AI Research Labs

Section 5 of 6 · 9 lessons

Regularisation and Model Selection

How penalties control model complexity, how ridge and LASSO differ, and how the regularisation strength λ shapes the fitted model.

5.1 Why Regularisation is Needed

5.2 Model Complexity and Feature Selection

5.3 Best Subset and Stepwise Selection

5.4 Ridge Regression (L2)

5.5 LASSO Regression (L1)

5.6 Understanding λ (Lambda)

5.7 Ridge versus LASSO

5.8 Mathematical Objectives and Worked Calculations

5.9 Selecting λ Using Cross-validation



Free course with quizzes, progress tracking and Scoopy, an AI tutor:
airesearchlabs.si/learn/ml

Machine Learning Foundations · Edition November 2026 · © 2026 AI Research Labs. Free for personal study.

Where this section fits

Every lesson is a step in building a model. These are the steps this section teaches; the Big picture PDF and the online map show how they connect to the rest.

5

SUPERVISED LEARNING · STEP 5

Control complexity

How simple can the model be and still capture the pattern?

Lessons here: 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9 · feeds into Turn predictions into decisions

Why Regularisation is Needed

Regularisation trades a little training fit for a simpler, more robust model. The two competing objectives, the benefits of parsimony, and the kinds of datasets where regularisation matters most.

Two goals in tension

Recall the U-shaped curve from 2.3: error on unseen data falls and then rises as complexity grows. **Regularisation** is a way to choose where to sit on that curve. It balances two conflicting objectives:

1. **Performance on the training data:** minimise in-sample error (RSS, misclassification).
2. **Robustness to new data:** minimise model complexity (number of variables, size of coefficients, tree depth, number of clusters). A simpler model is less likely to fit noise.

THE REGULARISED PROBLEM IN ONE LINE

$$\min_{\beta} \left\{ \underbrace{\text{training error}(\beta)}_{\text{fit the data}} + \lambda \cdot \underbrace{\text{Penalty}(\beta)}_{\text{keep it simple}} \right\}$$

The tuning parameter $\lambda \geq 0$ sets the exchange rate between the two.

There are two main families of penalty: **subset selection** (penalise how many coefficients are non-zero) and **shrinkage** (penalise how large they are: ridge and LASSO).

Why simpler models are worth it

BENEFIT	WHY
Performance	More likely to capture the true pattern than quirks of the training data
Robustness	Less likely to change dramatically when new data arrive
Computation	Easier to implement, maintain and re-estimate
Interpretability	Easier to explain and act on
Practicality	Fewer inputs to collect: lower storage, privacy and data costs (fewer sensors to buy)

When it matters most

- **Many predictors relative to observations** ($p \approx n$ or $p > n$). Genomics may have thousands of patients but a million gene features. Plain OLS fails outright when $p + 1 > n$.
- **Highly correlated predictors** (multicollinearity): OLS coefficients become unstable (3.9).
- **Noisy data**: more noise to accidentally fit.

In these cases OLS has high variance or no unique solution, while a regularised model still finds something robust.



Increasing λ raises training error steadily, but test error first falls (less variance) and then rises (too much bias).

Common doubts

FIRST PASS **How does regularisation change the objective?**

It adds a second term. Ordinary least squares minimises $\mathrm{RSS}(\beta)$; ridge minimises $\mathrm{RSS}(\beta) + \lambda \sum_j \beta_j^2$ and LASSO $\mathrm{RSS}(\beta) + \lambda \sum_j |\beta_j|$. The first term rewards fitting the data; the second charges for large coefficients. $\lambda = 0$ gives back OLS, and a large λ pushes the coefficients towards zero. The intercept is not penalised.

Cheatsheet

Regularised objective

$\min_\beta \{\text{training error}(\beta) + \lambda\,\text{Penalty}(\beta)\}$, $\lambda \geq 0$.

When to regularise

$p \approx n$ or $p > n$ · correlated predictors · noisy data.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/regularisation/why-regularisation-is-needed

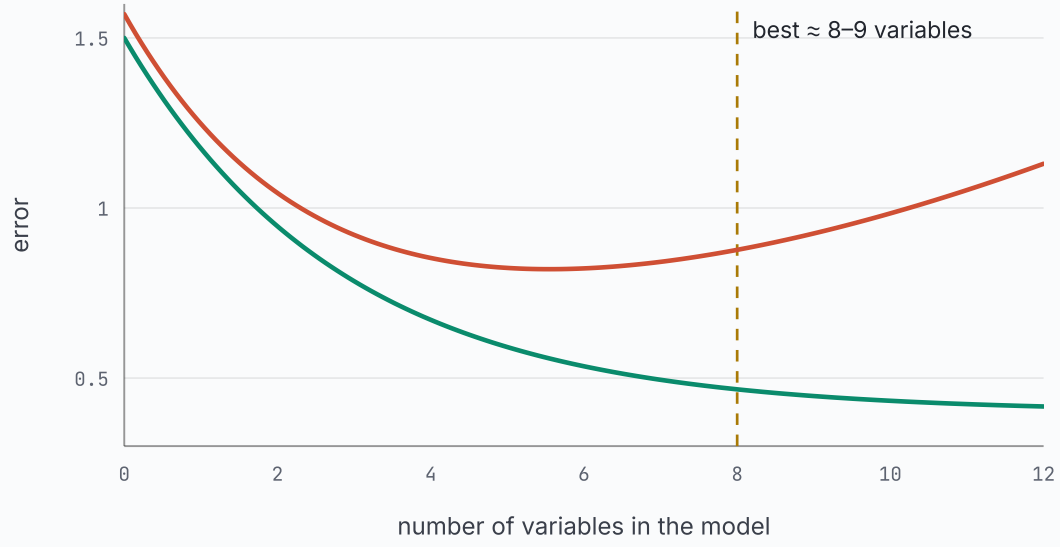
Model Complexity and Feature Selection

Complexity has many dials: number of features, coefficient size, depth, K. Why adding features always helps in-sample, why it eventually hurts out-of-sample, and how feature selection fits in.

