

Regularised Regression

Aayush Bajaj

July 10, 2026

Contents

1	motivation	1
2	ridge	2
2.1	closed form	2
2.2	the svd view: shrinkage factors	2
2.3	effective degrees of freedom	3
3	lasso	3
3.1	orthonormal design: soft-thresholding	3
3.2	why corners give sparsity	4
3.3	coordinate descent	4
4	elastic net	5
5	the bayesian view: penalties are priors	5
6	in code	5
6.1	ridge from scratch, with the svd check	5
6.2	lasso via coordinate descent	6
7	see also	7

this page collects the closed-form solutions to regularised regression (where they exist) and the iterative approximations we fall back on (where they don't). along the way we will see that regularisation is not an ad-hoc hack but a perfectly sensible artefact of estimation: it drops straight out of MAP (maximum a posteriori) inference once you put a prior on the coefficients.

ridge has a closed form. lasso does not, except in one charmed special case.

1 motivation

ordinary least squares is the best *unbiased* linear estimator — but nobody said unbiased was the place to be.

- when columns of X are correlated, or p approaches n , the matrix $X^\top X$ is near-singular: tiny wobbles in y produce wild swings in $\hat{\beta}$. variance explodes while bias stays at zero — a terrible trade (see the [bias-variance decomposition](#)).
- regularisation deliberately buys a little bias to sell a lot of variance: shrink the coefficients toward zero, and the estimator stops chasing noise.
- a second, independent payoff: if the penalty can zero coefficients out exactly (the lasso can), you get variable selection and an interpretable model for free.

two pieces of housekeeping before any penalty touches anything (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009):

- **standardise** the inputs first. penalised solutions are not equivariant under feature scaling — a coefficient on “metres” and one on “millimetres” would be punished unequally.
- **never penalise the intercept.** shrinking the mean of y toward zero is not a belief anyone holds. centre y and the columns of X , fit without an intercept, add the means back at the end.

2 ridge

2.1 closed form

ridge regression penalises the squared ℓ_2 norm:

$$\hat{\beta}^{\text{ridge}} = \arg \min_{\beta} \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2, \quad \lambda \geq 0. \quad (1)$$

the objective is a quadratic in β ; setting the gradient $-2X^\top(y - X\beta) + 2\lambda\beta$ to zero gives

$$\boxed{\hat{\beta}^{\text{ridge}} = (X^\top X + \lambda I)^{-1} X^\top y} \quad (2)$$

- the inverse **always exists** for $\lambda > 0$: $X^\top X$ is positive semi-definite, so every eigenvalue of $X^\top X + \lambda I$ is at least λ . this was the original selling point — ridge as a numerical fix for singular normal equations, the statistics arriving later.¹
- at $\lambda = 0$ we recover OLS (**ordinary least squares**); as $\lambda \rightarrow \infty$ the coefficients slide monotonically to zero. no coefficient ever becomes *exactly* zero at finite λ — shrinkage, not selection.

2.2 the svd view: shrinkage factors

take the thin SVD $X = UDV^\top$, with singular values $d_1 \geq d_2 \geq \dots \geq d_p \geq 0$. substituting into the closed form,

the whole operator diagonalises in the svd basis — this one substitution is the entire proof.

$$\begin{aligned} X\hat{\beta}^{\text{ridge}} &= X(X^\top X + \lambda I)^{-1} X^\top y \\ &= UDV^\top (VD^2V^\top + \lambda VV^\top)^{-1} VDU^\top y \\ &= UD(D^2 + \lambda I)^{-1} DU^\top y \\ &= \sum_{j=1}^p u_j \frac{d_j^2}{d_j^2 + \lambda} u_j^\top y. \end{aligned}$$

compare with OLS, which is $\sum_j u_j u_j^\top y$: ridge computes the same coordinates of y in the basis $\{u_j\}$, then **shrinks the j -th coordinate by the factor**

¹hoerl and kennard (1970), *ridge regression: biased estimation for nonorthogonal problems*, *technometrics* 12(1) — written for chemists whose design matrices kept going singular.

$$s_j(\lambda) = \frac{d_j^2}{d_j^2 + \lambda} \in [0, 1]. \quad (3)$$

- directions with large d_j (high-variance principal directions of the column space) are barely touched; directions with small d_j — precisely the ones where OLS variance blows up — are crushed (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009).
- ridge therefore assumes the signal lives in the high-variance directions of the inputs. usually reasonable; occasionally spectacularly wrong.

