Overfitting, Bias-Variance, and Regularization

How much should a model trust its training data?

CSCI 567 — Machine Learning • Lecture 03 • Sep 11, 2026

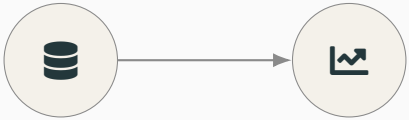
Outline

1. Bias: a model class that cannot represent the truth
2. Feature engineering: reducing bias by choosing the right map
3. Why not the highest-order polynomial? Overfitting and variance
4. The bias-variance decomposition: why the U-shape happens
5. Regularization I: Ridge (L2)
6. Regularization II: Lasso (L1) and the geometry of sparsity
7. A probabilistic view: MLE and MAP
8. Wrap-up: two loops closed

1. Bias

No amount of data fixes a model that cannot bend

The demo: $\mathbf{y} = \sin(2\pi x) + \varepsilon$



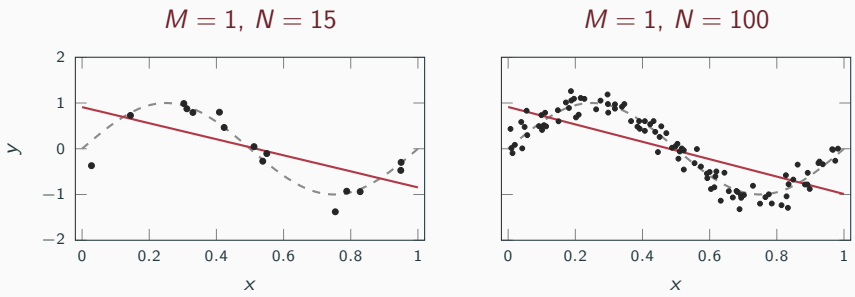
$N = 15$ points, x uniform on $[0, 1]$

fit degree $M \in \{0, 1, 3, 9\}$ by least squares

$$f^\star(x) = \sin(2\pi x), \quad \varepsilon \sim \mathcal{N}(0, \mathrm{Var}[\varepsilon]), \quad \mathrm{Var}[\varepsilon] = 0.04.$$

Measure MSE on the 15 training points and on a large held-out test set from the same distribution.

A straight line on sine data: more data does not help



$M = 1$	$N = 15$	$N = 100$	$N = 100{,}000$	noise floor
Test MSE	0.232	0.233	0.230	0.040

The line barely moves, and neither does its error: $6\times$ the noise floor at 15 points, and still $6\times$ at a hundred thousand.

Bias: the error a model class makes even with unlimited data

bias (informally, for now): the gap between the best function in the model class and the truth $f^\star$. Infinite data finds that best function exactly — and it is still wrong, because nothing in the class bends

what does not fix it: more data, a better optimizer, a smaller step size — these find the same best line faster or more precisely

what does: a richer model class — one that contains a function close to $f^\star$

Section 4 makes this precise as $\mathrm{Bias}(x_0)^2$; first, the tool that enlarges the class without giving up linear regression.

2. Feature engineering

Reducing bias by choosing the right map

Every model so far is linear in x: Section 1's failure is built into the class

$$f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$$

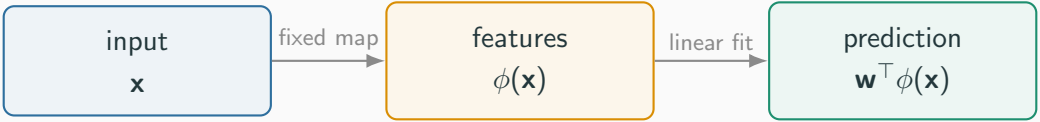
Linear in the parameters makes fitting easy — convexity, closed forms, gradient descent, all from Lecture 02.

But it is a real restriction on what the model can *express*: Section 1's straight line could not fit $\sin(2\pi x)$ however the weights were chosen, and more data did not help. To remove the bias we must change the **class**, not the sample — ideally without giving up any of Lecture 02's machinery.

Feature engineering: apply a fixed map ϕ first, then fit linearly

Choose a map $\phi : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ before seeing the fit, apply it to every input, and fit a linear model on top of $\phi(\mathbf{x})$ instead of $\mathbf{x}$:

$$f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}), \qquad \mathbf{w} \in \mathbb{R}^{\tilde{d}}$$



still linear *in* $\mathbf{w}$ — same loss, same convexity, same closed form and gradient descent as Lecture 02, with $\phi(\mathbf{x})$ playing the role $\mathbf{x}$ used to play; but nonlinear *in* $\mathbf{x}$, because ϕ is

Feature maps worth knowing, and what they have in common

- Polynomial features in $\mathbb{R}^d$: all monomials up to total degree M (e.g. $d = 2, M = 2$: $\phi(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$)
- Indicator/one-hot features for categorical inputs
- Interaction terms $x_i x_j$ for raw features suspected to interact
- Radial/distance features $\exp(-\|\mathbf{x} - \mathbf{c}\|^2)$ centered at chosen points $\mathbf{c}$
- Domain-specific transforms: log-scaling a skewed input, Fourier features for periodic structure