Many dials, one idea

MODEL	COMPLEXITY DIAL	MORE COMPLEX WHEN...
Linear regression	Number of predictors; size of coefficients	More variables, bigger $\lvert\beta_j\rvert$
Polynomial regression	Degree	Higher degree
K-nearest neighbours	K	*Smaller* K (more local, wigglier)
Decision tree	Depth	Deeper
K-means	Number of clusters K	Larger K
SVM	Cost C; kernel bandwidth	Larger C

More features: in-sample vs out-of-sample



In-sample, adding a variable can only help. Out-of-sample, the best performance comes at a moderate number of variables; beyond that, the model overfits.

Adding a predictor to OLS can never increase training RSS: at worst its coefficient is set to zero. So training error always prefers the biggest model. Choosing **which** features to keep must use validation data, cross-validation, or a criterion that penalises size (adjusted R^2, AIC/BIC).

Ways to choose features

- **Domain knowledge:** fine when you genuinely know what's relevant.
- **Subset selection:** best subset, forward or backward stepwise (5.3).
- **Shrinkage that selects:** LASSO sets some coefficients exactly to zero (5.5).

THE P-VALUE TRAP

"We removed every variable with $p > 0.05$, refitted, and now all $p < 0.05$." That's not a sound selection procedure. It ignores the uncertainty of the selection itself, makes the final p-values over-confident (they're reported as if no selection happened), and targets in-sample significance rather than out-of-sample prediction. Use CV-based stepwise selection or LASSO.

AND THE LEAKAGE TRAP

Feature selection is a learning step. It must happen inside each CV training fold, never on the full dataset first (lesson 2.8).

Cheatsheet

Select by p-values

Dropping $p > 0.05$ and refitting ignores selection uncertainty and over-states significance. Use CV stepwise or LASSO.

Complexity dials

#features, $|\beta|$, degree, tree depth, small K (K-NN), large K (K-means), large C (SVM).

Online version with the interactive quiz: airesearchlabs.si/learn/ml/regularisation/model-complexity-and-feature-selection

Best Subset and Stepwise Selection

Best subset selection tries every combination of predictors; forward and backward stepwise approximate it greedily. The algorithms step by step, how many models each fits, and a worked forward-stepwise example.

The L0 penalty

Subset selection penalises the **number** of non-zero coefficients:

$$\min_{\beta} \ \mathrm{RSS}(\beta) + \lambda \|\beta\|_0, \qquad \|\beta\|_0 = \#\{j : \beta_j \neq 0\}$$

Best subset selection

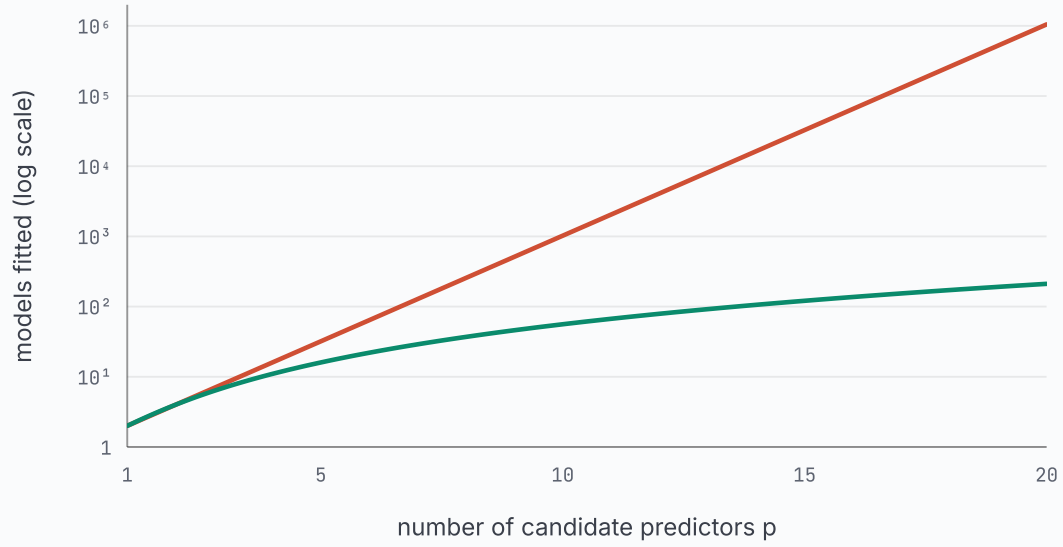
1. For each size $k = 0, 1, \ldots, p$: fit all $\binom{p}{k}$ models with exactly k predictors and keep the one with the lowest training RSS. Call it M_k.
2. Choose among $M_0, \ldots, M_p$ using a measure that accounts for complexity: validation or CV error, adjusted R^2, AIC/BIC. Not training RSS, which always favours M_p.

