Quantitative Research Methods

Chapter 6: Linear Model Selection and Regularisation

Prof. Dr. Christoph Weisser

HSBI

Summer Semester 2026

The course at a glance

	Lecture	Topic
	1	Ch. 1 + 2 — Introduction; what is statistical learning
	2	Ch. 2 — Accuracy, bias–variance, Bayes classifier, KNN
	3–4	Ch. 3 — Linear regression (estimation, inference, diagnostics)
	5–6	Ch. 4 — Classification (logistic, LDA/QDA, naive Bayes, ROC)
	7	Ch. 5 — Resampling (validation, CV, bootstrap)
▶	**8**	**Ch. 6 — Model selection and regularisation (ridge, lasso)**
	9	Ch. 7 — Moving beyond linearity (splines, GAMs)
	10	Ch. 8 — Tree-based methods (trees, forests, boosting)
	11	Ch. 10 — Deep learning (MLPs, CNNs, training)
	12	Ch. 13 — Multiple testing (FWER, FDR)

Twelve sessions of 180 minutes. ▶ marks today's chapter. Short exercises every ∼20 min; extended exercises every ∼45 min; each chapter has a companion lab notebook.

Contents

Optional and advanced material is collected in the **appendix** at the end of the deck.

Notation in this chapter

Symbol	Meaning
d	Number of predictors in a candidate model (2^p subsets in total)
RSS, TSS	Residual / total sum of squares
$\hat{\sigma}^2$	Estimated noise variance
C_p, AIC	$\frac{1}{n}(\mathrm{RSS} + 2d\hat{\sigma}^2)$ — lighter complexity penalty
BIC	$\frac{1}{n}(\mathrm{RSS} + \log(n)\, d\hat{\sigma}^2)$ — heavier penalty
adj. R^2	R^2 corrected for the model size d
λ	Tuning parameter: penalty strength, chosen by CV
$\lambda \sum_j \beta_j^2$	Ridge (ℓ_2) penalty: shrinks, never exactly zero
$\lambda \sum_j \lvert\beta_j\rvert$	Lasso (ℓ_1) penalty: shrinks and selects
Z_m, ϕ_{jm}	Derived components and their loadings
M	Number of components used in PCR (principal components regression) / PLS (partial least squares)

Sources and references

Primary textbook

James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). *An Introduction to Statistical Learning, with Applications in Python.* Springer.

About these slides

Each method is introduced with motivation, intuition, formal definition, and worked example. Slide titles are descriptive.

Learning objectives for this chapter

By the end of this chapter you will be able to:

- **Apply** best-subset and stepwise selection; **choose** among candidate models with C_p, AIC, BIC, adjusted R^2, or cross-validation.
- **Fit** ridge and lasso regression and **interpret** their coefficient paths.
- **Articulate** the geometric reason the lasso produces sparsity while ridge does not.
- **Use** principal-components and partial-least-squares regression for highly collinear or high-dimensional designs.
- **Reason** about the high-dimensional regime ($p > n$) where OLS breaks down.

Roadmap of this chapter

Three families that fix OLS's problems

1. **Subset selection**: best subset / forward / backward.
2. **Shrinkage**: ridge (ℓ_2), lasso (ℓ_1), elastic net.
3. **Dimension reduction**: PCR (unsupervised), PLS (supervised).

Cross-validation (Ch. 5) is used throughout to choose tuning parameters honestly.

Outline

Three pressures that push us beyond OLS

Plain ordinary least squares is excellent in the classical regime ($n \gg p$, predictors weakly correlated). In modern applications three pressures push us further.

The three motivations

1. **Prediction accuracy.** OLS is unbiased but can have very high variance, especially when $p \approx n$.
2. **Interpretability.** With many predictors, only a few usually matter. We want a model that highlights them.
3. **High dimensions.** When $p > n$, OLS is undefined — $\mathrm{X}^\top \mathrm{X}$ is singular.

Three families of solutions

Subset selection

Pick a subset of predictors. Combinatorial.

Best subset, forward, backward stepwise.

Shrinkage

Keep all predictors, shrink their coefficients.

Ridge, lasso, elastic net.

Dimension reduction

Construct fewer linear combinations of the original predictors.

PCR, PLS.

Where this chapter is used in industry

Sector	Concrete application	Which decision it drives
Banking	Attribute shortlisting for a credit scorecard	Which attributes ship on the scorecard
Pharma, food, chemicals	PLS on near-infrared spectra (hundreds of wavelengths)	Release the batch, or send it to the lab
Manufacturing, energy	Ridge on collinear sensor and telemetry channels	Which process settings to trust
Marketing	Mix model where channel spends move together	A defensible budget reallocation
Health, biotech	Lasso screening of gene-expression features	Which features get a follow-up study
Quantitative finance	Sparse selection among candidate return predictors	Which signals get capital

In industry — The common thread

Wide, correlated data plus a business need for a model small enough to explain — that is what regularisation delivers.

Industry case in depth: near-infrared spectroscopy and PLS

In industry — The setting

A plant needs the active-ingredient concentration of a blend, and the reference method is a slow destructive lab assay. **Response:** the reference value (% w/w). **Predictors:** absorbance at several hundred NIR wavelengths, adjacent ones nearly identical, on calibration sets of only tens of samples ($p \gg n$). **Output:** a PLS fit on a few components (M by CV), scored by root mean squared error of prediction.

Which decision it feeds and who signs it off

The calibration replaces routine lab testing: it feeds in-process control and real-time release of a batch. QA and regulatory affairs own the sign-off, because it forms part of a *validated* analytical method. The predicted concentration itself reaches a QA release manager or production supervisor, who must release the batch — or hold it for the reference assay — on the strength of it.

The honest caveat

Calibrations drift — a new supplier or instrument moves the spectra, so the model must be revalidated. And PLS mixes *all* wavelengths: it predicts without pointing at a cause.

Outline

Best subset selection: idea

The most direct way to find a small, interpretable linear model.

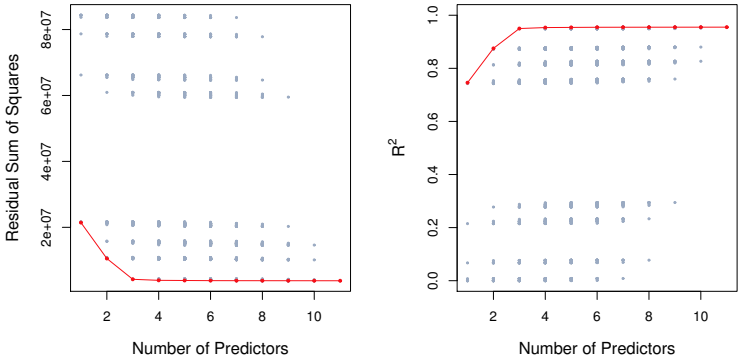
Algorithm

1. For each $k = 0, 1, \ldots, p$, find the model with k predictors that minimises RSS over all $\binom{p}{k}$ possibilities.
2. Among the $p + 1$ "best per-size" models, choose by cross-validation, C_p, AIC, BIC, or adjusted R^2.

Computational cost

Searching $\binom{p}{k}$ subsets at each k is feasible only for $p \lesssim 30$. Beyond that we use stepwise heuristics.

Best subset on the Credit data



RSS and R^2 as k varies: both improve monotonically with more predictors, so they cannot by themselves pick k — that is what C_p, BIC and adjusted R^2 (next) are for.

Source: Figure 6.1 — ISLP, James et al. (2023).

Forward stepwise selection

A greedy alternative when best subset is too expensive.

Algorithm

1. Start with the null model M_0 (only intercept).
2. For $k = 1, \ldots, p$: add to M_{k-1} the predictor that most reduces RSS. Call the result M_k.
3. Choose among $M_0, \ldots, M_p$ by CV / C_p / BIC.

How to read this

Considers $1 + \sum_k (p - k) = O(p^2)$ models instead of 2^p — scales to large p. Greedy: not guaranteed to find the best subset.