ϕ is chosen *before* fitting, using domain knowledge about what shape of nonlinearity might help — the fitting procedure afterward is unchanged plain linear regression

When each map earns its place, I: categories, scales, and interactions

Situation	Feature map	Why a linear model needs it
Rent from a ZIP code, or a flower's species	one-hot: $\phi(\mathbf{x})$ has one indicator per category	categories have no numeric order; treating ZIP 90089 as "90089" would make it 1 unit from 90090. One weight per category lets each get its own offset
Income, city population, word counts	$\log x$	inputs spanning several orders of magnitude; $\mathbf{w}^\top \log x$ says each *doubling* of x adds a fixed amount, which is how such effects usually behave
Drug dose $\times$ body weight; ad spend $\times$ holiday season	interaction $x_i x_j$	the effect of one input depends on another — no weights on x_i and x_j alone can express "dose matters more for lighter patients"; one weight on $x_i x_j$ can

in every row the fitting step is unchanged least squares; the domain knowledge went into choosing ϕ.

When each map earns its place, II: periodicity, locality, and curvature

Situation	Feature map	Why a linear model needs it
Hour of day, day of year, wind direction	Fourier: $\left(\sin \frac{2\pi t}{24}, \cos \frac{2\pi t}{24}\right)$	as raw numbers 23:00 and 01:00 are 22 apart; on the circle they are neighbors, so a linear model can treat them alike
House price vs. location; sensor reading vs. position	radial bumps $\exp(-\|\mathbf{x} - \mathbf{c}_k\|^2)$ at chosen centers $\mathbf{c}_k$	the effect is *local*: near downtown prices jump, far away they don't. Each bump measures nearness to $\mathbf{c}_k$; Lecture 04's RBF kernel puts a center at every training point
A response that bends: yield vs. temperature, our $\sin(2\pi x)$ demo	polynomial: monomials up to degree M	curvature and interactions at once; the price is size, $\tilde{d} = \binom{d+M}{M}$ ($d = 10$, $M = 3$: 286 features), the wrap-up's cost issue

The wrong map is a bias problem, not a variance one: no amount of data makes $\mathbf{w}^\top x$ fit a periodic signal in raw hours.

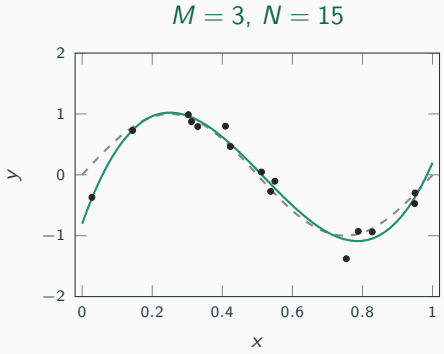
The family this lecture uses: powers of x, degree M as the complexity knob

Single input $x \in \mathbb{R}$, $\phi(x) = (1, x, x^2, \ldots, x^M) \in \mathbb{R}^{M+1}$, so $\tilde{d} + 1 = M + 1$ parameters:

$$f_{\mathbf{w}}(x) = \mathbf{w}^\top \phi(x) = \sum_{j=0}^{M} w_j x^j$$

- $M = 0$: constant only. $M = 1$: a line only. Larger M: bends more.
- One integer, M, moves the model class from too rigid to too flexible — exactly the knob needed to see underfitting and overfitting side by side.
- Same loss, same convexity, same closed form as Lecture 02 — only ϕ is new.

The payoff: $M = 3$ on the same 15 points removes the bias



M	Train MSE	Test MSE
1	0.298	0.232
3	0.016	0.065

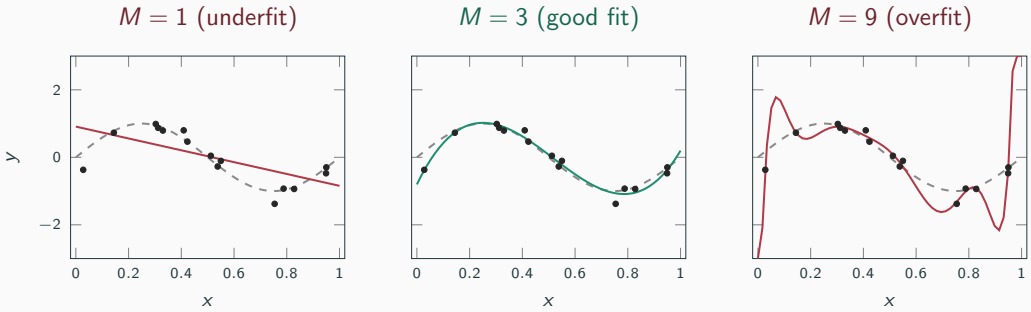
four features, $(1, x, x^2, x^3)$, and test error drops from 0.232 to 0.065 — within reach of the noise floor 0.04

So why not always pick the largest M we can afford?