Benefit: directly implements parsimony and finds the best subset of each size. **Problem:** it fits 2^p models. $p = 10$: 1,024. $p = 20$: over a million. $p = 64$: hopeless.

Greedy approximations

FORWARD STEPWISE	BACKWARD STEPWISE
1. Start with M_0 (mean only). 2. For $k = 0, \ldots, p-1$: try adding each remaining predictor; keep the one that most improves the fit, giving M_{k+1}. 3. Choose the best of $M_0..M_p$ by validation/CV.	1. Start with the full model M_p. 2. For $k = p, \ldots, 1$: try removing each predictor; keep the removal that hurts least, giving M_{k-1}. 3. Choose the best size by validation/CV.

Models fitted: $1 + p + (p-1) + \cdots + 1 = 1 + \frac{p(p+1)}{2}$. For $p = 10$: 56 instead of 1,024; for $p = 20$: 211 instead of 1,048,576.



Best subset grows exponentially; stepwise grows only quadratically. In one benchmark, best subset took 163 s for 9 variables and was infeasible at 10, while stepwise handled 64 in milliseconds.

Trade-off: stepwise isn't guaranteed to find the best subset. Once forward stepwise adds a variable, it never removes it, even if a later combination would be better without it.

WORKED EXAMPLE: FORWARD STEPWISE BY HAND

Three candidate predictors A, B, C. TSS $= 100$.

- **Step 0:** M_0 (mean only): RSS $= 100$.
- **Step 1:** one-variable models: A 60, B 45, C 80. Keep B: $M_1 = \{B\}$.
- **Step 2:** add one more to B: A+B 30, B+C 40. Keep $M_2 = \{A, B\}$.
- **Step 3:** $M_3 = \{A, B, C\}$: RSS 28.

Training RSS keeps falling (100, 45, 30, 28), as it always does. Now compare by **validation RMSE**: 10.0, 7.0, 5.5, 5.8. Choose $M_2 = \{A, B\}$.

Models fitted: $1 + 3 + 2 + 1 = 7$, versus $2^3 = 8$ for best subset. The gap grows explosively with p.

Cheatsheet

Forward stepwise

Start M_0. Repeatedly add the predictor that most lowers RSS → $M_1..M_p$. Choose the size by validation/CV.

Best subset

For each k, best of $\binom{p}{k}$ models by RSS → M_k. Pick among M_k by CV / adj. R^2 / AIC / BIC. Cost 2^p.

Model counts

Best subset: 2^p. Stepwise: $1 + \frac{p(p+1)}{2}$.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/regularisation/best-subset-and-stepwise-selection

Ridge Regression (L2)

Ridge adds a penalty on the sum of squared coefficients. Its closed-form solution, why it shrinks, why it always has a solution, how it handles correlated predictors, and two worked examples.

RIDGE OBJECTIVE AND SOLUTION

$$\min_{\beta} \ \mathrm{RSS}(\beta) + \lambda \sum_{j=1}^{p} \beta_j^2 \qquad \Longrightarrow \qquad \hat{\beta}_\lambda^{\text{ridge}} = (X^\top X + \lambda I)^{-1} X^\top y$$

The intercept β_0 is not penalised (center the data, or exclude it from the penalty).

▶ Show the derivation

Why it shrinks

Adding λI makes the "denominator" bigger. In the simplest case $X^\top X = I$ (orthonormal predictors):

$$\hat{\beta}_j^{\text{ridge}} = \frac{\hat{\beta}_j^{\text{OLS}}}{1 + \lambda}$$

Every coefficient is divided by something larger than 1. In general all coefficients move towards zero as λ grows, and $\hat{\beta} \to 0$ as $\lambda \to \infty$. They never become exactly zero.

EXAMPLE A: ONE PREDICTOR, BY HAND

Centred data $x = (-1, 0, 1)$, $y = (-2, 0, 2)$. $\sum xy = 4$, $\sum x^2 = 2$.

$$\hat{\beta}_1^{\text{ridge}} = \frac{\sum xy}{\sum x^2 + \lambda} = \frac{4}{2 + \lambda}$$

λ	0	2	6	∞
slope	2	1	0.5	0
training RSS	0	2	4.5	8 (= TSS)

As λ grows, the slope shrinks and training RSS rises.

EXAMPLE B: TWO CORRELATED PREDICTORS

$$X^\top X = \begin{pmatrix} 4 & 3.6 \\ 3.6 & 4 \end{pmatrix}, \quad X^\top y = \begin{pmatrix} 5 \\ 4.6 \end{pmatrix}$$

OLS: $\det = 16 - 12.96 = 3.04$, so $\hat{\beta} = \frac{1}{3.04}\begin{pmatrix} 4(5) - 3.6(4.6) \\ -3.6(5) + 4(4.6) \end{pmatrix} = \frac{1}{3.04}\begin{pmatrix} 3.44 \\ 0.4 \end{pmatrix} = (1.132,\ 0.132)$.

Ridge, $\lambda = 1$: $X^\top X + I = \begin{pmatrix} 5 & 3.6 \\ 3.6 & 5 \end{pmatrix}$, $\det = 25 - 12.96 = 12.04$, so $\hat{\beta} = \frac{1}{12.04}\begin{pmatrix} 25 - 16.56 \\ -18 + 23 \end{pmatrix} = (0.701,\ 0.415)$.

Ridge shrinks the large coefficient and shares credit with its correlated twin. The sum barely moves (1.26 vs 1.12), while the split is much more even and stable.

