Week 10 — Bias-Variance & Bayesian Nonparametrics

Friday, July 3, 2026 · async / recorded

Prof. Joseph Austerweil

This week is async

- **No in-person session** — this is the recorded Bayesian-nonparametrics lecture. Watch at your own pace; pause and play with the widgets.
- **Reflections:** Week 10 is reflection-eligible — post in the discussion thread.
- **Due Fri Jul 10, 8 PM:** the Reinforcement Learning assignment **and** the Monte Carlo assignment.
- Four interactive widgets are linked from the slides — open them and click around; that’s the point of recording it.

Agenda

Bias-variance → double descent — how complex should a model be?

Finite mixtures — the “how many clusters?” problem

The Chinese Restaurant Process (CRP)

The Dirichlet Process — three lenses on one object

DP mixture models — let the data choose K

One machine, many models — LDA, features

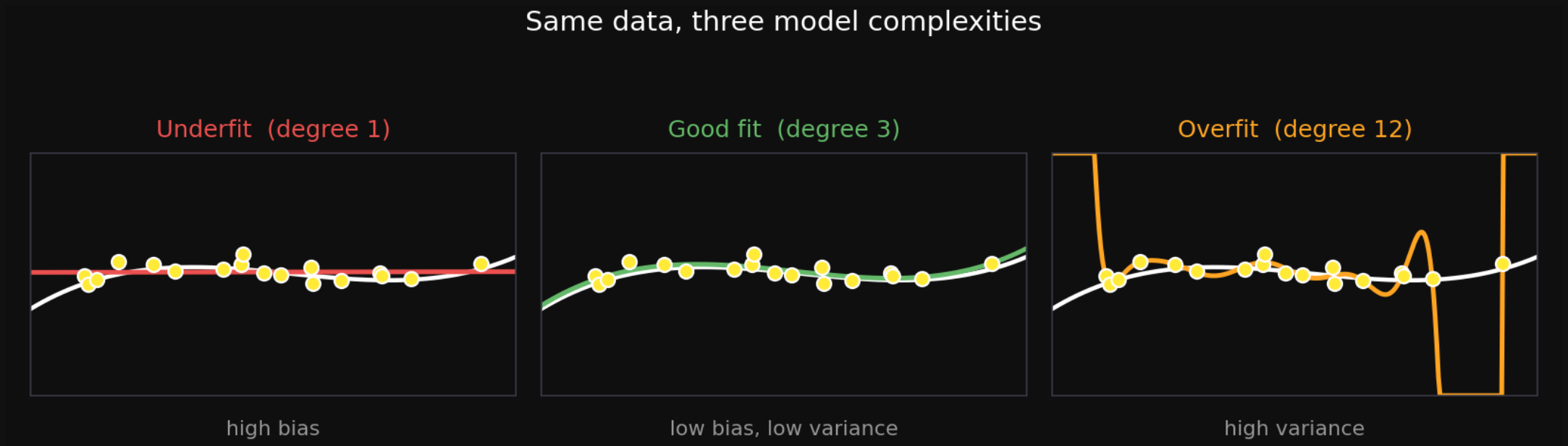
Gaussian processes & the modern tail (NNGP/NTK, RAG)

The one picture — parametric → nonparametric → BNP

How complex should the model be?

The setup: pick a model's complexity

We have noisy data and want to learn the mapping behind it. We must choose a **hypothesis space** — how flexible the model is allowed to be. Here: a polynomial of **degree 1, 3, or 12**.



Two ways to be wrong

Training error

- error on points the model has **already seen**
- **always falls** as complexity rises — a flexible model can memorize
- *not* the thing to optimize

Prediction (test) error

- error on **new**, unseen data
- the thing we **actually** care about
- rises again once the model overfits

The gap between them is **overfitting** — the model memorized noise instead of signal.