3. Why not the highest-order polynomial?

Overfitting, and variance as the other axis

What each fit actually looks like



dashed gray: true $f^\star(x) = \sin(2\pi x)$ • solid: the fit • dots: the 15 training points. Real fits, this lecture's Colab notebook.

Reading the fits: too rigid, about right, memorizing the noise

- $M = 1$: too rigid to bend through the curve at all — consistently wrong.
- $M = 3$: tracks $\sin(2\pi x)$ closely, ignores most of the noise.
- $M = 9$: has enough free parameters (10) to snake through every single noisy point.

⚠ $M = 9$'s curve does not look like $\sin(2\pi x)$ between the training points — it memorized 15 specific values, noise included

The numbers: train vs. test MSE

M	Train MSE	Test MSE	$\|\mathbf{w}\|_2$
0	0.529	0.537	0.0
1	0.298	0.232	2.0
3	0.016	0.065	53.5
9	0.007	4.477	1.27×10^6

Noise floor: $\mathrm{Var}[\varepsilon] = 0.04$ — the best any model could do, since ε is irreducible randomness nothing can predict.

Reading the table: underfit, good fit, overfit

$M = 0, 1$: **underfitting.** Train *and* test error both high — the class is too rigid to represent $f^\star$. No amount of extra data fixes this.

$M = 3$: **a good fit.** Train and test error both low, close to each other, and close to the noise floor 0.04.

$M = 9$: **overfitting.** Lowest train error (0.007), but test error 4.48 — over 100× the noise floor. The weight norm 1.27×10^6 is no coincidence: 15 points, 10 parameters, $X^\top X$ severely ill-conditioned (Lecture 02's warning; fixed in Section 5).

Complexity and data amount are separate knobs

Same $M = 9$ model class, more data:

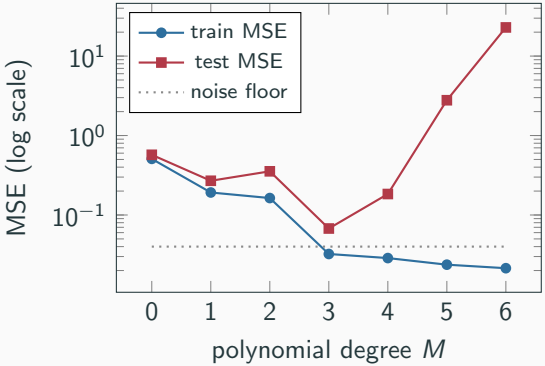
	$N = 15$	$N = 100$
Test MSE	4.48	0.043

overfitting is not intrinsic to a model class — it's a property of a model class *relative to how much data it sees*. No change to M, just more data, and the same class generalizes fine

Averaged over 200 resamples: test error is U-shaped in M

M	Train MSE	Test MSE
0	0.509	0.572
1	0.192	0.270
2	0.163	0.355
3	0.032	**0.068**
4	0.029	0.184
5	0.024	2.781
6	0.021	22.930

A single draw (previous slides) is illustrative but noisy; this averages 200 independent resamples at each degree.



Train error falls monotonically; test error is U-shaped

train error: falls *monotonically* in M — more parameters can only fit the training sample at least as well

test error: **U-shaped** — falls while the model is too rigid, bottoms near $M = 3$, explodes past it

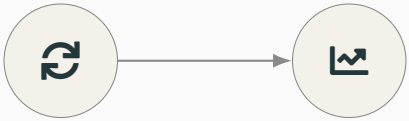
variance (informally, for now): error from chasing the particular sample that was drawn — the mirror image of Section 1's bias. Bias did not shrink with more data; variance did ($M = 9$: $4.48 \to 0.043$ from $N = 15$ to $N = 100$)

Next: make both terms precise, and see why the U-shape happens.

4. The bias-variance decomposition

Why the U-shape happens: bias falls, variance rises

Setup: resample S, and the fitted model f_S changes; $f^\star$ never does



$f^\star$: the fixed, unknown true function — it never changes.

f_S: the model *fit from one specific sample* S — Lecture 02's $\mathbf{w}^\star$ with the dependence on which sample we drew made explicit; different every time we resample.

The quantity to explain: expected squared error at a fixed test point

Fix a test point x_0 with $\mathbf{y}_0 = f^\star(x_0) + \varepsilon_0$, where ε_0 is fresh label noise, independent of S.

$$\mathrm{Error}(x_0) := \mathbb{E}_{S,\varepsilon_0}\left[(\mathbf{y}_0 - f_S(x_0))^2\right]$$

Averaged over *both* which training sample we happened to draw, and the test point's own noise.