Properties

- **Always solvable:** $X^\top X + \lambda I$ is invertible for any $\lambda > 0$, even when $X^\top X$ is singular ($p > n$, perfect collinearity). Adding λ to every eigenvalue makes them all positive.
- **Bias–variance:** as λ increases, bias increases and variance decreases.
- **Correlated predictors:** shares the credit instead of producing huge opposite-sign coefficients.
- **Cons:** many small but non-zero coefficients (no variable selection), and no convenient significance test for coefficients.

Cheatsheet

Ridge

$\min\ \mathrm{RSS} + \lambda \sum \beta_j^2$; $\hat{\beta} = (X^\top X + \lambda I)^{-1} X^\top y$.
Orthonormal: $\hat{\beta}^{\mathrm{OLS}}/(1+\lambda)$.

Ridge properties

Shrinks, never zero. Always solvable. Shares credit among correlated predictors. Bias ↑, variance ↓ with λ.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/regularisation/ridge-regression

LASSO Regression (L1)

LASSO penalises the sum of absolute coefficients, which shrinks some exactly to zero: built-in variable selection. Soft thresholding, a worked comparison with ridge, and when to prefer each.

LASSO OBJECTIVE

$$\min_{\beta} \ \mathrm{RSS}(\beta) + \lambda \sum_{j=1}^{p} |\beta_j|$$

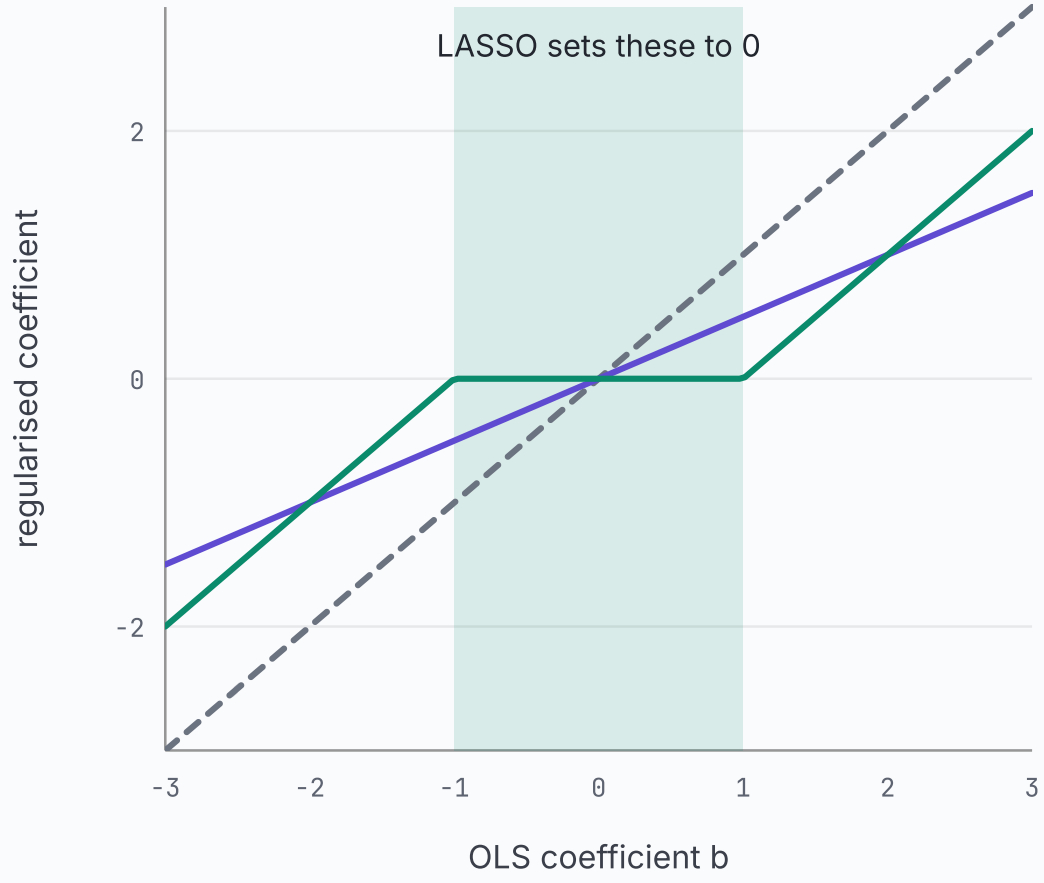
LASSO = Least Absolute Shrinkage and Selection Operator.

- Higher λ: smaller coefficients, higher bias, lower variance, just like ridge.
- **The key difference:** LASSO shrinks some coefficients to **exactly zero**, so it performs variable selection and gives sparse, interpretable models.
- No closed form: it's solved iteratively (e.g. coordinate descent). Harder than ridge, far easier than best subset.
- Like ridge, no convenient significance test for coefficients.

Soft thresholding

In the orthonormal case ($X^\top X = I$), with $b_j = \hat{\beta}_j^{\mathrm{OLS}}$ and the objective written as RSS $+\lambda\|\beta\|_1$:

$$\hat{\beta}_j^{\mathrm{ridge}} = \frac{b_j}{1+\lambda} \ \text{(proportional shrinkage)}, \qquad \hat{\beta}_j^{\mathrm{lasso}} = \mathrm{sign}(b_j)\ \max\Big(|b_j| - \tfrac{\lambda}{2},\ 0\Big) \ \text{(soft thresholding)}$$



Ridge scales every coefficient down. LASSO subtracts a constant and sets small coefficients exactly to zero.