The bias-variance decomposition

$$\mathbb{E}\big[(\hat{y} - y)^2\big] \;=\; \mathrm{Bias}^2 \;+\; \mathrm{Variance} \;+\; \sigma^2$$

- $\hat{y}$ the prediction, y the truth, $\mathbb{E}$ an average over re-drawn training sets.
- **Bias²** — the systematic miss of the *average* fit (too-simple models).
- **Variance** — how much the fit *wobbles* across datasets (too-complex models).
- σ^2 — irreducible noise, which no model can remove.

Geman, Bienenstock & Doursat (1992)

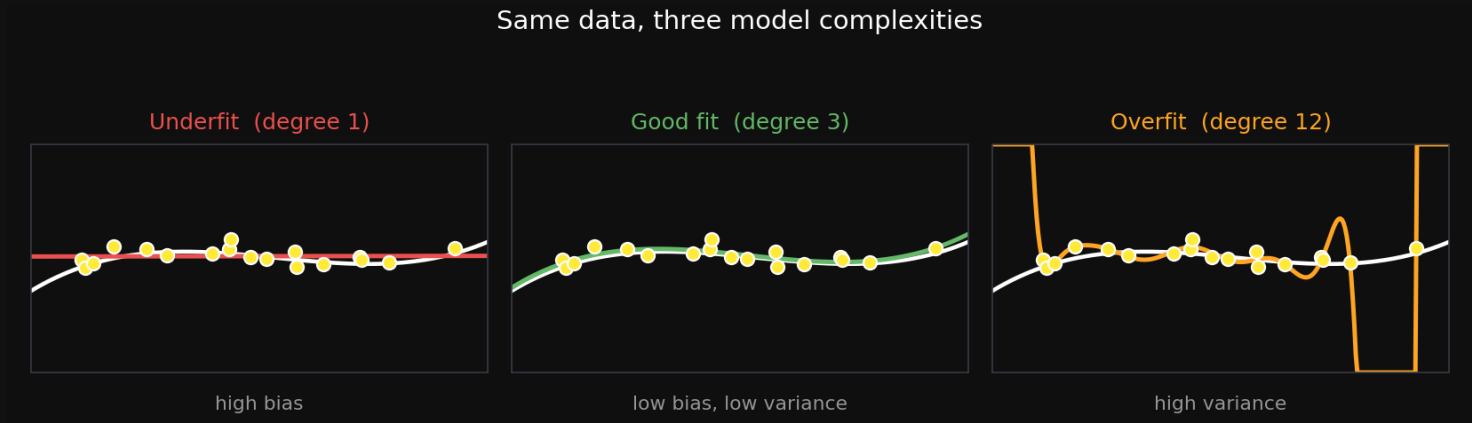
High bias vs. high variance

High bias (too simple)

