Lecture 1c: Model Selection and the Curse of Dimensionality

Choosing complexity honestly, and why high dimensions are strange

Jue Guo

University at Buffalo

May 7, 2026

Outline

Hold-out and cross-validation

Information criteria

The curse of dimensionality

Why learning still works

Wrap-up

Where we are

Recap.

- 1a – polynomial order M and regularizer λ emerged as hyperparameters controlling effective complexity.
- 1b – the same λ showed up as the precision of a Gaussian prior. Either way, we still have to pick a value.

Today.

- How to set hyperparameters honestly when we have data: hold-out and cross-validation.
- How to estimate complexity penalties without re-training: AIC, BIC.
- Why doing any of this in many input variables is much harder than in one: the curse of dimensionality.
- Why machine learning works at all in spite of it.

The model selection problem

Given training data $\mathcal{D}$ and a family of models indexed by some complexity knob (M, λ, depth, hidden units, . . .), pick the value that gives the best *out-of-sample* performance.

What we already know.

- Training error keeps decreasing with capacity – useless as a selector.
- Test error is the U-shaped object we care about, but the test set must stay untouched.

So: we need a third dataset, or a clever proxy.

Hold-out: train / validation / test

With plenty of data, partition into three disjoint pieces:

- Training set – fits $\mathbf{w}$ for each candidate model.
- Validation set – compares candidates; pick the best.
- Test set – used *exactly once*, at the very end, to report generalization.

Why a separate test set? If we tune to the validation set hard enough, we will eventually overfit *it* – the chosen model's validation score is no longer an unbiased estimate of generalization. The untouched test set remains honest.

Discipline matters

Every time you peek at the test set, you spend a bit of its statistical power. Treat it like the only fair coin you have left.

When data is scarce

Hold-out wastes data: the training set is smaller than it could be, and a small validation set gives a noisy estimate of generalization.

Cross-validation idea. Use *all* the non-test data for training, and *all* of it for validation, by rotating which slice is held out.

Definition (S-fold cross-validation)

Partition the data into S groups. For $s = 1, \ldots, S$: train on the other $S - 1$ groups, validate on group s. Report the average validation error across all S runs.

Each example serves as a training point exactly $S - 1$ times and as a validation point exactly once.

S-fold cross-validation, illustrated

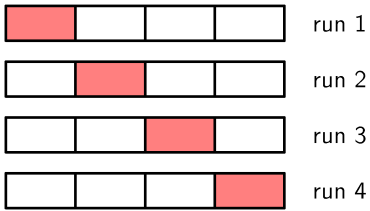

Figure 1.18

$S = 4$; *red* cells are validation in each run, *blue* cells are training. The cross-validation score is the mean of the four red-cell errors.

Special case. $S = N$ gives leave-one-out cross-validation (LOOCV) – maximally training-rich, often the lowest-bias estimator, sometimes feasible in closed form (e.g. for linear regression).

The downside of cross-validation

- Cost. Training time multiplies by S. For models that are already expensive (deep nets), even $S = 5$ hurts.
- Multiple hyperparameters. If we have k knobs, a grid sweep over them with S-fold CV at each cell is exponential in k.
- Validation noise. Each fold's score is itself a random variable; with small data the best fold may not be the truly best model.

Wishlist

A criterion that

- depends only on the training data,
- handles many hyperparameters in one pass,
- is honest about overfitting.

AIC – penalize parameter count

The Akaike information criterion chooses the model maximising

$$\mathrm{AIC}(\mathcal{M}) \;=\; \ln p(\mathcal{D} \mid \mathbf{w}_{\mathrm{ML}}, \mathcal{M}) \;-\; M, \tag{1}$$

where M is the number of free parameters in $\mathcal{M}$.

Interpretation. Best-fit log-likelihood, debited by one "unit" per parameter.