WORKED EXAMPLE: RIDGE VS LASSO ON THE SAME OLS COEFFICIENTS

OLS: $b = (3, -1.5, 0.4)$, $\lambda = 2$, so the LASSO threshold is $\lambda/2 = 1$.

Ridge: $b/3 = (1.0, -0.5, 0.133)$. All non-zero.

LASSO: $(3-1,\ -(1.5-1),\ 0) = (2, -0.5, 0)$. The small coefficient becomes exactly 0 (variable dropped); the others shrink by a constant 1.

Ridge or LASSO?

SITUATION	FAVOUR
Many predictors, you believe only a few matter, you want interpretability	**LASSO** (sparse)
Many correlated predictors that all plausibly contribute a little	**Ridge** (shares credit, stable)
Groups of correlated predictors *and* you want selection	Elastic net (a mix of both)

LASSO WITH CORRELATED PREDICTORS

Among highly correlated predictors LASSO tends to pick one and drop the rest, somewhat arbitrarily; which one can change with a small change in the data. Don't read a dropped variable as "irrelevant", and never describe LASSO-selected variables as "the causes".

Cheatsheet

LASSO

$\min\ \mathrm{RSS} + \lambda \sum |\beta_j|$. Orthonormal: $\mathrm{sign}(b)\max(|b| - \lambda/2, 0)$ (soft threshold).

Ridge vs LASSO in one line

Ridge divides (never zero); LASSO subtracts (exact zeros → selection).

Online version with the interactive quiz: airesearchlabs.si/learn/ml/regularisation/lasso-regression

Understanding λ (Lambda)

What happens at λ = 0, small, intermediate, large and infinite λ. Why regularisation never beats OLS on training error, and how bias, variance and coefficients move along the way.

The regimes of λ

Λ	COEFFICIENTS	TRAINING ERROR	BIAS / VARIANCE	TYPICAL TEST ERROR
0	OLS	**Minimum possible**	Lowest bias, highest variance	High if overfitting
Small	Slightly shrunk	Rises a little	Variance falls	Often falls
Intermediate	Moderately shrunk (LASSO: some zero)	Higher	Balanced	**Best** (chosen by CV)
Large	Heavily shrunk	High	High bias	Rises again
$\to \infty$	All slopes → 0 (LASSO: exactly 0 beyond a finite λ)	= TSS (predicts $\bar{y}$)	Pure bias	Underfitting

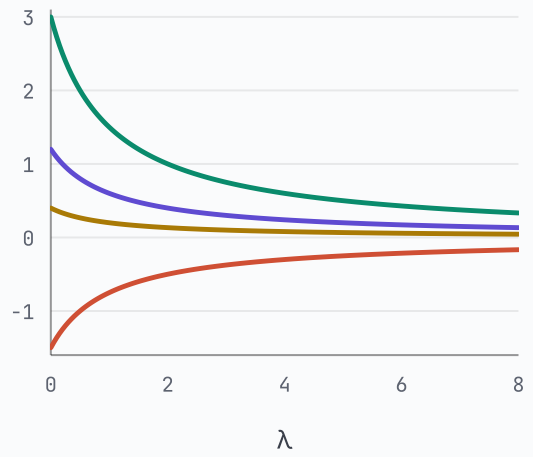
WHEN IS TRAINING ERROR LOWEST?

At $\lambda = 0$. With no penalty, ridge and LASSO both reduce to OLS, which by definition minimises RSS. Training error then increases (weakly) as λ grows. Regularisation buys better *test* error with extra bias; it never beats OLS on the training data.

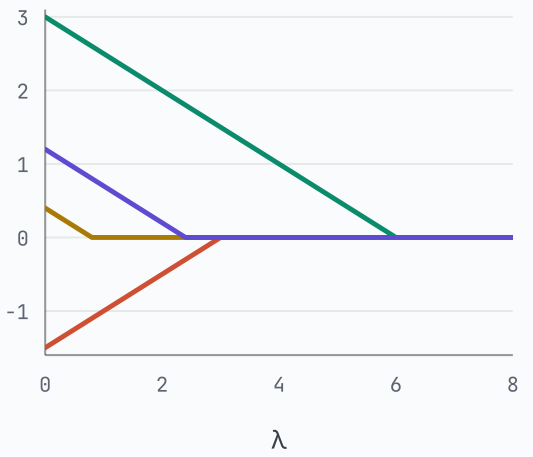
TRAINING LOSS VS PENALISED OBJECTIVE

For each λ, the regularised solution minimises *RSS + penalty*, so it beats OLS on that objective. But OLS beats it on plain RSS. If a question asks about "training error", it means RSS/MSE.

Coefficient paths



Ridge: every coefficient shrinks smoothly toward 0 but never reaches it.



LASSO: coefficients hit exactly 0 one after another (0.4 first, then 1.2, then −1.5) and stay there.

OLS coefficients 3 (teal), −1.5 (coral), 0.4 (amber) and 1.2 (violet), orthonormal design.

The order in which LASSO drops variables is a rough ranking of importance: the coefficients that survive longest matter most. In practice λ is plotted on a log scale.

INTERPRETING A CHANGE IN λ

Training MSE is 0.8 at $\lambda = 0$ and 1.1 at $\lambda = 5$; test MSE is 3.0 at $\lambda = 0$ and 1.9 at $\lambda = 5$. The unpenalised model overfits: the penalty raised training error slightly but cut test error substantially, by reducing variance.