In industry — Banking: from a thousand candidate attributes to twelve

Bureau files offer hundreds of candidate attributes; the scorecard that ships has perhaps 10–15. Parsimony here is a monitoring and governance requirement, not merely statistical taste.

Backward and hybrid stepwise

Backward

Start with the full model and drop the least useful predictor at each step. Requires $n > p$.

Hybrid

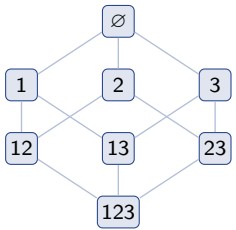
Alternate forward and backward steps. Recovers some of the flexibility of best subset at greedy cost.

All three are heuristics

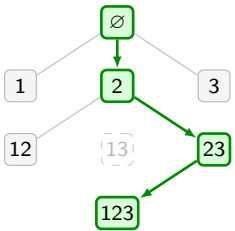
Forward, backward, and hybrid can disagree on the same data. Always cross-validate the chosen model.

Best subset vs. forward stepwise: two search strategies

Best subset: evaluate *all* 2^p



Forward: follow one greedy path



Takeaway

Best subset scores the *whole* lattice (2^p models); forward keeps only the winner at each level (the bold subsets on the arrowed path) and never even fits the dashed subset. At $p = 20$: $2^{20} \approx 10^6$ vs. 211 models.

Choosing k: four penalised criteria

Training RSS always decreases with k, so we need penalised criteria that approximate test error.

The four standard criteria

$$
\begin{aligned}
C_p &= \tfrac{1}{n}\big(\mathrm{RSS} + 2d\hat{\sigma}^2\big), \\
\mathrm{AIC} &\propto C_p, \\
\mathrm{BIC} &= \tfrac{1}{n}\big(\mathrm{RSS} + \log(n)\, d\hat{\sigma}^2\big), \\
\text{adj } R^2 &= 1 - \tfrac{\mathrm{RSS}/(n-d-1)}{\mathrm{TSS}/(n-1)}.
\end{aligned}
$$

Takeaway

All four share one shape — *training fit* + *complexity penalty*; they differ only in how steeply each extra predictor is taxed.

Reading the criteria symbol-by-symbol

How to read this

Symbol	Meaning
RSS	Residual sum of squares (training fit)
d	Number of predictors in the model
$\hat{\sigma}^2$	Estimated noise variance
n	Sample size
TSS	Total sum of squares

Practical guidance

BIC penalises more heavily for large $n \Rightarrow$ smaller models. Use BIC when parsimony matters most. Cross-validation is more robust when distributional assumptions are uncertain.

Exercise 6.2 — Why RSS and R^2 Cannot Pick the Size [Concept]

Exercise

1. Explain why training RSS and training R^2 can never be used to choose the *number* of predictors d.
2. What quantity do C_p, AIC, and BIC try to estimate that training RSS does not?
3. State qualitatively how C_p, BIC, and adjusted R^2 modify the raw training fit, and which of them most favours small models.

Solution — Exercise 6.2

Solution

(1) Monotonicity. Adding a predictor to a nested least-squares model can never increase RSS (the extra coefficient can always be set to zero), so RSS is *non-increasing* in d and $R^2 = 1 - \mathrm{RSS}/\mathrm{TSS}$ is *non-decreasing*. Both are therefore optimised by the largest model, giving no guidance on size and rewarding overfitting.
(2) What the criteria estimate. They estimate the *test* error (expected prediction error on new data), not the training error. Each adds a penalty that grows with d to counteract the optimism of the training fit.
(3) How they adjust the fit.

- $C_p = \frac{1}{n}(\mathrm{RSS} + 2d\hat{\sigma}^2)$: adds a penalty $2d\hat{\sigma}^2$ (AIC is proportional to it).
- $\mathrm{BIC} = \frac{1}{n}(\mathrm{RSS} + \log(n)\, d\hat{\sigma}^2)$: same shape but penalty $\log(n)\, d\hat{\sigma}^2$. Since $\log n > 2$ for $n > 7$, BIC penalises extra predictors more heavily and therefore *most favours small (parsimonious) models*.
- Adjusted $R^2 = 1 - \frac{\mathrm{RSS}/(n-d-1)}{\mathrm{TSS}/(n-1)}$: divides RSS by its degrees of freedom, so it can *decrease* when a useless predictor is added.

We pick the model that minimises C_p/AIC/BIC or maximises adjusted R^2.

Common mistake: comparing models of different sizes by training RSS or R^2 — the biggest model always wins by construction; that is exactly the problem these criteria solve.

Exercise 6.3 — Computing C_p and BIC [Math]

Exercise

With $n = 100$ observations and $\hat{\sigma}^2 = 4$, two candidate models give:

	d	RSS
Model A	3	480
Model B	6	450

Compute C_p and BIC for each model and state which model each criterion selects. Use $C_p = \frac{1}{n}(\mathrm{RSS} + 2d\hat{\sigma}^2)$ and $\mathrm{BIC} = \frac{1}{n}(\mathrm{RSS} + \log(n)\, d\hat{\sigma}^2)$.

Solution — Exercise 6.3 (1/2)

Solution

Here $n = 100$, $\hat{\sigma}^2 = 4$, and $\log(100) \approx 4.605$. Both criteria share the shape $\frac{1}{n}(\mathrm{RSS} + \text{price} \times d\hat{\sigma}^2)$; only the per-predictor price differs (2 for C_p vs. $\log n$ for BIC).
C_p (price 2 per predictor).

$$C_p^A = \tfrac{1}{100}(480 + 2 \cdot 3 \cdot 4) = \tfrac{1}{100}(480 + 24) = 5.04,$$
$$C_p^B = \tfrac{1}{100}(450 + 2 \cdot 6 \cdot 4) = \tfrac{1}{100}(450 + 48) = 4.98.$$

C_p is smaller for **Model B** ($4.98 < 5.04$), so C_p selects B: B's RSS saving of $480 - 450 = 30$ outweighs its extra penalty of $48 - 24 = 24$.

Solution — Exercise 6.3 (2/2)

Solution

BIC (price $\log(100) \approx 4.605$ per predictor).

$$\mathrm{BIC}^A = \tfrac{1}{100}\left(480 + 4.605 \cdot 3 \cdot 4\right) = \tfrac{1}{100}(480 + 55.3) = 5.353,$$
$$\mathrm{BIC}^B = \tfrac{1}{100}\left(450 + 4.605 \cdot 6 \cdot 4\right) = \tfrac{1}{100}(450 + 110.5) = 5.605.$$

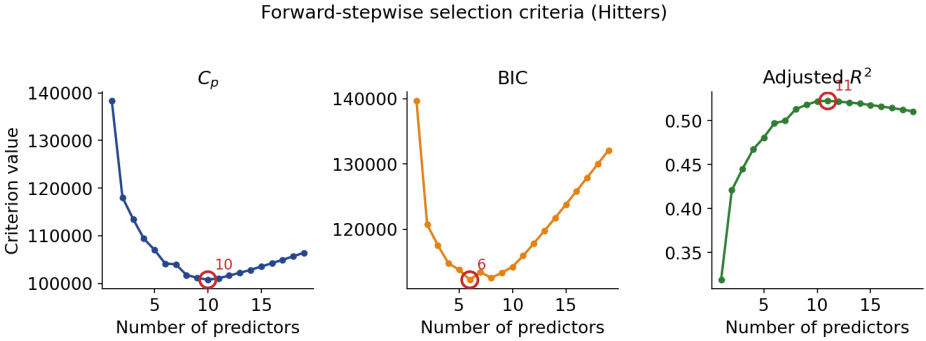
BIC is smaller for **Model A** ($5.353 < 5.605$), so BIC selects A: the same RSS saving of 30 must now beat an extra penalty of $110.5 - 55.3 = 55.2$ — and fails.

Interpretation. The two criteria disagree: BIC's heavier $\log(n)$ penalty punishes the three extra predictors of Model B more than they reduce RSS, so BIC prefers the smaller Model A, while the lighter C_p penalty lets Model B's better fit win. When parsimony matters, follow BIC.

