored against the forecaster.

Conceptual questions

"What does the forecast service send downstream?"
The contract of Table 16.5, not a number. A strong answer lists the expected response or its quantiles by horizon, the scenario set that preserves temporal and cohort correlation, the data-quality and model-health indicators, and the forecast and model identifiers that make the output traceable. It adds the constraint that makes the format non-negotiable: whatever is sent must preserve the temporal correlation, because an optimizer cannot reconstruct it from per-hour quantiles.

Conceptual questions

"What would you do after the event?"
Run a post-mortem that feeds back into the system. A strong answer compares predicted and observed response interval by interval, checks the baseline against the randomized holdout of Section 16.2.6, checks the rebound against its forecast, and diagnoses the difference by source using the lineage record. It then says what changes: the calibration is updated with the new coverage evidence, the clean event data enters the training set, and the question is asked whether the event exposed a regime the models had never seen, in which case the validated domain of Section 16.4.1 has just been enlarged and the shift diagnostics must be retuned.

16.6.2 The end-to-end architecture

Figure 16.7 assembles the nine components into the system that runs the case. The top row is the forward path. Weather forecasts and thermostat telemetry enter on the left; the data-quality stage measures completeness and staleness and emits the flags of Table 16.5; the state and feature stage produces the estimated indoor temperatures, duty cycles, availability and headroom per cohort; the two model boxes are the baseline $P^{\text{base}}(t)$ and the response model $p(F \mid X_t, u)$ of Figure 16.2; their output is combined into the probabilistic flexibility, which is packaged with the out-of-distribution and quality indicators as a scenario set; and the optimizer consumes that package and emits a dispatch. The bottom row is the return path: the dispatch produces an actual response, which is measured, which updates the state, which triggers a reforecast, which re-enters the optimizer. The loop is the design. A system that implements only the top row is a forecasting service; a system that implements both is a decision system, and the difference shows up at 16:05.

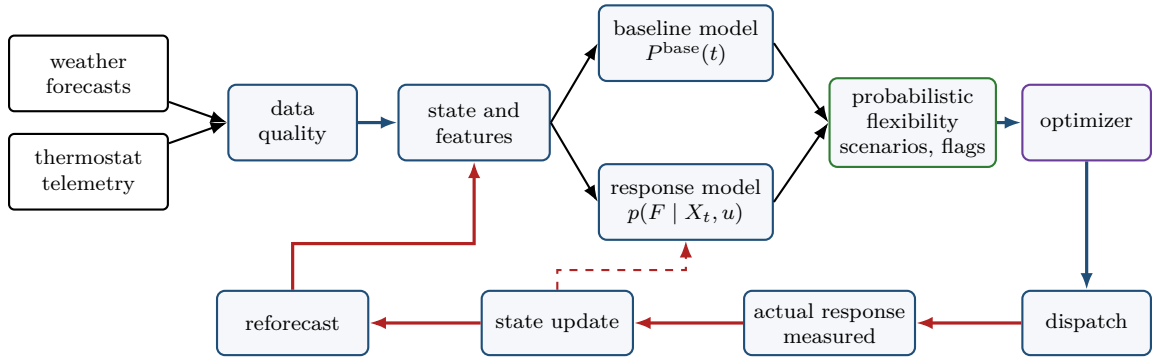


Figure 16.7: The end-to-end architecture. Top row, left to right: the forward path from inputs to dispatch. Bottom row, right to left: the return path from the measured response back into the state estimate and the response model. The nine components of this chapter are the boxes; the loop is what makes them a system.

16.6.3 The mathematical summary

The same system in symbols is short. Let x_t be the fleet state, collecting indoor temperatures, headroom per cohort, availability and device modes; let w_t be the exogenous disturbance, chiefly weather and occupancy; and let u_t be the control. The state evolves, and the response is a function of state and control,

$$x_{t+1} = g(x_t, u_t, w_t), \qquad F_t = h(x_t, u_t). \tag{16.26}$$

The forecasting system's deliverable is the conditional law of the response trajectory given the information set and a candidate control plan,

$$p(F_{t+1:t+H} \mid \mathcal{I}_t,\ u_{t:t+H}), \tag{16.27}$$

and the decision is taken by solving

$$u^* = \arg\min_{u \in \mathcal{U}} \ \mathbb{E}[C(u, F)] \quad \text{subject to} \quad \Pr(F_t \geq D_t\ \forall t) \geq 1 - \alpha \tag{16.28}$$

together with the comfort and physical constraints that define $\mathcal{U}$. The first element of u^* is executed, a measurement arrives, the state belief is updated, and equations (16.27) and (16.28) are solved again.