BIC. The *Bayesian information criterion* uses a heavier penalty:

$$\mathrm{BIC}(\mathcal{M}) \;=\; \ln p(\mathcal{D} \mid \mathbf{w}_{\mathrm{ML}}, \mathcal{M}) \;-\; \tfrac{1}{2} M \ln N. \tag{2}$$

Approximates the log model evidence under a Laplace approximation; we revisit it later.

Concrete example – AIC for $M = 3$ vs $M = 9$

Take the running curve-fitting setup: $N = 10$ noisy points, polynomial of order M, Gaussian noise of precision β_{ML}. Plug $\mathbf{w}_{\mathrm{ML}}$ into the Gaussian log-likelihood:

$$\ln p(\mathcal{D} \mid \mathbf{w}_{\mathrm{ML}}) \;=\; -\tfrac{N}{2}\big(1 + \ln(2\pi/\beta_{\mathrm{ML}})\big).$$

Suppose the residual variances are $1/\beta_{\mathrm{ML}} = 0.073$ at $M = 3$ and $1/\beta_{\mathrm{ML}} = 2.5 \times 10^{-5}$ at $M = 9$ (essentially zero training error).

	$M = 3$ (4 params)	$M = 9$ (10 params)
$\ln p(\mathcal{D} \mid \mathbf{w}_{\mathrm{ML}})$	−3.62	+47.10
parameter penalty M	4	10
AIC $= \ln p(\mathcal{D} \mid \mathbf{w}_{\mathrm{ML}}) - M$	**−7.62**	**+37.10**

AIC picks $M = 9$ – the catastrophic overfit. The exploding log-likelihood ($\beta_{\mathrm{ML}} \to \infty$ as residuals shrink) overwhelms the modest parameter-count penalty. Cross-validation, which actually inspects out-of-sample error, avoids the trap.

Why information criteria disappoint

Both AIC and BIC count parameters, not their effective constraint.

- A regularised model with $M = 9$ coefficients but λ large uses far less than 9 effective parameters; AIC/BIC don't see this.
- In practice they tend to favour *overly simple* models – an under-correction for overfitting.
- They ignore parameter *uncertainty*, which is exactly what a Bayesian treatment captures.

The principled fix (preview)

Replace the ad hoc penalty by the *model evidence* $p(\mathcal{D} \mid \mathcal{M}) = \int p(\mathcal{D} \mid \mathbf{w}, \mathcal{M})\, p(\mathbf{w} \mid \mathcal{M})\, d\mathbf{w}$. We come back to this in the chapter on linear models.

From one input to many

The curve-fitting example had a single input $x \in \mathbb{R}$. Real problems don't:

- 28×28 grayscale digit $\Rightarrow D = 784$.
- Modest tabular dataset $\Rightarrow D = 10$-50.
- ImageNet-scale image $\Rightarrow D \sim 10^5$.

Bellman, 1961. Many techniques that work in 1-2 dimensions degrade sharply as D grows; he gave the failure mode a name: the curse of dimensionality.

Two flavours, both on the next slides:

- Counting: parameter / cell counts blow up.
- Geometry: high-dimensional volumes don't behave like our 3D intuition says.

Counting cells: M^D

A naive nearest-neighbour-by-cell classifier divides each axis into M bins.

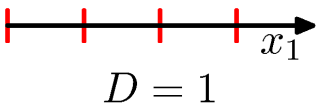

Figure 1.21a

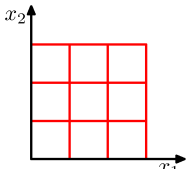

Figure 1.21b

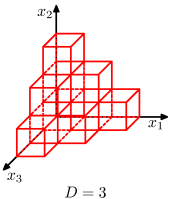

Figure 1.21c

Cell count M^D grows exponentially in D. At $M = 10$, $D = 10$ we already have 10^{10} cells – a fixed training set is hopelessly sparse.

Counting polynomial coefficients

A polynomial of order M in D inputs has

$$\binom{M+D}{D} = O(D^M) \text{ as } D \to \infty \tag{3}$$

independent coefficients.