Common mistake

Using $\log_{10}(100) = 2$ instead of the natural log $\ln(100) \approx 4.605$ — with base-10 logs the BIC penalty would collapse to C_p's $2d\hat{\sigma}^2$ here.

Choosing the model size: C_p, BIC and adjusted R^2



Takeaway

Each criterion trades goodness-of-fit against complexity; BIC's heavier penalty picks the smallest model, C_p and adjusted R^2 a slightly larger one.

Extended Exercise 6.1 — C_p, BIC and Adjusted R^2 Across Sizes [Math]

Extended exercise (15 min)

Best-subset selection on a problem with $n = 100$ and $p = 4$ returns the best model of each size, with residual sums of squares (and $\hat{\sigma}^2 = 5$ estimated from the full model):

d	0	1	2	3	4
RSS	900	500	340	322	314

1. Compute C_p, BIC and adjusted R^2 for each size, using $C_p = \frac{1}{n}(\mathrm{RSS} + 2d\hat{\sigma}^2)$, $\mathrm{BIC} = \frac{1}{n}(\mathrm{RSS} + \log(n)d\hat{\sigma}^2)$ and $\mathrm{adj}\, R^2 = 1 - \frac{\mathrm{RSS}/(n-d-1)}{\mathrm{TSS}/(n-1)}$ with $\mathrm{TSS} = \mathrm{RSS}_{d=0}$.
2. State the winner under each criterion and explain any disagreement.
3. How many models did best subset examine, and how many would forward stepwise fit?

Solution — Extended Exercise 6.1 (1/3)

Solution

With $n = 100$, $\hat{\sigma}^2 = 5$ and $\log(100) \approx 4.605$, the per-parameter penalties are $2\hat{\sigma}^2 = 10$ (C_p) and $\log(n)\hat{\sigma}^2 \approx 23.03$ (BIC). Dividing by $n = 100$:

d	0	1	2	3	4
C_p	9.00	5.10	3.60	3.52	3.54
BIC	9.00	5.23	3.86	3.91	4.06

For example $C_p(3) = \frac{1}{100}(322 + 2 \cdot 3 \cdot 5) = \frac{352}{100} = 3.52$ and $\mathrm{BIC}(2) = \frac{1}{100}(340 + 4.605 \cdot 2 \cdot 5) = \frac{386.05}{100} = 3.86$.

Solution — Extended Exercise 6.1 (2/3)

Solution

Adjusted R^2 with $\mathrm{TSS} = 900$ and $\mathrm{TSS}/(n-1) = 900/99 = 9.09$:

d	0	1	2	3	4
$\mathrm{RSS}/(n-d-1)$	9.09	5.10	3.51	3.35	3.31
$\mathrm{adj}\, R^2$	0.00	0.44	0.61	0.63	0.636

e.g. at $d = 4$: $1 - \frac{314/95}{900/99} = 1 - \frac{3.305}{9.091} = 0.636$.

Winners. $C_p \Rightarrow d = 3$; $\mathrm{BIC} \Rightarrow d = 2$; $\mathrm{adj}\, R^2 \Rightarrow d = 4$.

Common mistake

Dividing by $n - d$ instead of $n - d - 1$ in the numerator — the extra -1 is the intercept. At $d = 4$ the correct divisor is 95, not 96: $314/95 = 3.305$.

Solution — Extended Exercise 6.1 (3/3)

Solution

(2) Why they disagree. All three add a complexity penalty to the training fit, but with different strengths. BIC's penalty $\log(n) \approx 4.6$ per parameter is heaviest, so it stops earliest at the parsimonious $d = 2$. C_p/AIC use the lighter penalty 2, so they tolerate the small RSS drop $340 \to 322$ and pick $d = 3$. Adjusted R^2 penalises size only mildly and keeps improving to $d = 4$. Ordering by aggressiveness of shrinkage in model size, $\mathrm{BIC} \leq C_p \leq \mathrm{adj}\, R^2$ — exactly the pattern in Figure 6.2.

(3) Search cost. Best subset examines $2^p = 2^4 = 16$ models; forward stepwise fits $1 + \frac{p(p+1)}{2} = 1 + 10 = 11$. The gap is tiny here but explodes with p: at $p = 20$ it is $2^{20} \approx 10^6$ versus 211 — why best subset is infeasible past $p \approx 30$.

Outline

Why ridge: numerical stability via a penalty

Subset selection is discrete and combinatorial. A smoother alternative: keep all predictors but penalise large coefficients.

Intuition

The penalty pulls $\hat{\beta}$ toward zero, especially when the data don't strongly constrain it. We trade a small upward bias for a large reduction in variance.

How to read this

Two payoffs at once: the penalty (i) *stabilises* the fit when predictors are collinear, and (ii) *lowers variance* by refusing to chase noise with large coefficients.

In industry — Manufacturing and energy: collinear sensors

Process telemetry is full of channels that move together. Plain OLS then returns huge, sign-flipping coefficients that move with every data refresh; ridge returns a fit an engineer can act on.

The ridge regression objective

Objective

$$\hat{\beta}^{\mathrm{R}} = \arg\min_{\beta} \Bigg\{ \underbrace{\sum_{i=1}^{n} (y_i - \beta_0 - X_i^{\top}\beta)^2}_{\mathrm{RSS}} + \underbrace{\lambda \sum_{j=1}^{p} \beta_j^2}_{\ell_2 \text{ penalty}} \Bigg\}.$$

Takeaway

Minimise training error *plus* a size tax on the coefficients; the tuning parameter λ sets how hard we pull them toward zero.

Reading the ridge objective

How to read this

Term	Role
RSS	Training fit: pulls coefficients toward the OLS solution.
$\lambda \sum \beta_j^2$	Penalty: pulls coefficients toward zero.
λ	Tuning parameter that controls the trade-off.

- $\lambda = 0$ recovers ordinary least squares.
- $\lambda \to \infty$ drives every $\hat{\beta}^{\mathrm{R}} \to 0$.
- Closed form: $\hat{\beta}^{\mathrm{R}} = (\mathrm{X}^\top \mathrm{X} + \lambda I)^{-1} \mathrm{X}^\top \mathrm{y}$.

Three ways to think about ridge

Numerical stability

Adding λI stabilises the inverse when $\mathrm{X}^\top\mathrm{X}$ is near-singular (collinearity).

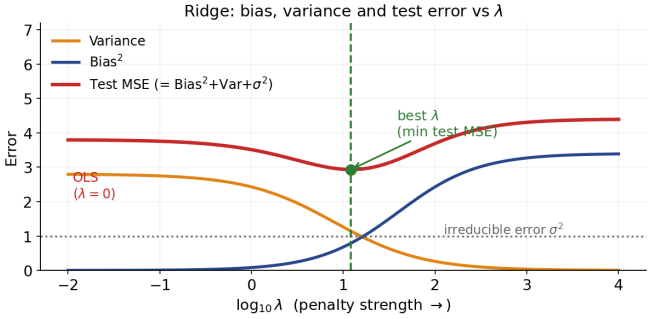
Bias-variance trade-off

Trades a small upward bias for a large reduction in variance. Especially helpful when p is large relative to n.

Standardise first!

The penalty is not scale-invariant. Standardise predictors before fitting, otherwise ridge will preferentially shrink variables with larger units.