Figure 16.8 draws that cycle as five boxes. The boxes are, clockwise from the left: the state x_t with its belief; the dynamics g driven by the disturbance w_t; the forecast (16.27); the decision (16.28), which emits the control u_t; and the observation, which returns a measurement and closes the cycle through the posterior update. The arrows carry the symbols that pass between stages, so the figure can be read as a dependency graph of equations (16.26)–(16.28). The comparison worth making is with Figure 16.7: the engineering diagram has nine boxes because data pipelines, quality gates and fallbacks are real work, and the mathematical diagram has five because that is how many distinct objects there are.

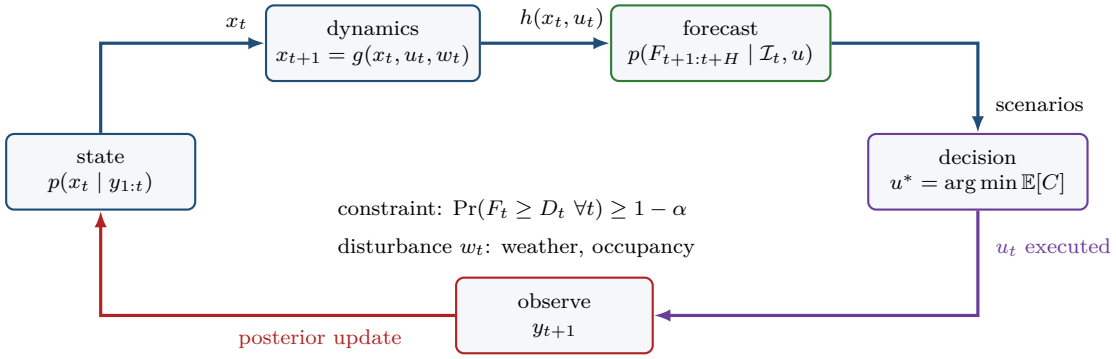


Figure 16.8: The decision loop of equations (16.26)–(16.28) as five objects. Clockwise from the left: the belief over the state, the dynamics that propagate it, the forecast of the response trajectory under a candidate control, the decision that selects a control, and the observation that closes the loop through a posterior update.

16.6.4 The capstone answer

Asked to design this system in the space of a few minutes, the answer that covers the case has eight moves, in this order.

Define the target as a multi-hour controlled-response trajectory, not a capacity figure: equation (16.3), over the window 16:00 to 20:00, because the request is for sustained delivery.

Separate the no-control baseline from the response under dispatch, and say that their difference is a counterfactual that is never observed: equation (16.2), with randomized holdouts as the way to make the first term measurable.

Build the baseline from fleet-level seasonal and weather-adjusted rules first, then gradient boosting on current load, lags, weather forecasts, availability, duty cycle, the indoor-setpoint gap, fleet composition and control history: Table 16.2 and equation (16.6). Test sequence models against that incumbent rather than assuming them.

Report probabilistically, as joint scenarios that preserve temporal and cohort correlation: equations (16.15) and (16.16), because equation (16.17) shows what per-hour marginals cost.

Let the optimizer enforce a delivery probability, or minimize expected underdelivery plus customer cost, while preserving headroom and accounting for rebound: equations (16.19) and (16.20), with the staged control of equation (16.21).

Flag the distribution shift explicitly, widen the uncertainty, and restrict actions to the region of the control space where the response has been observed: 42°C against a training maximum of 38°C, and ΔT_{set} within the range historically dispatched.

Operate the system closed-loop: the observed response at 16:05 updates the state, the forecast is revised, and additional cohorts are called from the reserve if the revised trajectory no longer sustains the commitment.

Validate with rolling event backtests, calibration by regime, sustained-delivery metrics and operational outcomes: Table 16.4, with the hot-day slice reported separately because that is the regime the system is used in.