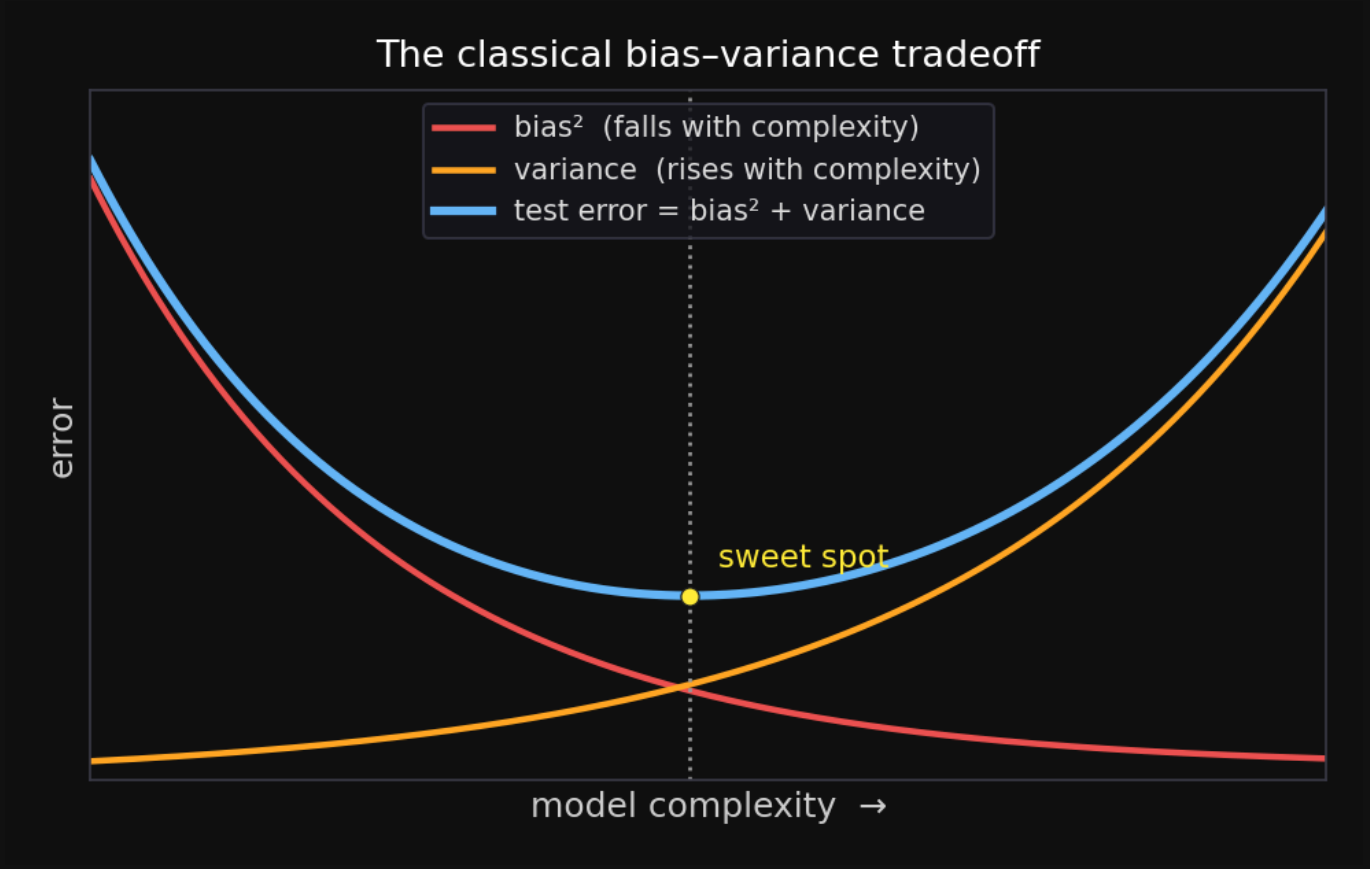
- average fit misses the curve
- but it barely moves if you resample
- *underfitting*

High variance (too complex)

- average fit can track the truth
- but each fit swings wildly with the data
- *overfitting*



The classical moral: the U-curve



Bias falls, variance rises → their sum is minimized at an **intermediate** complexity.

For decades this was the whole story: there's a sweet spot, and going past it **always hurts**.

Meet the controls

The data

- **training points** n — how many noisy examples we learn from. *More data ⇒ less variance.*
- **noise** σ — the spread of the random scatter added to the true signal. *Bigger σ ⇒ noisier data, easier to overfit.*

The model

- **polynomial degree** — its flexibility, i.e. **capacity**. *Low = stiff (underfit); high = wiggly (overfit).*
- **ridge** λ — a penalty on large coefficients that pulls the fit toward something tamer: a **regularizer**. $\lambda = 0$ = *off; bigger λ = simpler fit.* (Returns after the break.)

Play with it — classical regime

Left: drag **polynomial degree** — the per-dataset “spaghetti” fans out as variance rises. **Right:** the bias²/variance/test curve, shown up to $p = n$ — the classic U. Raise n and the spaghetti tightens (variance ↓).