Across many random training sets, regularised models give more **consistent** coefficients (less variable), at the price of bias. The net effect on prediction is best at an intermediate λ.

Cheatsheet

λ regimes

λ=0: OLS, min training error. Small: slight shrink, test often ↓. Middle: best (CV). Large: high bias. ∞: predicts $\bar{y}$.

Lowest training error?

Always at λ = 0. Regularisation never beats OLS on training RSS.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/regularisation/understanding-lambda

Ridge versus LASSO

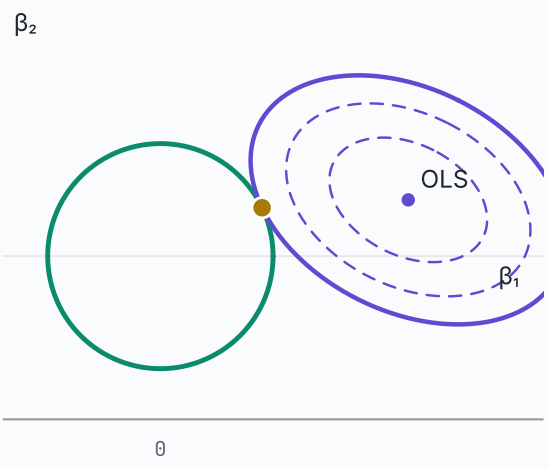
Why LASSO produces exact zeros and ridge does not, explained with the constraint geometry of a disc versus a diamond. A full side-by-side comparison with subset selection, plus an interactive shrinkage explorer.

The constraint view

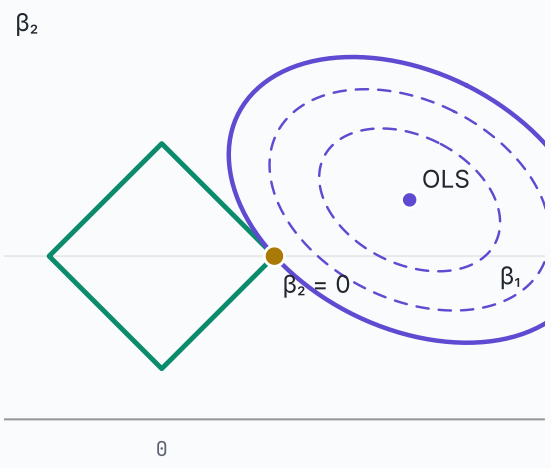
Each penalised problem is equivalent to minimising RSS **subject to a budget** on the coefficients:

$$\text{Ridge: } \min \text{RSS s.t. } \sum \beta_j^2 \leq s \qquad \text{LASSO: } \min \text{RSS s.t. } \sum |\beta_j| \leq s$$

RSS contours are ellipses centred on the OLS solution. The answer is where the expanding ellipse **first touches** the allowed region.



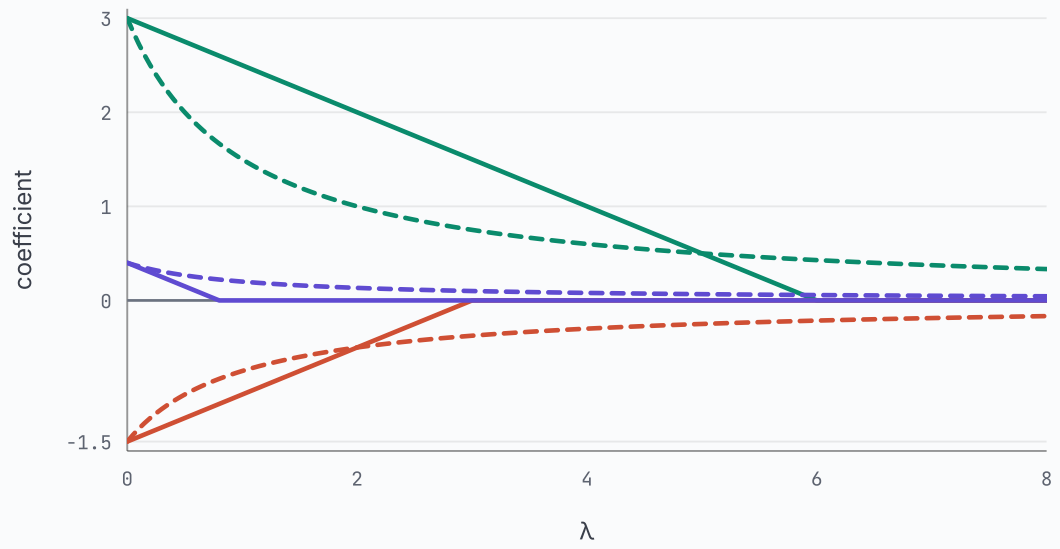
Ridge (disc): the ellipse touches a smooth boundary, so both coefficients are usually non-zero.



LASSO (diamond): the ellipse first touches a **corner** on an axis, where β_2 is exactly 0.

The diamond has corners sitting on the axes, and an ellipse coming from a typical direction very often hits a corner first. In high dimensions the L1 ball has many corners and edges, so many coefficients end up exactly zero. The ridge disc is round, with no corners to snag on.

RIDGE VS LASSO SHRINKAGE AS Λ GROWS



OLS coefficients b = (3, −1.5, 0.4) with an orthonormal design. Ridge divides every coefficient, so none reaches zero. LASSO subtracts, so 0.4 hits zero at λ = 0.8 and −1.5 at λ = 3.