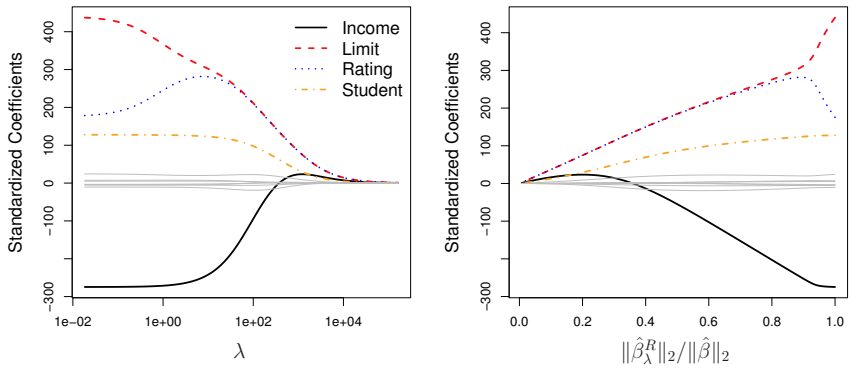
Ridge's bias-variance trade-off, visualised



Takeaway

As λ grows, variance falls but squared bias rises. Test error is their sum plus the irreducible σ^2, so it bottoms out at an *intermediate* λ — the one cross-validation aims to find.

Ridge: coefficient paths



Each line is one $\hat{\beta}_j$ as λ grows from left to right. Coefficients shrink smoothly toward zero — but never exactly reach zero.

Source: Figure 6.4 — ISLP, James et al. (2023).

Exercise 6.4 — Ridge: λ and Standardisation [Concept]

Exercise

1. Describe what happens to the ridge coefficients $\hat{\beta}^{\mathrm{R}}$ as λ increases from 0 to ∞, and to the model's bias and variance.
2. Why must predictors be standardised before ridge regression? What goes wrong if one predictor is measured in metres and another in millimetres?
3. Does ridge ever set a coefficient exactly to zero? What does this imply for interpretability?

Solution — Exercise 6.4

Solution

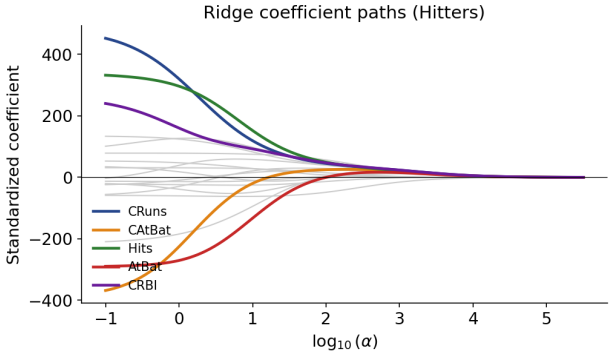
(1) Effect of λ. At $\lambda = 0$ ridge equals OLS. As λ grows the ℓ_2 penalty pulls all coefficients smoothly toward zero, and as $\lambda \to \infty$ every $\hat{\beta}_j^{\mathrm{R}} \to 0$ (only the intercept survives). Increasing λ *increases bias* (the fit is pulled away from the data) but *decreases variance* (the coefficients become more stable). The best λ, chosen by cross-validation, trades a little bias for a large variance reduction.

(2) Why standardise. The penalty $\lambda \sum_j \beta_j^2$ is *not scale invariant*. A predictor measured in metres has a coefficient $1000\times$ larger than the same predictor in millimetres, so it would be penalised 10^6 times more heavily and shrunk far more aggressively — an arbitrary artefact of units, not of importance. Standardising each predictor to unit variance (typically $\tilde{x}_{ij} = x_{ij}/\hat{\sigma}_j$) puts them on a common footing so the penalty treats them comparably.

(3) No exact zeros. Ridge shrinks coefficients toward zero but essentially never sets them *exactly* to zero (the ℓ_2 penalty is smooth at the origin). So ridge keeps all p predictors; it improves prediction and stability but does *not* select variables — for that, use the lasso.

Common mistake: treating standardisation as cosmetic. For plain OLS it is (predictions unchanged); with a penalty it *changes the model* — ridge on raw and on standardised predictors gives genuinely different fits.

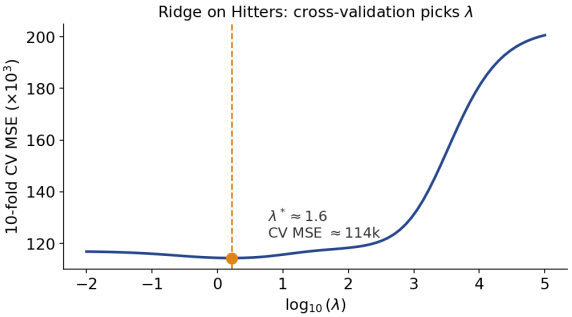
Ridge coefficient paths on Hitters



Takeaway

As λ grows every coefficient shrinks smoothly toward zero, but none is set *exactly* to zero — ridge keeps all predictors.

Choosing λ by cross-validation on Hitters



Takeaway

The 10-fold CV curve on standardised Hitters dips at $\lambda^* \approx 1.6$ (CV MSE $\approx$ 114,000) — mild shrinkage beats OLS, and the error climbs steeply once heavy shrinkage drowns the signal.

Outline

The lasso: change the penalty, change the behaviour

Same idea as ridge but with an ℓ_1 penalty instead of ℓ_2.

Objective

$$\hat{\beta}^{\mathrm{L}} = \arg\min_{\beta} \left\{ \mathrm{RSS} + \lambda \sum_{j=1}^{p} |\beta_j| \right\}.$$

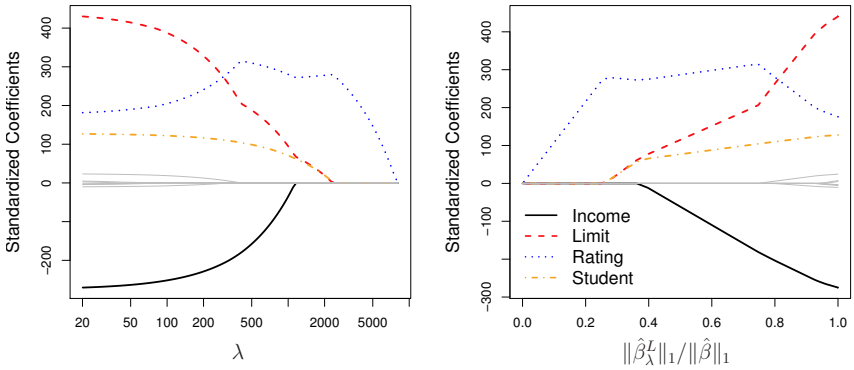
How to read this

The *only* change from ridge is $|\beta_j|$ in place of β_j^2 — and that kink at the origin is exactly what lets coefficients reach zero.

The crucial property

For λ large enough, some $\hat{\beta}_j^{\mathrm{L}} = 0$ *exactly*. The lasso *performs variable selection* automatically as a side effect of fitting.

Lasso: coefficient paths



As λ grows, coefficients hit zero one by one. The lasso delivers a continuum of sparse models indexed by λ.

Source: Figure 6.6 — ISLP, James et al. (2023).

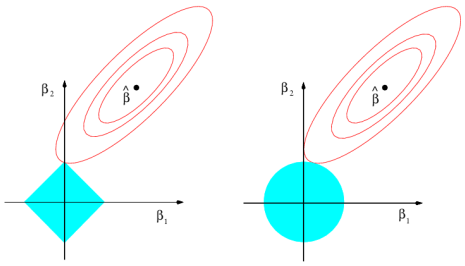
Why does lasso produce zeros? — the geometry

The unit balls

- Ridge constrains $\sum \beta_j^2 \leq t$ — a smooth ball.
- Lasso constrains $\sum |\beta_j| \leq t$ — a polytope with corners on the axes.

Takeaway

The level sets of the squared-error loss are likely to first touch the lasso ball at a corner — a corner has some $\beta_j = 0$.



Source: Figure 6.7 — ISLP, James et al. (2023).

Choose λ by cross-validation

Both ridge and lasso introduce a single tuning parameter.

The procedure

1. Run k-fold CV over a logarithmic grid of λ values (say, 50 from 10^{-3} to 10^{3}).
2. Pick the λ that minimises CV error, or apply the *one-standard-error rule*: the largest λ within one standard error of the minimum.
3. Re-fit on the full data at that λ.

How to read this

In Python: `sklearn.linear_model.RidgeCV` and `LassoCV` perform the grid search internally.

In industry — Retail analytics: wide SKU and customer tables

With hundreds of engineered columns, λ — not the coefficient list — is what you defend.