2.3 effective degrees of freedom

ridge is a linear smoother, $\hat{y} = H_\lambda y$ with $H_\lambda = X(X^\top X + \lambda I)^{-1} X^\top$. its effective degrees of freedom is the trace:

$$\text{df}(\lambda) = \text{tr}(H_\lambda) = \sum_{j=1}^p \frac{d_j^2}{d_j^2 + \lambda}, \quad (4)$$

a smooth dial from $\text{df}(0) = p$ (full OLS) down to $\text{df}(\infty) = 0$ (constant model). it is the honest way to compare a ridge fit against a subset-selection fit “with k variables” — both are spending degrees of freedom, ridge just spends them in fractional coins (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009).

3 lasso

swap the squared norm for an absolute one and the character of the problem changes completely:

$$\hat{\beta}^{\text{lasso}} = \arg \min_{\beta} \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1. \quad (5)$$

the objective is convex but no longer differentiable at $\beta_j = 0$ — and that kink is exactly what manufactures sparsity.²

3.1 orthonormal design: soft-thresholding

when $X^\top X = I$ the problem separates across coordinates and each penalty has a closed form in terms of the OLS estimate $\hat{\beta}_j$ (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009):

estimator	formula	action
ridge	$\hat{\beta}_j / (1 + \lambda)$	shrink proportionally
lasso	$\text{sign}(\hat{\beta}_j) (\hat{\beta}_j - n\lambda)_+$	shrink by a constant, clip to zero ($n\lambda$ because eq. above averages the rss by $\frac{1}{2n}$)
best subset	keep $\hat{\beta}_j$ if large enough, else zero	keep-or-kill

the lasso formula is the **soft-thresholding operator** $S_\lambda(z) = \text{sign}(z)(|z| - \lambda)_+$: translate every coefficient λ toward zero, and everything that reaches zero stays there. ridge never zeroes; hard thresholding zeroes discontinuously; soft thresholding zeroes *continuously*.

²tibshirani (1996), *regression shrinkage and selection via the lasso*, jrss-b 58(1).

3.2 why corners give sparsity

write each penalised problem in its equivalent constrained form: minimise the residual sum of squares subject to $\|\beta\|_1 \leq t$ (lasso) or $\|\beta\|_2^2 \leq t^2$ (ridge). the solution is the first point of the constraint set touched by the expanding elliptical contours of the RSS.

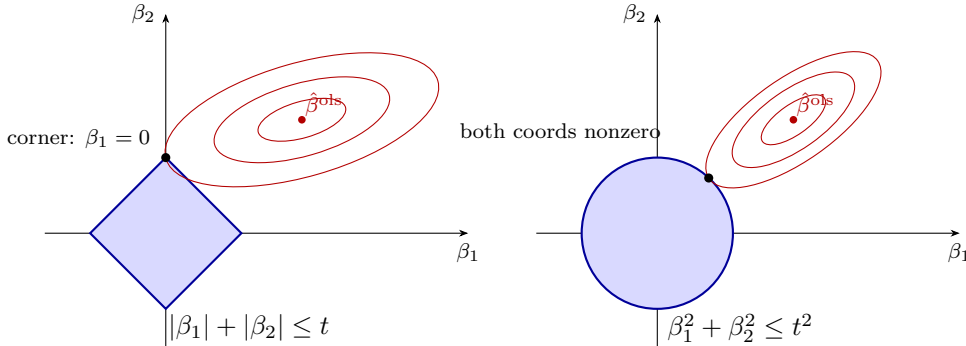


Figure 1: the classic picture: rss contours expand from the ols solution until they touch the constraint set. the ℓ_1 ball has corners on the coordinate axes, so first contact generically happens at a corner — a sparse solution. the ℓ_2 ball is smooth, so first contact is almost never on an axis.

- the ℓ_1 ball is a cross-polytope: its corners sit *on the coordinate axes*, where some coefficients are exactly zero. an expanding ellipse has positive probability of first touching a corner (or an edge, zeroing several coefficients at once).
- the ℓ_2 ball is rotationally symmetric — no privileged directions, no corners, no zeros.
- in p dimensions the effect intensifies: the ℓ_1 ball is nearly all corner.

the same geometry read backwards: the ℓ_1 ball is the convex hull of the coordinate axes' unit points — convexified sparsity.