Side by side

	SUBSET (L0)	RIDGE (L2)	LASSO (L1)
Penalty	Number of non-zeros	$\sum \beta_j^2$	$\sum \lvert\beta_j\rvert$
Exact zeros?	Yes	No	Yes
Closed form?	No (search over subsets)	Yes: $(X^\top X + \lambda I)^{-1} X^\top y$	No (iterative)
Computation	Very hard (2^p)	Easy	Moderate
$X^\top X$ singular / $p > n$	Subsets still small	Fine	Fine (selects at most n variables)
Correlated predictors	Picks one arbitrarily	Shares credit	Tends to pick one, drop others (unstable)
Interpretability	High	Medium (many small values)	High
Coefficient p-values	No convenient test	No convenient test	No convenient test

Elastic net mixes both penalties: a weight of 0 on the L1 part is ridge, 1 is LASSO, and values in between combine selection with ridge's stability for correlated groups.

"LASSO IS A MIDDLE GROUND BETWEEN BEST SUBSET AND RIDGE"

Like subset selection it produces sparse models (exact zeros); like ridge it is a continuous shrinkage problem that's cheap to solve. It gets most of subset selection's interpretability without the 2^p search.

Cheatsheet

Subset vs ridge vs LASSO

Zeros: yes / no / yes. Closed form: no / yes / no. Cost: 2^p / easy / moderate. Correlated: arbitrary / shares / picks one.

Why LASSO gives zeros

L1 constraint = diamond with corners on the axes; RSS ellipses usually touch a corner first.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/regularisation/ridge-versus-lasso

Mathematical Objectives and Worked Calculations

Norms, penalised objectives and how to compute ridge, LASSO and subset "errors" by hand, including choosing between two candidate fits, and penalised logistic regression.

Norms

For $\beta = (\beta_1, \ldots, \beta_p)$ (the intercept is excluded):

$$\|\beta\|_0 = \#\{j : \beta_j \neq 0\}, \qquad \|\beta\|_1 = \sum_j |\beta_j|, \qquad \|\beta\|_2^2 = \sum_j \beta_j^2$$

$\|\beta\|_1$ is the "Manhattan distance" from zero; $\|\beta\|_2^2$ the squared Euclidean distance.

EXAMPLE

$\beta = (3, -1, 0, 2)$: $\|\beta\|_0 = 3$, $\|\beta\|_1 = 3 + 1 + 0 + 2 = 6$, $\|\beta\|_2^2 = 9 + 1 + 0 + 4 = 14$.

The three objectives

$$\text{Ridge: } \text{RSS} + \lambda\|\beta\|_2^2 \qquad \text{LASSO: } \text{RSS} + \lambda\|\beta\|_1 \qquad \text{Subset: } \text{RSS} + \lambda\|\beta\|_0$$

COMPUTING THE "ERRORS" (OBJECTIVE VALUES)

A fit has RSS $= 12$ and $\hat{\beta} = (3, -2, 0.5, 0)$. Take $\lambda = 2$.

- Ridge: $12 + 2(9 + 4 + 0.25 + 0) = 12 + 26.5 = \mathbf{38.5}$
- LASSO: $12 + 2(3 + 2 + 0.5 + 0) = 12 + 11 = \mathbf{23}$
- Subset: $12 + 2 \times 3 = \mathbf{18}$ (three non-zero coefficients)

CHOOSING BETWEEN TWO CANDIDATE FITS

Fit A: RSS $= 10$, $\beta = (4, -3)$. Fit B: RSS $= 13$, $\beta = (2, -1)$.

OBJECTIVE	A	B	WINNER
Ridge, $\lambda = 1$	$10 + 25 = 35$	$13 + 5 = 18$	B
LASSO, $\lambda = 1$	$10 + 7 = 17$	$13 + 3 = 16$	B
Ridge, $\lambda = 0.1$	$10 + 2.5 = 12.5$	$13 + 0.5 = 13.5$	A

With a small penalty the better-fitting model wins; with a big penalty the smaller-coefficient model wins.

CONVENTIONS DIFFER

Some software divides RSS by $2n$ (scikit-learn's Lasso) or uses $\frac{1}{2}$RSS, which rescales what "λ" means. The story is identical (bigger penalty → smaller coefficients). If an exam question gives a formula, use that one.

Regularised logistic regression

The same idea applies to classification. Penalise the negative log-likelihood:

$$\min_{\beta} \{ -\ell(\beta) + \lambda \,\text{Penalty}(\beta)\}, \qquad \text{ridge: } \lambda \sum \beta_j^2, \quad \text{LASSO: } \lambda \sum |\beta_j|$$

- Ridge-logistic: smooth shrinkage, stabilises optimisation.
- LASSO-logistic: sparsity; effective when $p > n$ or features are correlated.
- It fixes perfect separation: coefficients can no longer diverge.
- λ is chosen by CV, e.g. minimising log-loss or maximising AUC.

PENALISED LOGISTIC OBJECTIVE BY HAND

$-\ell = 5.2$, $\beta = (1.5, -2)$, $\lambda = 0.5$. Ridge: $5.2 + 0.5(2.25 + 4) = 8.325$. LASSO: $5.2 + 0.5(1.5 + 2) = 6.95$.

Non-parametric models are regularised too: larger K in K-NN, shallower trees, and in SVMs a smaller cost C (6.7).