Predict before we continue

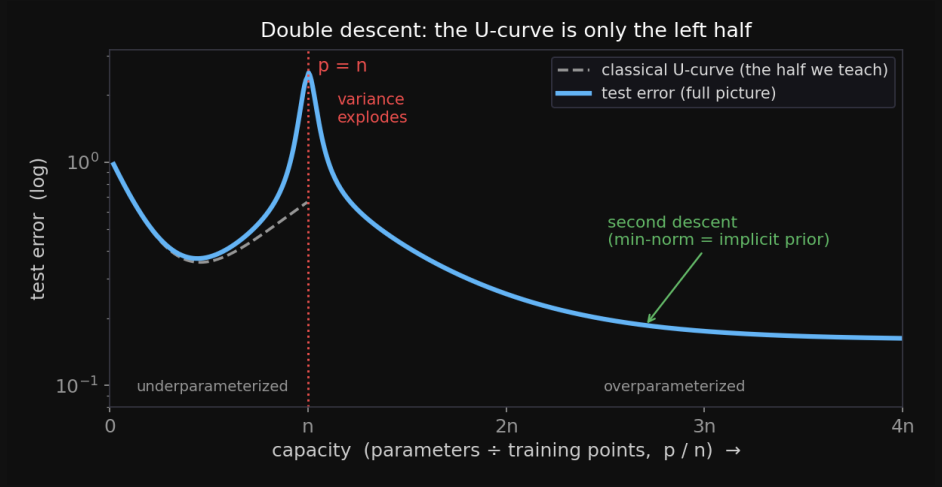
Keep adding parameters **past** the point where the model can fit every training point exactly. What happens to **test** error?

A. keeps rising — more capacity, more overfitting

B. flattens out

C. rises to a peak, then falls again

Double descent — the U-curve is half the story



Past the **interpolation threshold** ($p = n$: as many parameters as data points), test error spikes — then **descends a second time**, often below the classical sweet spot. The U-curve is the *left half*.

Belkin et al. (2019, PNAS); Nakkiran et al. (2019)

Why the spike at $p = n$?

Right panel, “closed-form min-norm”: drag **capacity p** to n. The test curve spikes — and the purple **‖weights‖₂** panel spikes with it.

Past the threshold: which fit?

With **more parameters than data**, there are **infinitely many** fits that nail the training points exactly.

So the question changes:

> *not “can it fit?” but “which perfect fit?”*

Gradient descent from zero (and the pseudoinverse) picks the **minimum-norm** fit — the one with the **smallest weights**.

Smallest weights → smoothest → **generalizes**.

That choice is a prior.

Norm ↔ prior: the dictionary

ℓ_2 norm — $\|\beta\|_2^2$ *(squared Euclidean length)*

⇕

a **Gaussian prior** on the weights

→ **ridge regression**

ℓ_1 norm — $\|\beta\|_1$ *(sum of absolute values)*

⇕

a **Laplace prior** on the weights

→ **lasso** → *sparsity*

Gradient descent from zero converges to the minimum-**ℓ_2** fit — the **Gaussian**-prior one. *(Not L1.)*

Ridge: the explicit twin

Drag **ridge λ** up: the spike at $p = n$ collapses (≈337 → ~1) — explicit ℓ_2 is the *twin* of the implicit min-norm bias. A moderate λ gives the lowest test of all; over-shrink and you under-fit (the best test creeps back up).

Early stopping: the implicit twin

Selector → **“neural net + GD”**. Sweep **hidden units H**; set **GD iterations T**. Small T → flat, no spike (early stopping); large T → the spike returns at $H = n$.

Benign overfitting & the honest caveats

- **Benign overfitting** (Bartlett et al. 2020): in high dimensions a model can interpolate the *noise* and still generalize — the noise spreads thin across many directions.
- **Caveat:** double descent needs the right regime — high-dimensional, roughly isotropic features. Low-dim correlated features (the widget's *left* panel) don't show it.
- The classical U-curve was never *wrong* — just **incomplete**.

So — how complex should the model be?