The bias-variance decomposition

Let $\bar{f}(x_0) := \mathbb{E}_S[f_S(x_0)]$ — the average prediction, over infinitely many resamples of S.

$$\text{Error}(x_0) = \underbrace{\text{Var}[\varepsilon]}_{\text{noise}} + \underbrace{\left(f^\star(x_0) - \bar{f}(x_0)\right)^2}_{\text{Bias}(x_0)^2} + \underbrace{\mathbb{E}_S\left[(f_S(x_0) - \bar{f}(x_0))^2\right]}_{\text{Var}_S[f_S(x_0)]}$$

Three non-negative terms, exact sum — no leftover. **Where does this come from?**

Derivation, steps 1–2: split off the noise, then add and subtract $\bar{f}$

Step 1: expand the square. $(\mathbf{y}_0 - f_S(x_0))^2 = \big(f^\star(x_0) + \varepsilon_0 - f_S(x_0)\big)^2$. ε_0 is independent of S (hence of f_S) and mean zero, so its cross term with $f^\star(x_0) - f_S(x_0)$ vanishes in expectation, and $\mathbb{E}[\varepsilon_0^2] = \mathrm{Var}[\varepsilon]$:

$$\mathrm{Error}(x_0) = \mathrm{Var}[\varepsilon] + \mathbb{E}_S\big[(f^\star(x_0) - f_S(x_0))^2\big]$$

Step 2: add and subtract the average prediction $\bar{f}(x_0) = \mathbb{E}_S[f_S(x_0)]$ inside the remaining square:

$$f^\star(x_0) - f_S(x_0) = \big[f^\star(x_0) - \bar{f}(x_0)\big] + \big[\bar{f}(x_0) - f_S(x_0)\big]$$

Derivation, step 3: square it, and the cross term vanishes

Squaring and taking $\mathbb{E}_S$: the cross term vanishes because $\mathbb{E}_S[\bar{f}(x_0) - f_S(x_0)] = 0$, by definition of $\bar{f}$.

$$\mathbb{E}_S\left[(f^\star(x_0) - f_S(x_0))^2\right] = \mathrm{Bias}(x_0)^2 + \mathrm{Var}_S[f_S(x_0)]$$

Combined with Step 1: exactly the boxed decomposition. **No leftover term.**

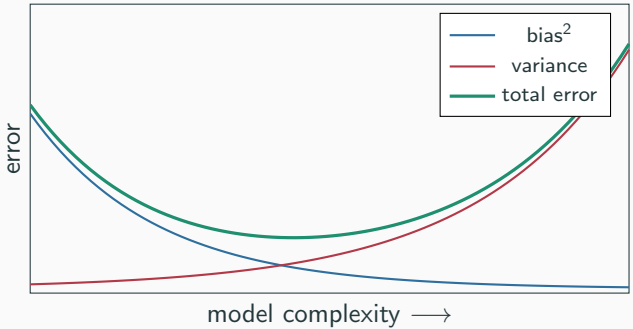
Reading the three terms: noise, bias, variance

🚫 **noise** $\mathrm{Var}[\varepsilon]$: **irreducible**. Even a model that knew $f^\star$ exactly would still miss by this much, on average. In the demo, the well-fit $M = 3$ test MSE (0.068) couldn't go much below 0.04.

🎯 **bias** $\mathrm{Bias}(x_0) = f^\star(x_0) - \bar{f}(x_0)$: how far the *average* prediction (over resampled S) is from the truth. High bias $\Rightarrow$ the model class itself is too rigid — **underfitting**.

🔀 **variance** $\mathrm{Var}_S[f_S(x_0)]$: how much the prediction *swings* depending on which training sample was drawn. High variance $\Rightarrow$ the model chases the specific noise it saw — **overfitting**.

The tradeoff



schematic — shape only, not literal data.

The U-shape is $\mathrm{Bias}^2 + \mathbf{Variance}$, traced over model complexity: bias falls, variance rises, total is minimized where they cross.

Verified against the demo: the decomposition is an identity

$x_0 = 0.3$, $N = 15$, 3000 resamples:

M	Bias^2	Var	Noise	Total (direct sim.)
1 (underfit)	0.306	0.027	0.040	0.374 (0.367)
3 (good fit)	0.002	0.012	0.040	0.054 (0.053)
9 (overfit)	0.034	223.74	0.040	223.82 (223.90)

Decomposed sum matches direct simulation — an identity, not an approximation.

Reading the verification: $M = 1$ is bias-dominated, $M = 9$ is variance-dominated

$M = 1$: bias-dominated (0.306 vs. variance's 0.027) — consistently wrong about the curve, regardless of which 15 points it saw

$M = 9$: variance catastrophic (223.74 vs. bias^2's 0.034) — unbiased *on average*, but any *one* fit swings wildly

The question Section 4 leaves: control variance without shrinking M

Section 4 explains *why* $M = 9$ overfits: catastrophic variance.

> 💡 shrinking M outright would fix variance but reintroduce bias (the $M = 1$ row) — is there a way to control variance **without** shrinking the model class?