3.3 coordinate descent

no closed form in general, but the problem is tailor-made for coordinate descent (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009). fix all coordinates except β_j and define the **partial residual** $r^{(j)} = y - \sum_{k \neq j} x_k \beta_k$. the one-dimensional problem in β_j is a quadratic plus an absolute value, minimised exactly by a soft-threshold:

$$\beta_j \leftarrow \frac{S_{n\lambda}(x_j^\top r^{(j)})}{\|x_j\|_2^2}. \quad (6)$$

- **input** standardised X , centred y , penalty λ , initial $\beta = 0$.
- **loop** cycle $j = 1, \dots, p$: form the partial residual cheaply (add $x_j \beta_j$ back into the running residual), soft-threshold, subtract the new contribution out.
- **stop** when no coordinate moves more than a tolerance.

convergence is guaranteed here because the non-smooth part of the objective is *separable* across coordinates — the one situation where coordinate descent on a non-differentiable convex function is safe. each sweep costs $O(np)$. in practice one solves a whole grid of λ values from large to small, warm-starting each solve at the previous solution — the “pathwise” strategy that makes a full lasso path nearly as cheap as a single fit.

4 elastic net

correlated features embarrass the lasso: among a group of near-duplicates it arbitrarily picks one and zeroes the rest, and the choice flips under resampling. the elastic net compromises:³

$$\hat{\beta} = \arg \min_{\beta} \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \left(\alpha \|\beta\|_1 + \frac{1-\alpha}{2} \|\beta\|_2^2 \right). \quad (7)$$

- $\alpha = 1$ is the lasso, $\alpha = 0$ is ridge; anything strictly between keeps the corners (sparsity survives) while the quadratic term makes the objective strictly convex — unique solutions even when $p > n$, and a **grouping effect**: strongly correlated features receive similar coefficients instead of a winner-takes-all.
- coordinate descent barely changes; the update becomes

$$\beta_j \leftarrow \frac{S_{n\lambda\alpha}(x_j^\top r^{(j)})}{\|x_j\|_2^2 + n\lambda(1-\alpha)}, \quad (8)$$

a soft-threshold followed by a ridge-style proportional shrink.

5 the bayesian view: penalties are priors

the cleanest justification for all of the above. model the data as gaussian, $y \mid \beta \sim \mathcal{N}(X\beta, \sigma^2 I)$, and put a prior on β . the MAP estimate maximises the log posterior:

$$\hat{\beta}^{\text{map}} = \arg \max_{\beta} \log p(y \mid \beta) + \log p(\beta). \quad (9)$$

- **gaussian prior** $\beta \sim \mathcal{N}(0, \tau^2 I)$: the log prior is $-\frac{1}{2\tau^2} \|\beta\|_2^2$ plus a constant, so

$$\hat{\beta}^{\text{map}} = \arg \min_{\beta} \|y - X\beta\|_2^2 + \frac{\sigma^2}{\tau^2} \|\beta\|_2^2, \quad (10)$$

which is **ridge with** $\lambda = \sigma^2/\tau^2$ (Deisenroth, Marc Peter and Faisal, A. Aldo and Ong, Cheng Soon, 2020). tight prior (small τ) means heavy shrinkage; the dial λ is literally your prior confidence that coefficients are small. for the gaussian–gaussian pair the posterior is itself gaussian, so the MAP estimate coincides with the posterior mean.

- **laplace prior** $p(\beta_j) \propto \exp(-|\beta_j|/b)$: the log prior is $-\frac{1}{b} \|\beta\|_1$ plus a constant, and the MAP estimate is the **lasso**. the laplace density’s spike at zero is the probabilistic shadow of the ℓ_1 ball’s corner.
- one honest caveat: the *full* laplace posterior puts zero mass on the event $\beta_j = 0$ — sparsity is a property of the mode, not of the posterior. a bayesian who reports a posterior mean gets no zeros at all.

a bayesian who wants genuine posterior sparsity reaches for spike-and-slab priors instead.

6 in code

6.1 ridge from scratch, with the svd check

closed form, the SVD route, and the effective degrees of freedom — all on a synthetic problem with 3 real signals and 5 correlated decoys:

³zou and hastie (2005), *regularization and variable selection via the elastic net*, jrssi-b 67(2).

```

1 import numpy as np
2
3 rng = np.random.default_rng(42)
4 n, p = 60, 8
5 # correlated design: 3 real signals, 5 decoys correlated with them
6 Z = rng.normal(size=(n, 3))
7 X = np.hstack([Z, Z @ rng.normal(size=(3, 5)) * 0.5 + rng.normal(size=(n, 5))])
8 beta_true = np.array([4.0, -2.0, 1.0, 0, 0, 0, 0, 0])
9 y = X @ beta_true + rng.normal(0, 1.0, n)
10
11 # standardise columns, centre y (never penalise the intercept)
12 X = (X - X.mean(0)) / X.std(0)
13 y = y - y.mean()
14
15 def ridge(X, y, lam):
16     p = X.shape[1]
17     return np.linalg.solve(X.T @ X + lam * np.eye(p), X.T @ y)
18
19 # svd view: identical fit via shrunken singular values
20 U, d, Vt = np.linalg.svd(X, full_matrices=False)
21 def ridge_svd(lam):
22     return Vt.T @ ((d / (d**2 + lam)) * (U.T @ y))
23
24 lams = [0.0, 1.0, 10.0, 100.0, 1000.0]
25 print(f"{'lam':>7} {'df':>6} coefficients (first 5)")
26 for lam in lams:
27     b = ridge(X, y, lam)
28     assert np.allclose(b, ridge_svd(lam)), "svd route disagrees"
29     df = np.sum(d**2 / (d**2 + lam))
30     print(f"{'lam':>7.0f} {'df':>6.2f} " + " ".join(f"{'v':>7.3f}" for v in b[:5]))

```

lam	df	coefficients (first 5)				
0	8.00	3.211	-2.021	0.848	0.218	0.248
1	7.74	3.118	-1.923	0.771	0.244	0.247
10	6.20	2.542	-1.404	0.396	0.350	0.248
100	2.61	1.095	-0.547	0.056	0.246	0.188
1000	0.44	0.183	-0.099	0.008	0.042	0.039

- the assert passes: $(X^T X + \lambda I)^{-1} X^T y$ and $V(D^2 + \lambda I)^{-1} D U^T y$ are the same estimator.
- df slides smoothly from 8 down to under half a degree of freedom.
- every coefficient shrinks, none dies: at $\lambda = 1000$ the decoys are small but stubbornly nonzero. that is the coefficient path of a shrinker, not a selector.

6.2 lasso via coordinate descent

same data, soft-thresholding in the loop:

```

1 def soft(z, g): # soft-threshold operator
2     return np.sign(z) * np.maximum(np.abs(z) - g, 0.0)
3
4 def lasso_cd(X, y, lam, iters=500, tol=1e-8):
5     n, p = X.shape
6     b = np.zeros(p)
7     col_ss = (X**2).sum(0) # ||x_j||^2
8     r = y.copy() # residual y - Xb
9     for _ in range(iters):
10         b_old = b.copy()
11         for j in range(p):

```

```

12     r += X[:, j] * b[j]      # remove j's contribution
13     b[j] = soft(X[:, j] @ r, n * lam) / col_ss[j]
14     r -= X[:, j] * b[j]      # add it back
15     if np.max(np.abs(b - b_old)) < tol:
16         break
17     return b
18
19 print(f"{'lam':>6} {'nonzero':>8} coefficients")
20 for lam in [0.001, 0.05, 0.2, 0.5, 1.0]:
21     b = lasso_cd(X, y, lam)
22     nz = int((np.abs(b) > 1e-10).sum())
23     print(f"{'lam':>6.3f} {'nz':>8} " + " ".join(f"{'v':>7.3f}" for v in b))

```

lam	nonzero	coefficients							
0.001	8	3.209	-2.018	0.846	0.217	0.247	0.319	-0.138	-
0.030									
0.050	7	3.131	-1.879	0.743	0.164	0.188	0.301	-0.000	-
0.032									
0.200	5	3.073	-1.603	0.554	0.000	0.000	0.292	0.000	-
0.027									
0.500	5	2.768	-1.226	0.132	0.000	0.000	0.221	0.000	-
0.021									
1.000	3	2.302	-0.814	0.000	0.000	0.000	0.003	0.000	-
0.000									

- exact zeros appear and multiply as λ grows — the corners at work.
- the decoys die first; the two strong signals are the last coefficients standing (the weak third signal is eventually mistaken for a decoy — sparsity is not free, it is bias by another name).
- note the survivors are also *shrunk*: soft thresholding both selects and biases. this is the lasso's standing critique, and the usual patch is to refit OLS on the selected support.

7 see also

- [ordinary least squares](#) — the $\lambda = 0$ endpoint of everything here
- [bias-variance decomposition](#) — the currency this page trades in
- [cross validation](#) — how λ actually gets chosen
- [kernel methods](#) — ridge in feature space is kernel ridge regression
- [logistic regression](#) — the same penalties bolt straight onto the logistic loss

references

Deisenroth, Marc Peter and Faisal, A. Aldo and Ong, Cheng Soon (2020). *Mathematics for Machine Learning*, Cambridge University Press.

Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome (2009). *The Elements of Statistical Learning*, Springer.