Ridge vs. lasso vs. elastic net: when each wins

Ridge	Lasso	Elastic net
Many small effects, none dominant. Predictors all somewhat useful.	Truth is sparse: a few predictors carry the signal, the rest are noise.	$\alpha\|\beta\|_1 + (1-\alpha)\|\beta\|_2^2$. Useful for sparse + correlated predictors.

Default workflow

Try lasso first. If you see correlated predictors splitting their estimated effects, fall back to elastic net or ridge.

In industry — Quantitative finance: is the selected set stable?

With many candidate predictors and a tiny signal-to-noise ratio, shrinkage beats unconstrained least squares. But the lasso's chosen *set* is unstable under correlation — so practitioners re-run it over bootstrap samples and only believe a variable that survives most of them.

Mini-check: ridge vs. lasso

Quick self-assessment

For each scenario, which method is more appropriate?

1. Many predictors, all expected to have small effects.
2. Truth is sparse: only a handful of predictors matter.
3. Severe multicollinearity, want a stable fit.
4. Want automatic variable selection.

Answers

1. Ridge. 2. Lasso. 3. Ridge. 4. Lasso (or elastic net).

Exercise 6.5 — Lasso Sparsity: ℓ_1 vs. ℓ_2 [Concept/Math]

Exercise

1. Write the constrained (budget) form of ridge and lasso and describe the shape of each constraint region in two dimensions.
2. Use that geometry to explain why the lasso produces exact zeros while ridge does not.
3. *Orthonormal design.* When $\mathbf{X}^\top \mathbf{X} = \mathbf{I}$, the lasso has the closed-form soft-threshold solution $\hat{\beta}_j^{\mathrm{L}} = \operatorname{sign}(\hat{\beta}_j^{\mathrm{OLS}})\big(|\hat{\beta}_j^{\mathrm{OLS}}| - \lambda/2\big)_+$. If $\hat{\beta}_j^{\mathrm{OLS}} = 1.5$ and $\lambda = 4$, compute $\hat{\beta}_j^{\mathrm{L}}$.

Solution — Exercise 6.5 (1/2)

Solution

(1) Constraint regions. Both methods can be written as minimise RSS subject to a budget:

$$\text{ridge: } \sum_j \beta_j^2 \le t, \qquad \text{lasso: } \sum_j |\beta_j| \le t.$$

In two dimensions the ridge region is a *disc* (smooth, rounded), while the lasso region is a *diamond* (a square rotated 45°) with sharp corners lying on the coordinate axes.

(2) Why lasso gives zeros. The elliptical contours of the RSS expand until they first touch the constraint region; the solution is that contact point. The lasso diamond has corners *on the axes*, where one coordinate is exactly zero, and contours are very likely to hit a corner first — yielding $\hat{\beta}_j = 0$. The ridge disc has no corners, so the contact point almost never lands exactly on an axis; ridge shrinks but does not zero out. This is why the ℓ_1 penalty performs variable selection and the ℓ_2 penalty does not.

Solution — Exercise 6.5 (2/2)

Solution

(3) Soft-thresholding. With $\hat{\beta}_j^{\mathrm{OLS}} = 1.5$ and $\lambda = 4$:

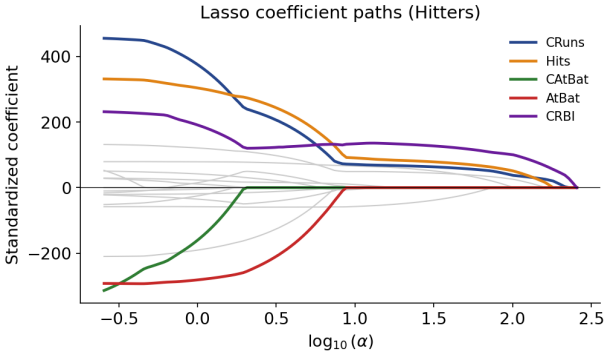
$$|\hat{\beta}_j^{\mathrm{OLS}}| - \lambda/2 = 1.5 - 2.0 = -0.5 < 0,$$

so $(\,\cdot\,)_+ = 0$ and $\hat{\beta}_j^{\mathrm{L}} = 0$: the predictor is dropped. Had $\lambda = 2$ instead, we would get $\mathrm{sign}(1.5)(1.5 - 1.0) = +0.5$ — shrunk toward zero but retained. This shows algebraically how the lasso zeros out coefficients once the penalty exceeds twice the OLS magnitude, while the ridge solution $\hat{\beta}_j/(1+\lambda)$ would only shrink, never reaching zero.

Common mistake

Thresholding at λ instead of $\lambda/2$. Here both give zero, but at $\lambda = 2$ the wrong threshold yields $1.5 - 2 < 0 \Rightarrow 0$, killing a coefficient that actually survives as $1.5 - 1.0 = 0.5$.

Lasso coefficient paths on Hitters



Takeaway

Unlike ridge, the lasso drives coefficients to *exactly* zero as λ grows — it performs automatic variable selection.

Outline

Dimension reduction: idea

Construct $M < p$ new predictors $Z_1, \ldots, Z_M$ as linear combinations of the original X_j, then fit linear regression on these.

Common form

$$Z_m = \sum_{j=1}^{p} \phi_{jm} X_j, \quad Y \approx \theta_0 + \theta_1 Z_1 + \cdots + \theta_M Z_M.$$

Two flavours

- **PCR**: the ϕ_{jm} come from PCA on X alone. Unsupervised.
- **PLS**: the ϕ_{jm} are chosen to maximise covariance with Y. Supervised.

Principal components regression (PCR)

Recipe

1. Compute the principal components of X (after standardisation).
2. Use the first M components as predictors in a linear regression on Y.

Choose M by cross-validation.

Caveat

PCR ignores Y when forming the components. The leading components capture maximal X-variance, but the most predictive direction for Y may lie in lower-variance components.

Outline

The high-dimensional regime

Modern data sets routinely have p in the thousands (genomics, text) or millions (images). When $p > n$ classical methods break down.

What goes wrong with OLS

- $X^\top X$ is singular — OLS is undefined.
- R^2 can be made arbitrarily inflated by adding noise predictors.
- Standard p-values, C_p, AIC, BIC do not apply.

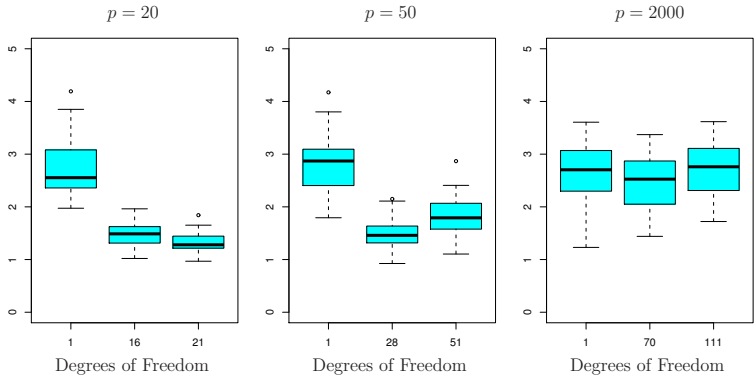
What works instead

Regularisation (ridge, lasso, elastic net), dimension reduction (PCR, PLS), and modern post-selection inference (Ch. 13).

In industry — Biotech: biomarker screening as triage

Biomarker studies have thousands of features on a few hundred subjects; the lasso's job there is *triage*.

Lasso in high dimensions



Lasso keeps working when $p \gg n$ if the truth is sparse *and* the sample can see it: with $n = 100$ and 20 true predictors, by $p = 2{,}000$ (right) no amount of shrinkage recovers the fit.

Source: Figure 6.24 — ISLP, James et al. (2023).

Outline

Python lab — run it live in the notebook

Companion notebook: `chapter_06_lab.ipynb`

The code in this section is meant to be **demonstrated live**. Open the companion Jupyter notebook and run it cell by cell, section by section:

- §2, *Best-subset and forward stepwise selection* — on the Hitters data loaded in §1;
- §3, *Ridge regression*, and §4, *Lasso* — each with its coefficient-path subsection;
- §5, *PCR and PLS*.