Everything in this section assumed the model class is fixed and its complexity stays **below** the amount of data; the wrap-up returns to what feature engineering does to that assumption.

5. Regularization I: Ridge (L2)

Penalize complexity directly

Ridge's idea: penalize large weights instead of shrinking the model class

keep M large, but discourage the fitting procedure from choosing large, wild weights unless the data really demands them

This avoids shrinking the model class outright — which Section 4 already showed reintroduces bias.

Ridge's objective: empirical risk plus an ℓ_2 penalty

$$\hat{\mathcal{R}}_\lambda(\mathbf{w}) := \hat{\mathcal{R}}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 = \frac{1}{2n}\sum_{i=1}^{n}(\mathbf{w}^\top \mathbf{x}_i - \mathbf{y}_i)^2 + \frac{\lambda}{2}\sum_j w_j^2$$

$\lambda > 0$: **regularization strength**. $\lambda = 0$ recovers plain ERM; larger λ trades training fit for smaller $\|\mathbf{w}\|$

Ridge regression is due to Hoerl and Kennard (1970).

Ridge's objective is still convex

$\hat{\mathcal{R}}_\lambda(\mathbf{w})$ is a sum of two convex quadratics:

- $\hat{\mathcal{R}}(\mathbf{w})$ — Lecture 02's Hessian-PSD argument.
- $\frac{\lambda}{2}\|\mathbf{w}\|_2^2$ — Hessian $\lambda I \succeq 0$.

convex + convex = convex $\Rightarrow$ every stationary point is the global minimum $\Rightarrow$ solve $\nabla\hat{\mathcal{R}}_\lambda(\mathbf{w}) = 0$ directly

Ridge's closed form: the normal equations plus $n\lambda I$

$$\nabla \hat{\mathcal{R}}_\lambda(\mathbf{w}) = \frac{1}{n} X^\top (X\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w} = 0 \qquad \Longrightarrow \qquad \big(X^\top X + n\lambda I\big)\mathbf{w} = X^\top \mathbf{y}$$

$$\mathbf{w}^\star_{\text{ridge}} = \big(X^\top X + n\lambda I\big)^{-1} X^\top \mathbf{y}$$

Same shape as Lecture 02's normal equations — one extra term, $n\lambda I$.

The problem this solves: $X^\top X$ can have zero eigenvalues

Lecture 02 flagged: $X^\top X$ can be **singular** ($d + 1 > n$) or **near-singular** (collinear features) — plain least squares becomes unstable or undefined

Does adding $n\lambda I$ actually fix this? **Let's prove it, not just claim it.**

Setup. $X^\top X$ is symmetric PSD $\Rightarrow$ it has an eigendecomposition

$$X^\top X = \sum_k \mu_k \mathbf{v}_k \mathbf{v}_k^\top, \qquad \mu_k \geq 0,$$

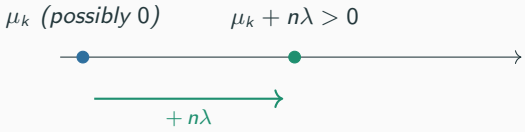
and some directions can have $\mu_k = 0$ exactly — that is precisely what "singular" means.

Adding $n\lambda I$ shifts every eigenvalue by $n\lambda$

Since $I\mathbf{v}_k = \mathbf{v}_k$, adding $n\lambda I$ leaves eigenvectors unchanged and shifts every eigenvalue:

$$\left(X^\top X + n\lambda I\right)\mathbf{v}_k = \left(\mu_k + n\lambda\right)\mathbf{v}_k$$

Every eigenvalue becomes strictly positive, so the matrix is always invertible



For any $\lambda > 0$: every shifted eigenvalue is strictly positive.

$X^\top X + n\lambda I$ is therefore **always invertible**, regardless of X

Numerical check

$n = 5$, $d = 8$ ($d + 1 > n$, singular by construction):

eigenvalues of $X^\top X$:
$\{0, 0, 0, 1.30, 2.86, 5.28, 8.20, 13.66\}$

with $\lambda = 0.1$ ($n\lambda = 0.5$):
$\{0.5, 0.5, 0.5, 1.80, 3.36, 5.78, 8.70, 14.16\}$ — **all positive**

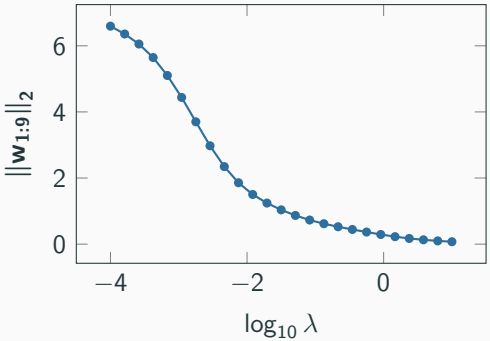
The same penalty caps the separable-data weight blowup in logistic regression

the same $\frac{\lambda}{2}\|\mathbf{w}\|_2^2$ term, added to logistic regression's empirical risk, caps separable data's weight blowup (seen earlier today, in Lecture 02's classification half) — nothing in the plain objective penalized $\|\mathbf{w}\| \to \infty$; this does

One fix, two warning signs closed.