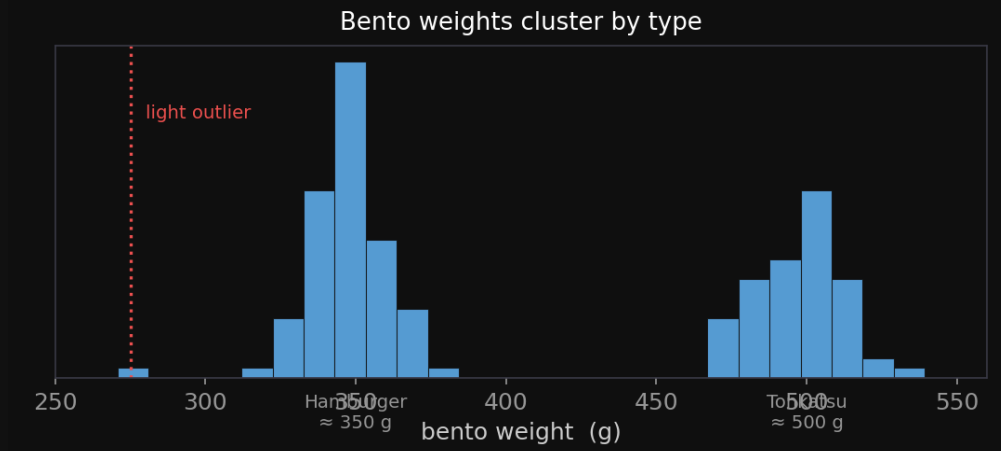
Two principled modern answers:

1. **Go huge, then regularize** — make the model enormous and let an implicit/explicit prior pick a simple interpolant (the deep-learning answer we just dissected).
2. **Let the model grow with the data** — under an explicit prior that says how eagerly to add complexity. That is **Bayesian nonparametrics** — the rest of today.

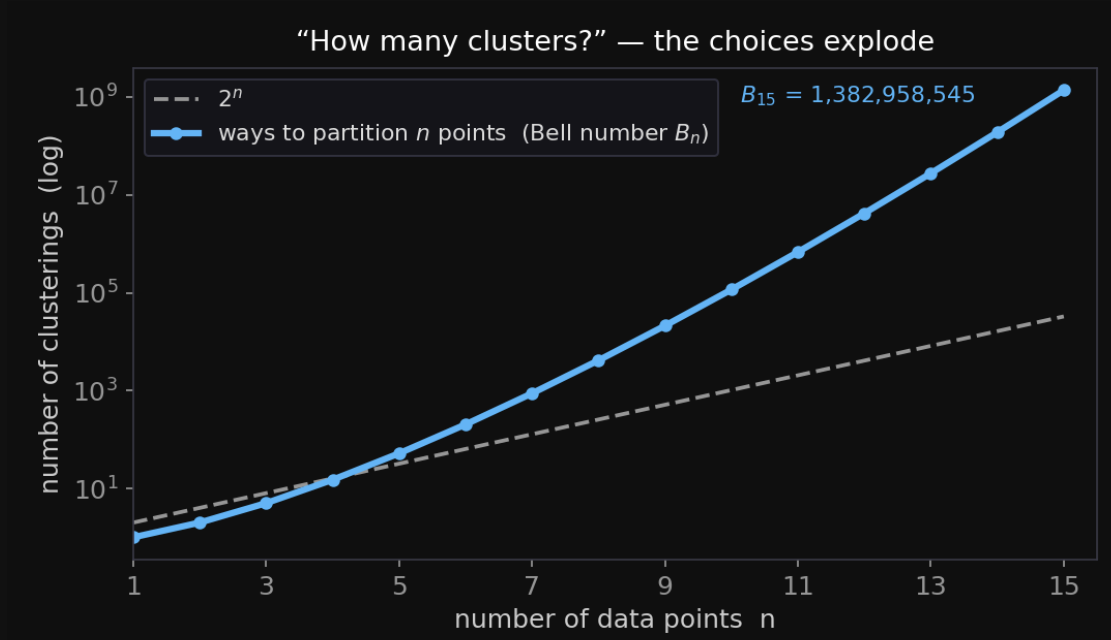
Let the data decide: Bayesian nonparametrics

Recall: clustering bento weights

From the mixture-models chapter: bento weights cluster by type — **Hamburger** (≈ 350 g) and **Tonkatsu** (≈ 500 g). A **Gaussian mixture model (GMM)** gives each cluster a Gaussian and a weight, and treats *which cluster each point came from* as a **latent variable to infer** — we reason about the posterior, we don't solve it in closed form. But you must still **fix K** (the number of clusters) up front.