- $D = 10,\ M = 3$: $\binom{13}{10} = 286$ coefficients.
- $D = 10,\ M = 5$: $\binom{15}{10} = 3003$ coefficients.
- $D = 100,\ M = 3$: $\binom{103}{100} = 176{,}851$ coefficients.

Power-law growth is gentler than exponential, but expressive parametric models still become unaffordable long before the data does. This is one of the reasons we eventually trade explicit basis expansions for *learned* features (kernels, neural networks).

High-dimensional volume hugs the surface

Volume of a unit ball: $V_D(r) = K_D r^D$. Fraction within distance ε of the surface,

$$\frac{V_D(1) - V_D(1-\varepsilon)}{V_D(1)} = 1 - (1-\varepsilon)^D. \tag{4}$$

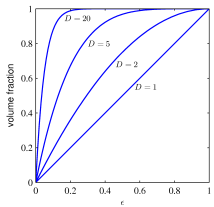

Figure 1.22

For large D, the fraction tends to 1 even for small ε. Almost all of a high-dimensional ball's volume sits in a thin shell near the surface. (PRML Fig. 1.22.)

Even Gaussians live in a thin shell

Switch a standard D-dim Gaussian to spherical coordinates and integrate out the angles:

$$p(r) \;=\; \frac{r^{D-1}}{2^{D/2-1}\,\Gamma(D/2)}\, e^{-r^2/2}, \qquad p(r)\,\delta r \approx \Pr(\|\mathbf{x}\| \in (r, r+\delta r)). \tag{5}$$

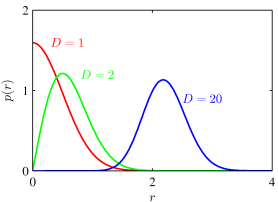

Figure 1.23

For $D = 20$, the density peaks at $r = \sqrt{D-1} \approx 4.36$, nowhere near the origin – the probability mass sits in a thin shell far from the mode. Pairwise distances concentrate, so naive distance-based methods (k-NN, k-means, fixed-bandwidth RBFs) lose their resolving power. (PRML Fig. 1.23.)

Real data is not uniform in $\mathbb{R}^D$

Manifold hypothesis. Real data typically concentrates on a low-dimensional *manifold* embedded in the ambient input space.

- A 28×28 digit image lives in $\mathbb{R}^{784}$, but the set of "handwritten 7s" has only a few tens of degrees of freedom (slant, stroke thickness, identity).
- A 1024×1024 photo has $\sim 10^6$ pixels but only a handful of underlying scene parameters.

The intrinsic dimensionality d of the data manifold can be $\ll D$. Methods that find this manifold (or behave well on it) sidestep the worst of the curse.

Smoothness lets us interpolate

Second escape: target functions are usually smooth. Small changes in the input lead to small changes in the target – so information from one labelled point transfers to its neighbours. **How successful methods exploit one or both properties.**

- Kernel methods. Encode smoothness via the kernel; the choice of kernel *is* a choice of smoothness assumption.
- Local methods (k-NN, splines). Assume smoothness; degrade gracefully when intrinsic dimension is low.
- Deep networks. Learn the manifold and the function on it jointly; rich representation learning rather than fixed bases.
- Linear models on engineered features. A short cut: hand-pick features that align with the manifold.

Takeaways

- Use the train / validation / test discipline; spend the test set last.
- S-fold cross-validation buys honest hyperparameter selection when data is scarce; LOOCV is the limit $S = N$. Pay in compute.
- AIC/BIC are cheap but blunt – they count parameters, ignore regularisation. Bayesian model evidence is the principled successor.
- The curse of dimensionality attacks both counting (cells, parameters) and geometry (shells dominate volumes; distances concentrate).
- It does not stop us, because real data is low-dimensional and target functions are smooth – exploit one or both.

Reading

Bishop, PRML (2006), §§1.3-1.4.