Data loads via the `ISLP` package or the bundled CSV files; §6, *Exercises*, closes the notebook.

Best subset and stepwise (manual)

```
import statsmodels.api as sm, itertools
from ISLP import load_data
Hitters = load_data("Hitters").dropna() # drop rows with NA salary
y = Hitters["Salary"] # response
preds = [c for c in Hitters.columns if c not in ("Salary",)]

best = {}
for k in range(1, 6): # model sizes k = 1..5
    best_rss = None
    for combo in itertools.combinations(preds, k): # all C(p,k) subsets
        X = sm.add_constant(Hitters[list(combo)])
        rss = sm.OLS(y, X).fit().ssr # training RSS of subset
        if best_rss is None or rss < best_rss:
            best_rss, best_combo = rss, combo # keep best model per size
    best[k] = (best_combo, best_rss)
```

Ridge regression

```
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
import numpy as np

X, y = Hitters[preds], Hitters["Salary"].values
# Scale INSIDE the search, so the scaler is re-fit on each training fold.
# Scaling first lets every fold help standardise itself: CV then over-shrinks.
ridge = GridSearchCV(make_pipeline(StandardScaler(), Ridge()),
                     {"ridge__alpha": np.logspace(-2, 4, 50)}, cv=10).fit(X, y)
print("best alpha:", ridge.best_params_["ridge__alpha"])
print(dict(zip(preds, ridge.best_estimator_[-1].coef_))) # all stay nonzero
```

The lasso

```
from sklearn.linear_model import Lasso

# same discipline: the scaler belongs inside the cross-validated search
lasso = GridSearchCV(make_pipeline(StandardScaler(), Lasso(max_iter=10000)),
                     {"lasso__alpha": np.logspace(-3, 1, 50)}, cv=10).fit(X, y)
print("best alpha:", lasso.best_params_["lasso__alpha"])
coefs = lasso.best_estimator_[-1].coef_
print([(p, c) for p, c in zip(preds, coefs) if c != 0]) # only the survivors
```

Principal components regression

```
from sklearn.decomposition import PCA
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.model_selection import cross_val_score

scores = []
for M in range(1, len(preds) + 1): # try every component count
    pcr = Pipeline([("pca", PCA(n_components=M)), # M principal comps
                    ("ols", LinearRegression())]) # then plain OLS
    s = -cross_val_score(pcr, X, y, cv=10, # 10-fold CV MSE
                         scoring="neg_mean_squared_error").mean()
    scores.append(s) # CV curve over M
```

Exercise 6.7 — CV Ridge and Lasso on Hitters [Python]

Exercise

On the `Hitters` data (predict `Salary`), (i) fit lasso with a cross-validated λ on standardised predictors, (ii) report the selected λ, and (iii) list which features the lasso keeps versus drops. Contrast with ridge. What do the results tell you?

Solution — Exercise 6.7 (1/2)

```
import numpy as np, pandas as pd
from sklearn.linear_model import Lasso, Ridge
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
# path as seen from Chapters/, where you run the notebook
df = pd.read_csv("../ALL CSV FILES - 2nd Edition/Hitters.csv").dropna()
y = df["Salary"].values
X = pd.get_dummies(df.drop(columns=["Salary"]), drop_first=True)
preds = X.columns # scale inside the CV, not here

lasso = GridSearchCV(make_pipeline(StandardScaler(), Lasso(max_iter=10000)),
                     {"lasso__alpha": np.logspace(-3, 2, 50)}, cv=10).fit(X, y)
print("lasso alpha:", round(lasso.best_params_["lasso__alpha"], 3))
coefs = lasso.best_estimator_[-1].coef_
kept = [p for p, c in zip(preds, coefs) if abs(c) > 1e-8]
dropped = [p for p, c in zip(preds, coefs) if abs(c) <= 1e-8]
print("kept :", kept, "\ndropped:", dropped) # retained vs zeroed exactly
```

Solution — Exercise 6.7 (2/2)

```
ridge = GridSearchCV(make_pipeline(StandardScaler(), Ridge()), # scale in CV
                     {"ridge__alpha": np.logspace(-2, 4, 50)}, cv=10).fit(X, y)
rc = ridge.best_estimator_[-1].coef_
print("ridge alpha:", round(ridge.best_params_["ridge__alpha"], 3),
      "-- nonzero coefs:", int(np.sum(np.abs(rc) > 1e-8))) # all p
```

Solution — expected result and interpretation

The lasso selects a moderate λ and *drives several coefficients to exactly zero* — typically it keeps strong career and recent-performance predictors (e.g. `CRBI`, `Hits`, `Walks`, `PutOuts`, `DivisionW`) and drops weak, redundant ones. Ridge, by contrast, keeps *all* predictors nonzero (only shrinks them). Takeaway: lasso yields a sparse, more interpretable model performing automatic variable selection, whereas ridge stabilises the fit while retaining every predictor. Choose based on whether sparsity or keeping all predictors is preferred.

Common mistake

Comparing the printed `alpha` with a textbook λ — `scikit-learn` scales the lasso penalty by $1/(2n)$, so the numbers differ across formulations even when the fitted models agree.

Extended Exercise 6.3 — CV Ridge and Lasso Test MSE on Hitters [Python]

Extended exercise (15 min)

On the `Hitters` data (predict `Salary`; drop missing salaries; dummy-encode `League`, `Division`, `NewLeague`):

1. Split into train/test, standardise predictors *using the training set only*, and cross-validate to choose λ for both ridge and lasso.
2. Report each chosen λ, the *test* MSE, and how many coefficients are non-zero.
3. List the predictors the lasso drops and contrast ridge vs. lasso.

Run this live: *Lecture exercises — worked Python solutions*

The runnable solution, with the printed numbers, is the second entry of that notebook section, headed *Extended Exercise 6.3* to match this deck.

Solution — Extended Exercise 6.3 (1/3)

```
import numpy as np, pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import RidgeCV, LassoCV
from sklearn.metrics import mean_squared_error

D = "../ALL CSV FILES - 2nd Edition/" # path from Chapters/
df = pd.read_csv(D+"Hitters.csv").dropna(subset=["Salary"]) # drop NA y
y = df["Salary"].values
X = pd.get_dummies(df.drop(columns=["Salary"]),
                   drop_first=True).astype(float) # encode factors
Xtr, Xte, ytr, yte = train_test_split(X.values, y,
                      test_size=0.33, random_state=1) # 2/3-1/3 split
sc = StandardScaler().fit(Xtr) # fit on TRAIN only: no leakage
Xtr, Xte = sc.transform(Xtr), sc.transform(Xte) # apply to both sets
```

Solution — Extended Exercise 6.3 (2/3)

```
ridge = RidgeCV(alphas=np.logspace(-2, 4, 50)).fit(Xtr, ytr) # CV grid
lasso = LassoCV(cv=10, max_iter=100000,
                random_state=0).fit(Xtr, ytr) # 10-fold CV for lambda

for name, m in [("ridge", ridge), ("lasso", lasso)]:
    mse = mean_squared_error(yte, m.predict(Xte)) # TEST-set MSE
    nz = int(np.sum(np.abs(m.coef_) > 1e-8)) # nonzero coefs
    print(name, "lambda=", round(m.alpha_, 3),
          "testMSE=", round(mse, 0), "nonzero=", nz)

dropped = [c for c, w in zip(X.columns, lasso.coef_)
           if abs(w) <= 1e-8] # zeroed by lasso
print("lasso drops:", dropped)
```

Solution — Extended Exercise 6.3 (3/3)

Solution — expected result and interpretation

With $n = 263$, $p = 19$ and this split you should see roughly:

- **Ridge**: $\lambda \approx 0.1$, test MSE $\approx 1.2 \times 10^5$, *all* 19 *coefficients non-zero*.
- **Lasso**: $\lambda \approx 0.8$, test MSE $\approx 1.2 \times 10^5$, about 16 non-zero — it drops `Years`, `CRBI`, `Errors` (weak or redundant given the correlated career totals).