The “how many clusters?” problem



You can’t just try every clustering: the number of ways to partition n points (the **Bell number**) explodes — for 15 points it’s already **1.4 billion**. And the “right” K may grow as more data arrives.

Watch fixed-K miss a cluster

Set **GMM K = 2**. The lone light bento (≈275 g) gets swallowed into the low cluster — its mean is dragged down. The **DPMM** (blue) gives it its own cluster.

The Chinese Restaurant Process

A law over partitions

A restaurant with infinitely many tables; customers enter one at a time. Customer $n+1$:

- joins an occupied table k with probability $\propto n_k$ (people already there),
- starts a **new** table with probability $\propto \alpha$.

That's the **Chinese Restaurant Process (CRP)** — a law over *partitions* with no fixed K. α is the **concentration**: the appetite for new tables.

$$P(\text{partition}) \;=\; \frac{\alpha^{K}\,\prod_{k=1}^{K}(n_k-1)!}{\alpha\,(\alpha+1)\cdots(\alpha+n-1)}$$

K = occupied tables, n_k = people at table k, n = total customers. **Exchangeable** — it depends only on the block sizes, not who arrived when.

Seat the customers — live

Auto-seat customers and watch tables appear. The bar shows the next customer’s odds (∝ table size, plus NEW ∝ α). Big tables pull harder — **rich-get-richer**.

Rich-get-richer, and what α does

- **Rich-get-richer:** big clusters attract more members (the $\propto n_k$ rule). Clusters that start big tend to stay big.
- **Small** $\alpha \rightarrow$ few big clusters. **Large** $\alpha \rightarrow$ many small clusters.
- The number of clusters grows slowly with data — about $\alpha \log n$ — it is **never fixed**.
- This *growing-but-controlled* capacity is the **nonparametric answer** to Block 1 — capacity set by the data, not chosen by us.

From the CRP to the Dirichlet Process

Three lenses on one object

The **Dirichlet Process (DP)** is one object you can look at three ways:

- **CRP** — the distribution over *partitions* (customers → tables).
- **Pólya urn** — the *predictive rule*: how the next draw behaves once the random distribution is integrated out.
- **Stick-breaking** — an *explicit construction* of the random distribution itself.

Same DP; pick the lens that fits the job.

Pólya urn = the DP's predictive rule

- Draw colored balls from an urn: each draw **repeats an existing color** (∝ its count) or **invents a new color** (∝ α).
- This is **Blackwell–MacQueen (1973)** — the DP's **predictive marginal**, after integrating the random distribution G out.
- It is the **same rule as the CRP**, now with a *dish* (a parameter θ_k) attached to each table.
    - In the bento demo, that dish was the **mean of a Gaussian cluster** — each new table gets its own center θ_k, and that table's points are noisy draws around it.
- It is *not* the DP itself — it's what you see when you **only watch the draws**.