Cheatsheet

Norms

$\|\beta\|_0$ = #non-zeros; $\|\beta\|_1 = \sum |\beta_j|$; $\|\beta\|_2^2 = \sum \beta_j^2$ (intercept excluded).

Penalised objective by hand

1) Compute RSS (or $-\ell$). 2) Compute the norm of the slopes. 3) Add λ × norm. Compare candidates on the same λ.

Regularised logistic

$\min\{-\ell(\beta) + \lambda \sum \beta_j^2\}$ (ridge) or $+\lambda \sum |\beta_j|$ (LASSO).

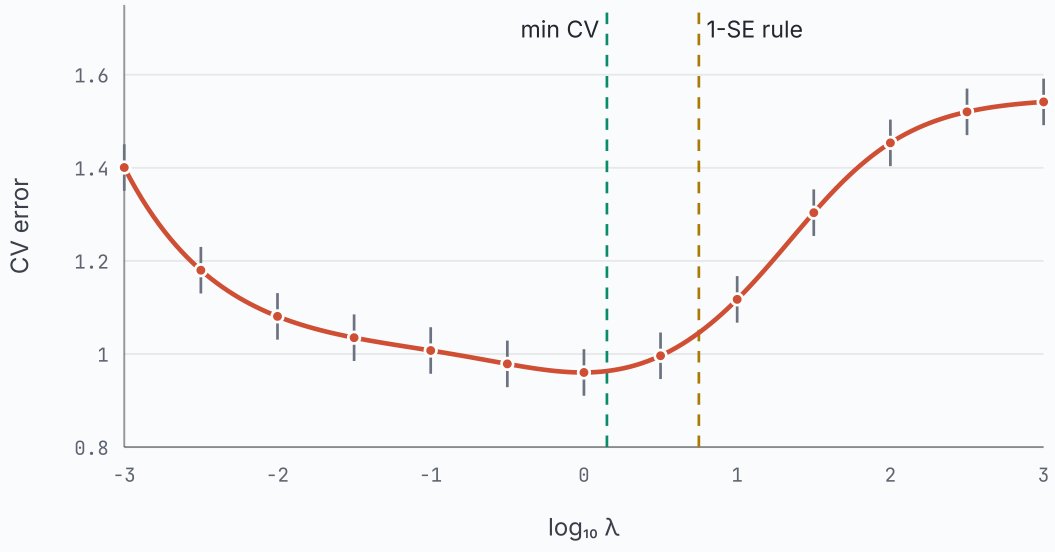
Online version with the interactive quiz: airesearchlabs.si/learn/ml/regularisation/mathematical-objectives-and-worked-calculations

Selecting λ Using Cross-validation

λ is a hyperparameter: choose it by cross-validation on standardised training data, then test once. The full procedure, the one-standard-error rule, a case study, and the mistakes to avoid.

The procedure

1. **Split** off a test set and leave it alone.
2. Choose a **grid of λ values**, log-spaced (e.g. 10^{-4} to 10^{2}).
3. For each λ, run **K-fold CV** on the training data. In each fold, **standardise using that fold's training part only**, fit, and record the validation error.
4. Pick the λ with the lowest mean CV error (or use the 1-SE rule below).
5. **Refit** on all training data with that λ.
6. **Evaluate once** on the test set.



Small λ (left) overfits; large λ (right) underfits. Bars show ± 1 standard error across folds. The 1-SE rule picks the largest λ within one SE of the minimum: a simpler model with nearly the same accuracy.

Why standardise first?

The penalty treats every coefficient the same, but a coefficient's size depends on its predictor's units. A variable measured in small units needs a big coefficient and gets penalised unfairly. Standardising (mean 0, SD 1) puts every predictor on the same footing. And to avoid leakage, the mean and SD must come from training folds only. The intercept is never penalised.

The one-standard-error rule

CV error is itself noisy. The **1-SE rule** chooses the largest λ whose CV error is within one standard error of the minimum. You get a simpler (more regularised, sparser for LASSO) model that is statistically indistinguishable in accuracy.

CASE STUDY: AN AIR-QUALITY SENSOR MODEL

Predicting CO from 11 standardised sensors ($n \approx 6{,}000$ training rows), λ by 5-fold CV:

	TEST R^2	TEST RMSE	NMHC SENSOR COEF.
OLS	0.885	0.478	−0.051
Ridge	0.885	0.477	−0.006
LASSO	0.885	0.477	0.000

Prediction: essentially identical. With 6,000 rows and only 11 predictors, OLS already has low variance; regularisation buys little accuracy.

Coefficients: the near-duplicate NMHC sensor (correlation 0.98 with benzene, VIF 61) is shrunk almost to zero by ridge and dropped by LASSO; the inflated benzene coefficient falls from 0.586 to 0.553. Coefficients become more stable and the model cheaper: one fewer sensor to buy.

Takeaway: regularisation's benefit isn't always accuracy. Stability, interpretability and cost can matter as much.

COMMON MISTAKES

- Selecting λ on the test set (then the test score is optimistic).
- Standardising on the whole dataset before CV (leakage).
- Forgetting to standardise at all.
- Calling LASSO-selected variables "the causes", or reporting p-values from an OLS refit on them as if no selection happened.

Cheatsheet

Choosing λ

Split off test → log-spaced λ grid → K-fold CV (standardise inside folds) → pick min CV or 1-SE → refit on all training → test once.

Regularisation & leakage

Standardise within training folds. Never choose λ on the test set.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/regularisation/selecting-lambda-using-cross-validation