Contrast. Both attain very similar test error, but the lasso reaches it with a *sparser, more interpretable* model (automatic variable selection), whereas ridge keeps and merely shrinks all predictors. Exact numbers shift with the split/seed; the qualitative message — lasso selects, ridge keeps all — is stable.

Common mistake

Reading the dropped variables as "irrelevant to salary". `Years` and `CRBI` are highly correlated with the kept career totals — the lasso drops *redundant* predictors, and a different split can swap which of a correlated group survives.

Outline

Chapter 6 in one slide

What we accomplished

Three families of methods that turn the linear model into a workhorse for high-dimensional and sparse problems.

- Subset selection (best, forward, backward).
- Shrinkage (ridge, lasso, elastic net).
- Dimension reduction (PCR, PLS).
- Honest tuning via cross-validation throughout.

Key formulas at a glance

C_p	$C_p = \frac{1}{n}\big(\mathrm{RSS} + 2d\hat{\sigma}^2\big)$ (lower is better).
AIC / BIC	$\mathrm{AIC} \propto \mathrm{RSS} + 2d\hat{\sigma}^2$; $\mathrm{BIC} = \frac{1}{n}\big(\mathrm{RSS} + \log(n)\, d\, \hat{\sigma}^2\big)$.
Adjusted R^2	$1 - \dfrac{\mathrm{RSS}/(n-d-1)}{\mathrm{TSS}/(n-1)}$.
Ridge (ℓ_2)	$\min_\beta \ \mathrm{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2$ (shrinks, keeps all predictors).
closed form	$\hat{\beta}^{\mathrm{R}} = (\mathbf{X}^\top \mathbf{X} + \lambda I)^{-1} \mathbf{X}^\top \mathbf{y}$ (β_0 unpenalised).
Lasso (ℓ_1)	$\min_\beta \ \mathrm{RSS} + \lambda \sum_{j=1}^{p} \lvert\beta_j\rvert$ (shrinks and selects: some $\hat{\beta}_j = 0$).
soft-threshold	$\hat{\beta}_j^{\mathrm{L}} = \mathrm{sign}(\hat{\beta}_j^{\mathrm{OLS}})\big(\lvert\hat{\beta}_j^{\mathrm{OLS}}\rvert - \lambda/2\big)_+$ (orthonormal $\mathbf{X}^\top\mathbf{X} = I$ only; no general closed form).
Tuning	choose d or λ by cross-validation.

Vocabulary checklist

Term	Meaning
Best subset	Search all $\binom{p}{k}$ models for each k
Forward / backward	Greedy add / drop one predictor at a time
C_p / AIC / BIC	Penalised RSS criteria for model size
Adjusted R^2	R^2 penalised for the number of predictors
Ridge (ℓ_2)	Penalty $\lambda \sum \beta_j^2$
Lasso (ℓ_1)	Penalty $\lambda \sum \lvert\beta_j\rvert$
Elastic net	Mix of ℓ_1 and ℓ_2 penalties
Tuning parameter λ	Strength of the penalty
PCR	Regression on principal components of X
PLS	Regression on directions correlated with Y
Sparsity	Many estimated β_j exactly zero

Decision rules of thumb

- Always standardise predictors before ridge / lasso / PCR / PLS.
- Choose λ by 10-fold CV; consider the one-standard-error rule for parsimony.
- Use BIC for parsimonious model selection; AIC / C_p for predictive accuracy.
- In $p > n$ regimes: ridge or lasso is mandatory.
- Re-fit OLS on the lasso-selected variables ("relaxed lasso") if you want unbiased coefficients with the sparse support.

Common pitfalls

Don't

1. Apply lasso / ridge without standardising.
2. Treat post-selection p-values as valid (Ch. 13).
3. Use AIC / BIC for high-dimensional data — they assume $n \gg p$.
4. Tune λ on the same data you evaluate on.
5. Throw away PCs based purely on variance — a low-variance PC can be highly predictive of Y.
6. Forget that PCR ignores Y; PLS is the supervised alternative.

Connections to the rest of the course

- Ch. 3: linear regression with no penalty.
- Ch. 5: how to choose λ honestly via CV.
- Ch. 7: GAMs are an additive ridge-like extension.
- Ch. 8: random forests / boosting handle high-dim differently.
- Ch. 10: deep nets use weight decay (ridge in disguise).
- Ch. 13: post-selection inference and the lasso.

Self-check questions

Each question names the slide that answers it — a wrong answer tells you where to read.

1. Why does ridge regression always have a unique solution? (p. 35)
2. Why can lasso set coefficients to exactly zero, but ridge cannot? Sketch the unit balls. (p. 45)
3. For two perfectly correlated predictors, what does ridge do? What does lasso do? What does elastic net do? (p. 47)
4. Compute the ridge estimate when $\mathrm{X}^\top \mathrm{X} = I$ — a clean special case. (appendix, p. 88)
5. PCR ignores Y when forming components. When can this hurt and when does it help? (p. 55)
6. Outline the one-standard-error rule. (p. 46)

Eight things to remember

1. Subset selection: clean conceptually but combinatorial.
2. Stepwise selection: greedy and fast.
3. Ridge: ℓ_2 penalty $\rightarrow$ smooth shrinkage, no zeros.
4. Lasso: ℓ_1 penalty $\rightarrow$ sparse solutions.
5. Always standardise before shrinkage.
6. Choose λ by cross-validation.
7. PCR / PLS: dimension reduction; PLS uses Y.
8. In high dimensions, regularisation is essential.

What's next

Chapter 7: *Moving Beyond Linearity*.

- Polynomial and step functions.
- Regression splines and smoothing splines.
- Local regression.
- Generalised additive models.

Appendix — what is in here, and why

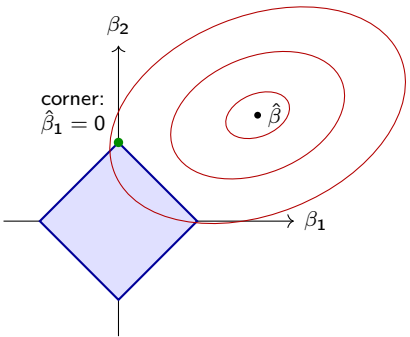
Nothing in the appendix is needed to follow the chapter: each slide is a formal derivation, a longer worked exercise, or a side topic. Read it when you want the full story, or when a later chapter sends you back here.

Contents of the appendix

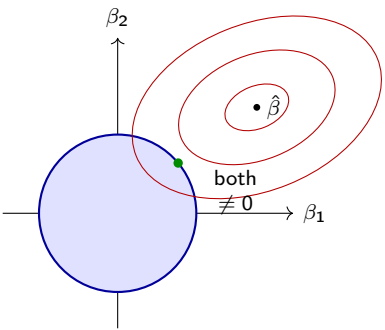
- **The constraint geometry, drawn a second time** — a duplicate of the diamond-vs-circle picture that is already in the main deck.
- **Counting the models (Exercise 6.1)** — how many subsets best subset and forward stepwise actually fit, 2^p against $1 + p(p+1)/2$. The lattice picture in the main deck makes the same point in one slide.
- **Ridge and lasso under an orthonormal design (Extended Exercise 6.2)** — the closed forms that explain *why* the lasso thresholds coefficients to zero and ridge only shrinks them. The cleanest proof in the chapter — and the hardest algebra.
- **Partial least squares** — PLS as the supervised cousin of PCR, with Exercise 6.6. PCR is the version that appears in the lab and the exam.

Why the lasso is sparse: the constraint geometry

Lasso (ℓ_1 diamond)



Ridge (ℓ_2 circle)



Takeaway

RSS contours first touch the ℓ_1 diamond at a *corner* ($\hat{\beta}_j = 0$); the smooth ℓ_2 circle almost never does.