Stick-breaking — build the distribution

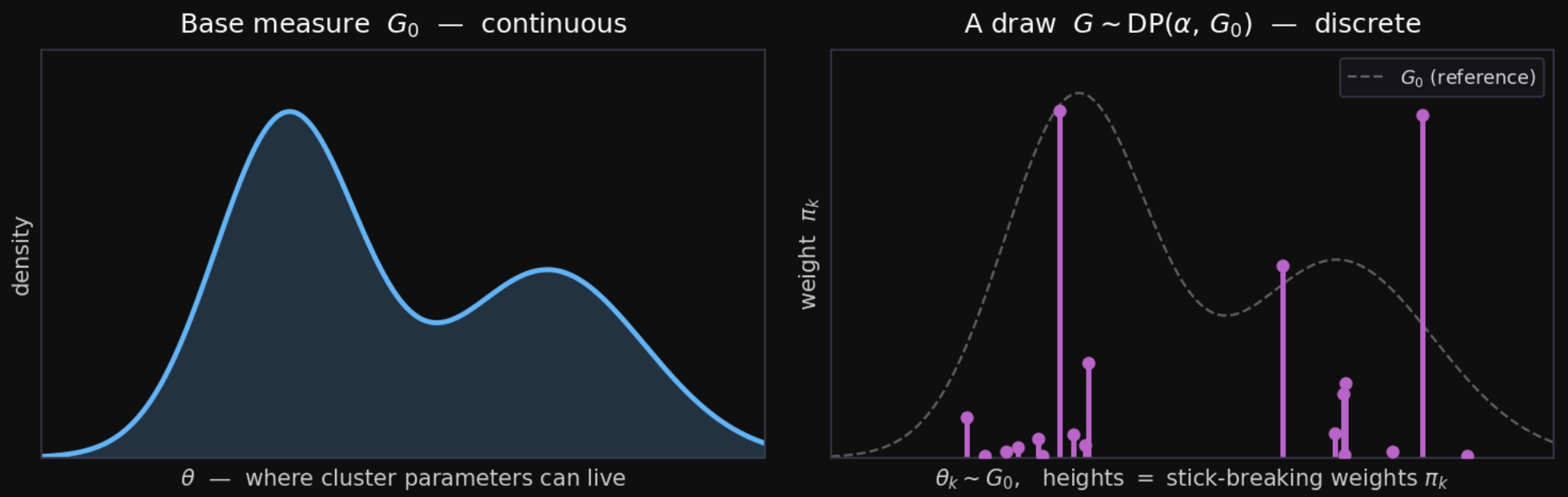
Take a unit stick; break off $\beta_k \sim \mathrm{Beta}(1, \alpha)$ of what remains each time. The pieces are the cluster weights π_k. Drag **α**: small α → a few big pieces; large α → many small ones.

The Dirichlet Process itself

$$G \sim \mathrm{DP}(\alpha, G_0)$$

- G is a **random probability distribution**; G_0 is the **base measure** (where clusters live); α the concentration.
- A draw G is **almost surely discrete** — a weighted set of spikes ($\sum_k \pi_k \delta_{\theta_k}$, the stick-breaking weights at locations from G_0).
- **Discrete → repeats → clustering.** That spikiness is *why* the DP groups data.

G_0 in → a discrete G out



Sample locations θ_k from the smooth G_0, then give them **stick-breaking weights** π_k. The draw G is a handful of **weighted spikes** — discrete, so repeated draws collide and **cluster**.

The trio, tied together

- **Stick-breaking** builds the random discrete G (the weights π_k and locations θ_k).
- **CRP** is what the *partition* looks like once you draw data through G and integrate G out.
  - *Handwave:* do that integration over n draws and you recover **exactly** the partition law $P = \dfrac{\alpha^K \prod_k (n_k - 1)!}{\alpha(\alpha+1)\cdots(\alpha+n-1)}$ from the CRP slide.
- **Pólya urn** is the *sequential predictive rule* you get at the same time.

One DP, three faces — and we never fixed the number of clusters.

Dirichlet Process Mixture Models

DP prior + Gaussian likelihood

A **DP mixture model (DPMM)**:

1. Draw a random discrete $G \sim \mathrm{DP}(\alpha, G_0)$ — the clusters and their weights.
2. For each data point, pick a cluster from G and draw the point from a **Gaussian** around that cluster's mean.

The CRP says how points share clusters; the Gaussian says what a cluster looks like. **K is inferred, not fixed.**

DPMM vs KDE vs GMM — live

Three density models on the same clicks: **DPMM** (infers clusters), **KDE** (a bump per point, no clusters), **GMM** (you fix K). Drag DPMM's **α**; add points and watch clusters appear.