16.7 What to retain

Definition 16.1 (A forecasting system for a dispatch decision)**.** The object such a system produces is not a forecast of historical load continued forward. It maps the current fleet state, the forecast weather and a candidate control action to a distribution over the future controlled response, $p(F_{t+1:t+H} \mid \mathcal{I}_t, u_{t:t+H})$, and hands that distribution to an optimizer that chooses the action. The system is successful only if it helps produce reliable decisions, which means that its target is a trajectory, its output is a joint distribution with quality and domain flags attached, and its loop closes on measured response.

Table 16.6 collects the nine components, the subsection of this chapter that specifies each, and the earlier chapter whose material it draws on. The table is also a reading of the book: the chapters were written in the order of the right-hand column, and the case needs all of them simultaneously.

Table 16.6: The nine components of the system, where this chapter specifies each, and the chapters they draw on.

Component	What it produces	Section	Draws on
1. Baseline forecast	$P^{\text{base}}(t)$ over the window, with a hierarchy behind it	16.2.1	Ch. 2, 3, 6, 5
2. Response model	$p(F \mid X_t, u)$, conditional on the action	16.2.3	Ch. 4, 7, 8, 12
3. Fleet representation	cohort responses summing to the fleet total	16.3.1	Ch. 11
4. Probabilistic output	quantiles and joint scenarios over the window	16.3.4	Ch. 10
5. Shift handling	the out-of-distribution flag and its consequences	16.4.1	Ch. 13
6. Optimization	u^* under a sustained-delivery constraint	16.4.2	Ch. 14
7. Closed-loop updating	revised forecasts at 15:00, 15:45 and 16:05	16.5.1	Ch. 1, 14
8. Validation plan	backtest scores by dimension and by slice	16.5.4	Ch. 9
9. Failure handling	tiered fallbacks and the degradation flag	16.5.5	Ch. 15

16.8 Exercises

Exercise 16.1. Take the decay trajectory of equation (16.4), $[90, 82, 68, 52]$ MW. (a) What is the largest commitment D that is met in every hour of the window, what is the largest met on the four-hour average, and in which hours does the expectation constraint of equation (16.18) hold at $D_t = 75$ MW? (b) By how much must each hour be increased for $D = 75$ MW to be met throughout, and what is the total additional energy in megawatt-hours? (c) Suppose the system instead reports $\Pr(F_t \geq 75\,\text{MW}) = 0.95$ for each of the four hours. Compute the joint probability under independence, equation (16.17), and state which direction positive correlation between the hours moves it. (d) Explain why the answer to (c) cannot be obtained from the four marginal probabilities alone.

Exercise 16.2. Use the three scenarios of equation (16.16). (a) For each scenario, list the hours in which the 75 MW request is met. (b) Treating the three as equally likely, estimate Pr(sustain 75 MW) and the per-hour exceedance probability for each of the four hours; comment on the gap between the first-hour figure and the sustained figure. (c) Fit a normal distribution to the first-hour quantiles of equation (16.15) using P50 and P90, compute the implied P10, and compare it with the reported 68 MW; what does the discrepancy say about the shape of the distribution? (d) Using that fitted normal, compute $\Pr(F_{16} \geq 75\,\text{MW})$.

Exercise 16.3. Cohort A can deliver 40 MW at once and its delivery decays at 20 % per hour while it is held; cohort B can deliver 30 MW and decays at 10 % per hour; cohort C holds a further 30 MW in reserve. (a) Tabulate the four-hour delivered trajectory for the strategy that calls A and B together at 16:00, and for the strategy that calls B at 16:00 and adds A at 17:00, assuming a cohort's decay starts when it is called. (b) For each strategy, name the first hour in which delivery falls below 75 MW. (c) How much fresh capacity must be called at 17:00 in the first strategy to restore 75 MW in that hour? (d) Using Figure 16.4, explain why calling A first makes the answer to (c) larger than it would otherwise be.

Exercise 16.4. Take the contract of Table 16.5: median 84 MW, P10 69 MW, P90 98 MW, and Pr(sustain 75 MW) = 0.72. (a) Fit a normal to the P10 and P90 pair and compute its standard deviation. (b) Using that normal with mean 84 MW, compute the first-hour exceedance probability $\Pr(F_{16} \geq 75\,\text{MW})$, and explain why the reported sustained probability of 0.72 is lower. (c) What offered capacity D would make the first-hour probability 95 %, and why is the sustainable offer lower still? (d) List the five responses available in Section 16.5.6 and state which of them change the probability and which change only the commitment.