Exercise 6.1 — Counting Models in Subset Selection [Math]

Exercise

A regression problem has $p = 10$ candidate predictors.

1. How many models does *best subset* selection consider in total (including the null model)? Give the general formula and the number.
2. How many models does *forward stepwise* selection fit in total? Give the general formula and the number.
3. By roughly what factor does best subset exceed forward stepwise here, and why does the gap explode as p grows?

Solution — Exercise 6.1 (1/2)

Solution

(1) Best subset. For each size k there are $\binom{p}{k}$ models, and summing over all sizes gives

$$\sum_{k=0}^{p} \binom{p}{k} = 2^p.$$

For $p = 10$: $2^{10} = 1024$ models.

(2) Forward stepwise. Start from the null model, then at step k (for $k = 0, \ldots, p-1$) fit the $p - k$ models that add one predictor to the current model:

$$1 + \sum_{k=0}^{p-1} (p - k) = 1 + \frac{p(p+1)}{2}.$$

For $p = 10$: $1 + \frac{10 \cdot 11}{2} = 1 + 55 = 56$ models. (Backward stepwise fits the same count, $O(p^2)$.)

Solution — Exercise 6.1 (2/2)

Solution

(3) The gap. Here $1024/56 \approx 18\times$. Best subset grows *exponentially* (2^p) while forward stepwise grows only *quadratically* ($O(p^2)$). For $p = 20$, best subset already considers $2^{20} \approx 10^6$ models versus $1 + \frac{20 \cdot 21}{2} = 211$ for forward stepwise — which is why best subset is infeasible beyond $p \approx 30$ and stepwise heuristics are used instead.

Interpretation. The saving is bought with greed: forward stepwise commits to one predictor per step and walks a single path through the model lattice, so it can miss the best subset of a given size — its winner must still be validated by CV or a penalised criterion.

Common mistake

Dropping the null model from either count — both include it ($2^{10} = 1024$ subsets *including* $\varnothing$; $1 + 55 = 56$ stepwise fits, the leading 1 being M_0).

Extended Exercise 6.2 — Orthonormal Design: Ridge vs. Lasso [Math]

Extended exercise (15 min)

Assume an *orthonormal* design, $\mathsf{X}^\top\mathsf{X} = I$, so the OLS estimates are $\hat\beta_j^{\mathrm{OLS}} = (\mathsf{X}^\top \mathsf{y})_j$ and the objective separates coordinate-by-coordinate.

1. **Ridge.** Minimise $\|\mathsf{y} - \mathsf{X}\beta\|^2 + \lambda\|\beta\|_2^2$ and show $\hat\beta_j^{\mathrm{R}} = \hat\beta_j^{\mathrm{OLS}}/(1+\lambda)$.
2. **Lasso.** Minimise $\|\mathsf{y} - \mathsf{X}\beta\|^2 + \lambda\|\beta\|_1$ and derive the soft-threshold solution $\hat\beta_j^{\mathrm{L}} = \mathrm{sign}(\hat\beta_j^{\mathrm{OLS}})\big(|\hat\beta_j^{\mathrm{OLS}}| - \lambda/2\big)_+$.
3. **Apply** both with $\lambda = 1$ to $\hat\beta^{\mathrm{OLS}} = (3.0, -1.5, 0.4, -0.2)$ and comment on shrinkage vs. selection.

Solution — Extended Exercise 6.2 (1/2)

Solution

Because $\mathsf{X}^\top\mathsf{X} = \mathit{I}$, $\|\mathsf{y} - \mathsf{X}\beta\|^2 = \text{const} + \sum_j (\beta_j - \hat\beta_j^{\mathrm{OLS}})^2$, so every penalty separates across j.

(1) Ridge. Minimise $(\beta_j - \hat\beta_j^{\mathrm{OLS}})^2 + \lambda\beta_j^2$. Setting the derivative to zero, $2(\beta_j - \hat\beta_j^{\mathrm{OLS}}) + 2\lambda\beta_j = 0 \Rightarrow (1+\lambda)\beta_j = \hat\beta_j^{\mathrm{OLS}}$, hence $\boxed{\hat\beta_j^{\mathrm{R}} = \hat\beta_j^{\mathrm{OLS}}/(1+\lambda)}$ — a constant *shrinkage factor*, never exactly zero.

(2) Lasso. Minimise $(\beta_j - \hat\beta_j^{\mathrm{OLS}})^2 + \lambda|\beta_j|$. For $\beta_j > 0$, $2(\beta_j - \hat\beta_j^{\mathrm{OLS}}) + \lambda = 0$ gives $\beta_j = \hat\beta_j^{\mathrm{OLS}} - \lambda/2$ (valid while positive); symmetrically for $\beta_j < 0$. Combining and clamping at 0, $\boxed{\hat\beta_j^{\mathrm{L}} = \operatorname{sign}(\hat\beta_j^{\mathrm{OLS}})(|\hat\beta_j^{\mathrm{OLS}}| - \lambda/2)_+}$.

Solution — Extended Exercise 6.2 (2/2)

Solution

(3) Apply with $\lambda = 1$ (ridge factor $1/(1+\lambda) = 0.5$, lasso threshold $\lambda/2 = 0.5$):

$\hat{\beta}^{\mathrm{OLS}}$	3.0	-1.5	0.4	-0.2
$\hat{\beta}^{\mathrm{R}} = \hat{\beta}^{\mathrm{OLS}}/2$	1.5	-0.75	0.20	-0.10
$\hat{\beta}^{\mathrm{L}} = \mathrm{soft}(\cdot, 0.5)$	2.5	-1.0	0	0

Comment. Ridge multiplies *every* coefficient by the same factor 0.5: all four survive, merely shrunk. The lasso subtracts 0.5 from each magnitude and clamps at zero, so the two small coefficients ($0.4, -0.2$, below the threshold 0.5) are set *exactly to zero* while the large ones are shrunk by the same absolute amount. This is precisely why the ℓ_1 penalty performs *variable selection* and the ℓ_2 penalty only shrinks.

Common mistake

Thinking the lasso shrinks proportionally. It subtracts the same *absolute* amount from every coefficient; ridge is the proportional one — compare the 3.0 and 0.4 rows.

Partial least squares (PLS)

PLS components are designed to maximise covariance with Y.

Construction

- First component: weights $\phi_{j1} \propto \mathrm{cov}(X_j, Y)$.
- Subsequent components: same idea after orthogonalising against earlier components.

Takeaway

PLS often performs comparably to ridge regression and can outperform PCR on noisy data. Cross-validate M for both methods on the same grid.

Exercise 6.6 — Dimension Reduction: PCR vs. PLS [Concept]

Exercise

1. In one sentence each, describe the key idea of dimension reduction and how PCR and PLS choose their directions $Z_1, \ldots, Z_M$.
2. State the essential difference between PCR and PLS (which one uses Y?).
3. Give a concrete situation in which PCR can perform poorly, and explain how PLS addresses it.

Solution — Exercise 6.6

Solution

(1) The idea. Dimension reduction replaces the p correlated predictors with $M < p$ linear combinations $Z_m = \sum_j \phi_{jm} X_j$ and fits an ordinary regression on those few components, cutting variance and handling collinearity. PCR takes the ϕ_{jm} from the *principal components* of X (the directions of maximal predictor variance); PLS chooses the ϕ_{jm} to maximise the *covariance between* Z_m *and* Y.
(2) The essential difference. PCR is *unsupervised* — it ignores Y when building components. PLS is *supervised* — it uses Y, so its directions are tilted toward what predicts the response.
(3) When PCR struggles. If the direction that best predicts Y happens to be a *low-variance* direction of X, PCR's leading components (chosen purely for high X-variance) will miss it, and you may need many components before the predictive direction enters — wasting degrees of freedom. PLS avoids this by explicitly seeking directions correlated with Y, so it can capture the predictive signal with fewer components. In practice PLS often performs comparably to ridge regression.

Common mistake

Concluding supervised must be better — using Y also raises variance, so PCR often matches PLS. And neither selects variables: every component Z_m mixes *all* p predictors.