Effect on the demo: Ridge shrinks, but never exactly to zero

$M = 9$, $N = 30$, standardized features, λ from 10^{-4} to 10:



	$\lambda = 10^{-4}$	$\lambda = 10$
$\|\mathbf{w}_{1:9}\|_2$	6.59	0.07

coefficients shrink smoothly — **no individual coefficient** reaches exactly zero, at any λ tested

For *exact* sparsity, we need a different penalty.

6. Regularization II: Lasso (L1)

Exact zeros from a different norm

Lasso's objective: convex, but not differentiable at zero, so no closed form

$$\hat{\mathcal{R}}_\lambda(\mathbf{w}) := \hat{\mathcal{R}}(\mathbf{w}) + \lambda \|\mathbf{w}\|_1 = \frac{1}{2n} \sum_{i=1}^{n} (\mathbf{w}^\top \mathbf{x}_i - \mathbf{y}_i)^2 + \lambda \sum_j |w_j|$$

Same idea, different norm. Tibshirani (1996).

convex (sum of convex $|w_j|$'s) $\Rightarrow$ every stationary point is still a global minimum

not differentiable at $w_j = 0 \Rightarrow$ no closed form, plain GD does not apply — solved iteratively (proximal gradient, three slides ahead)

Constrained view: the smallest ellipse that touches the constraint region

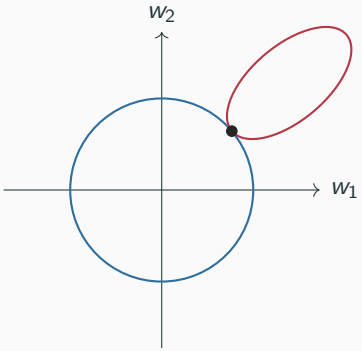
standard convex-duality fact (not derived here): minimizing $\hat{\mathcal{R}}(\mathbf{w}) + \lambda \|\mathbf{w}\|_p$ gives the **same** minimizer as minimizing $\hat{\mathcal{R}}(\mathbf{w})$ subject to $\|\mathbf{w}\|_p \leq t$, for a corresponding $t > 0$

$\hat{\mathcal{R}}(\mathbf{w})$ is a quadratic bowl in $\mathbf{w} \Rightarrow$ its level sets (constant training-error contours) are ellipses centered at the unconstrained least-squares solution.

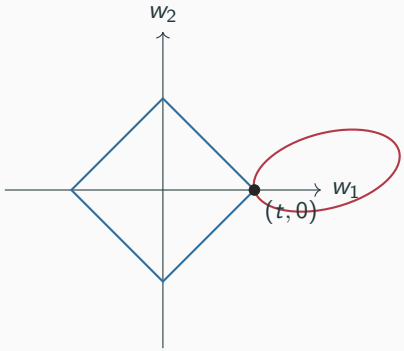
the constrained minimizer is the point where the *smallest* such ellipse first touches the constraint region $\{\|\mathbf{w}\|_p \leq t\}$

This constrained view makes the sparsity difference *visible*.

Ridge's ball is a circle, Lasso's is a diamond: the tangent point lands on a corner



Ridge, $\|\mathbf{w}\|_2 \leq t$: the boundary is smooth **everywhere** — an ellipse can be tangent at any point, with roughly equal ease.



Lasso, $\|\mathbf{w}\|_1 \leq t$: the tangent point lands **at a corner** — and a corner of the diamond has $w_2 = 0$ exactly.

Corners are where the ellipse lands first, and corners have exact zeros

- ℓ_2 (smooth boundary): tangency at any point needs *one* exact ellipse orientation — no point is special.
- ℓ_1 (corners on the axes): a whole *range* of ellipse orientations is tangent right at a corner.

corners are disproportionately likely to be
where the smallest ellipse first touches

a corner of the ℓ_1 ball has one or more coordinates **exactly zero**
— this, not an approximation, is why Lasso zeros out weights

Ridge's ball has no corners — it essentially never lands on an axis exactly.

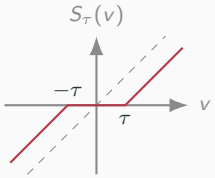
Solving Lasso, step 1: the 1-D subproblem in closed form (added Sep 17)

Plain GD fails at $w_j = 0$, where $|w_j|$ has no derivative. The fix: a gradient step on $\hat{\mathcal{R}}$, then handle the penalty *exactly*, one coordinate at a time:

$$\min_w \ \tfrac{1}{2}(w - v)^2 + \tau|w| \qquad (v = \text{that coordinate after the gradient step},\ \tau > 0)$$

Three cases (Lecture 02's subgradients at the kink):

- $w > 0$: $w - v + \tau = 0 \Rightarrow w = v - \tau$, valid iff $v > \tau$
- $w < 0$: $w - v - \tau = 0 \Rightarrow w = v + \tau$, valid iff $v < -\tau$
- $w = 0$: subgradients $-v + \tau s$, $s \in [-1, 1]$, contain 0 iff $|v| \le \tau$



The cases partition $\mathbb{R}$; strict convexity makes the minimizer unique.

soft-threshold: $S_\tau(v) = \text{sign}(v)\ \max(|v| - \tau, 0)$ — shrink toward zero by τ, and snap to **exactly zero** once $|v| \le \tau$

Solving Lasso, step 2: proximal gradient descent (ISTA) (added Sep 17)

repeat: ; $\mathbf{v} \leftarrow \mathbf{w} - \eta \nabla \hat{\mathcal{R}}(\mathbf{w})$ then $w_j \leftarrow S_{\eta\lambda}(v_j)$ for every j

A gradient step on the smooth loss, then the previous slide's exact minimizer with threshold $\eta\lambda$ (step size times penalty weight): the "proximal" step. Converges for the same step sizes as GD on $\hat{\mathcal{R}}$ alone. Known as ISTA (Daubechies, Defrise, De Mol, 2004).

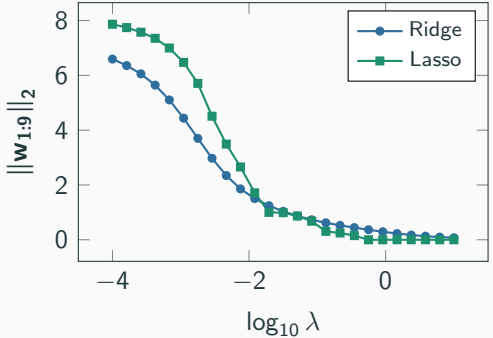
Ridge, one GD step on the penalty: $w_j \leftarrow (1 - \eta\lambda)\, w_j$ — multiplies by a factor in $(0, 1)$; a nonzero weight never becomes exactly zero

Lasso, one proximal step: $w_j \leftarrow S_{\eta\lambda}(v_j)$ — subtracts a constant $\eta\lambda$ and snaps to 0 once $|v_j| \leq \eta\lambda$; zeros are exact

The diamond-vs-circle picture at the level of a single update: the corner is the flat piece of S_τ. HW2 has you implement this.

Effect on the demo: Lasso hits exactly zero, Ridge never does

Same $M = 9$, $N = 30$ setup, same λ grid:



Lasso's curve hits exactly 0 and *stays* there; Ridge's keeps falling but never touches it.

At $\lambda = 0.0187$:

	Ridge	**Lasso**
$\|\mathbf{w}_{1:9}\|_2$	1.26	0.99
# exactly zero	0 / 9	6 / 9

at $\lambda = 1.23$: Lasso is fully sparse (9/9 zero); Ridge still nonzero everywhere

Lasso as feature selection: the surviving coefficients name the useful inputs

which coefficients **survive** at a given λ tells you which inputs the model judged useful

Ridge shrinks everything a little. Lasso shrinks some things to exactly nothing.

7. A probabilistic view: MLE and MAP

Regularization as a prior belief

Recap: MLE view of least squares

Lecture 02 showed both our losses are negative log-likelihoods in disguise: logistic loss under a Bernoulli model, squared error under Gaussian noise $\mathbf{y}_i = \mathbf{w}^\top \mathbf{x}_i + \varepsilon_i$, $\varepsilon_i \sim \mathcal{N}(0, \mathrm{Var}[\varepsilon])$ — the same noise model this lecture's bias-variance decomposition builds on.

MLE under Gaussian noise
$\iff$ **least-squares ERM**

Next question: what if we also have a prior belief about $\mathbf{w}$ itself?

From MLE to MAP: add a Gaussian prior on w

maximum a posteriori (MAP): add a prior $p(\mathbf{w})$, encoding a belief about $\mathbf{w}$ *before* seeing data, and maximize the posterior $p(\mathbf{w} \mid X, \mathbf{y}) \propto p(\mathbf{y} \mid X, \mathbf{w})\, p(\mathbf{w})$

MLE is the special case of a *flat* (uninformative) prior.

$\mathbf{w} \sim \mathcal{N}(0, \tau_w^2 I)$ — "small weights are more likely, a priori"

$$\log p(\mathbf{w}) = -\frac{1}{2\tau_w^2} \|\mathbf{w}\|_2^2 + \text{const}$$

The negative log-posterior: a squared-error term plus an ℓ_2 term

Maximize posterior $\Leftrightarrow$ minimize negative log-likelihood $+$ negative log-prior:

$$-\log p(\mathbf{w} \mid X, \mathbf{y}) = \frac{1}{2\mathrm{Var}[\varepsilon]} \sum_{i=1}^{n} (\mathbf{y}_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \frac{1}{2\tau_w^2} \|\mathbf{w}\|_2^2 + \text{const}$$

It matches Ridge's objective when $\tau_w^2 = \mathrm{Var}[\varepsilon]/(n\lambda)$

Multiply through by $\mathrm{Var}[\varepsilon]$ (positive, doesn't change the arg min) and compare to Ridge's $n\hat{\mathcal{R}}_\lambda(\mathbf{w})$: the two match term for term when

$$\tau_w^2 = \frac{\mathrm{Var}[\varepsilon]}{n\lambda}$$

Ridge regression is exactly MAP estimation under a Gaussian prior on w.

Worked check: λ sets the prior width

$$\lambda = 0.01,\ n = 15,\ \mathrm{Var}[\varepsilon] = 0.04 \implies \tau_w^2 = \frac{0.04}{15 \times 0.01} \approx 0.267$$

Larger λ = tighter prior (smaller τ_w^2) = a stronger belief that weights should stay near zero.

Regularization is a prior belief, encoded as if it were extra data.

Not an ad hoc trick — exactly MAP estimation under a natural, named prior.

8. Wrap-up

One U-shape explained, two warning signs repaired

Method comparison

Method	Penalty	Closed form?	Sparse?	View
Plain ERM	none	yes	—	MLE
Ridge (L2)	$\frac{\lambda}{2}\|\mathbf{w}\|_2^2$	yes	no	MAP, Gaussian prior
Lasso (L1)	$\lambda\|\mathbf{w}\|_1$	no	yes	(Laplace prior)

What we built, section by section

Section 1: bias — a class that cannot bend stays wrong with unlimited data

Section 2: feature engineering — a fixed ϕ enlarges the class and removes the bias

Section 3: too rich a class chases the sample — variance; test error is U-shaped in M

Section 4: the decomposition makes both exact — bias falls, variance rises, noise is a floor

Sections 5–6: Ridge and Lasso control variance without shrinking the class

Section 7: regularization is exactly MAP estimation under a natural prior

Both of Lecture 02's warning signs, closed by the same penalty

ill-conditioning: Ridge's eigenvalue-lifting proof — $X^\top X + n\lambda I$ is always invertible

separable-data weight divergence: the same $\frac{\lambda}{2}\|\mathbf{w}\|_2^2$ term caps $\|\mathbf{w}\|$ in logistic regression too

One fix, both loops closed.

Feature engineering makes Lecture 02's $d + 1 > n$ boundary a choice

Lecture 02: $X^\top X$ is invertible only when $n \geq d + 1$. Ridge (Section 5) assumed we were on the right side of that line, just close to it.

with $\phi(\mathbf{x}) \in \mathbb{R}^{\tilde{d}}$ in place of $\mathbf{x}$, the feature-space design matrix Φ (rows $\phi(\mathbf{x}_i)^\top$) plays X's role — so compare n to $\tilde{d} + 1$, not $d + 1$, and $\tilde{d}$ is whatever we picked ϕ to produce

$M = 9$, $N = 15$: $\tilde{d}+1 = 10 < 15$, underparameterized (today's demo)

$M = 20$, same $N = 15$: $\tilde{d}+1 = 21 > 15$, $\Phi^\top\Phi$ rank-deficient, on purpose

Same data. Our choice of ϕ decides which side of the line we are on — overparameterization becomes a **design choice**, not an edge case.

The catch: explicit features cost $O(\tilde{d})$, and the most useful maps are infinite

Forming $\phi(\mathbf{x})$ and computing $\mathbf{w}^\top \phi(\mathbf{x})$ costs $O(\tilde{d})$ per input.

dozens
fine (today's polynomials)

millions
painful (all pairwise interactions of a few thousand raw features)

infinite
impossible: $\phi(\mathbf{x})$ cannot be written down at all

Lecture 04 fixes this: kernels compute in feature space without ever forming $\phi(\mathbf{x})$, even at $\tilde{d} = \infty$ — and then run the experiment this lecture's theory says nothing about: what test error does *past* the $\tilde{d} + 1 > n$ line (double descent)

Next time

Lecture 04: Kernel Methods; Double Descent; Implicit Regularization — computing with ϕ without forming it, then crossing the $\tilde{d} + 1 > n$ line and finding test error can fall a second time

HW1 (Lectures 02–03) due Mon 9/14, 11:59pm.

References

- Hoerl, Kennard. *Ridge Regression: Biased Estimation for Nonorthogonal Problems.* Technometrics, 12(1), 1970.
- Tibshirani. *Regression Shrinkage and Selection via the Lasso.* Journal of the Royal Statistical Society, Series B (Methodological), 58(1), 1996.

Overfitting/bias-variance/regularization content adapted from EPFL's `ML_course`.

Full citation details: `references/sources/lecture03-ridge-lasso-citations.md`, `references/sources/epfl-l03-generalization-regularization.md`