Parametric / nonparametric / BNP

Parametric *(GMM, fixed K)*

- fixed # of parameters
- capacity set in advance
- e.g. “exactly 2 clusters”

Nonparametric *(KDE)*

- complexity grows with data
- but **no model / no prior** — just smoothing

Bayesian nonparametric *(DPMM)*

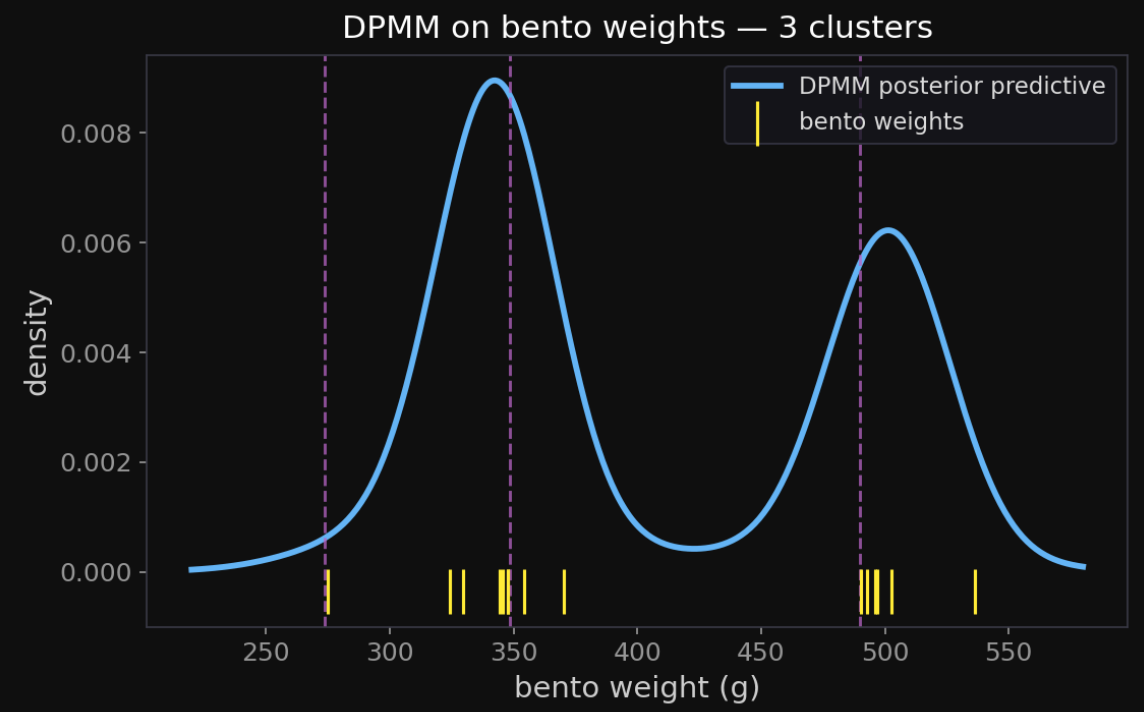
- complexity grows with data
- **and** a generative model + prior (α)

How is a DPMM actually fit?

- **Collapsed CRP Gibbs** (Neal 2000): reseat one point at a time — exactly the "customer picks a table" rule the widget runs. Intuitive, exact-ish, slow.
- **Variational / truncated stick-breaking** (Blei & Jordan 2006): cap the clusters and optimize — the scalable default (what scikit-learn's `BayesianGaussianMixture` ships).
- Honest note: as a *clustering* tool the DPMM is now **niche** — but the DP as a *building block* is everywhere (next block).

The DPMM, as a GenJAX program

```python
@gen
def dpmm(alpha, mu0, sig0, sigx):
    # stick-breaking → a prior over clusters
    betas = [beta(1., alpha) @ f"beta_{k}"
             for k in range(K)]
    pis = stick_break(betas)        # weights
    mus = [normal(mu0, sig0) @ f"mu_{k}"
           for k in range(K)]
    for i in range(N):              # each bento:
        z = categorical(pis)     @ f"z_{i}"
        x = normal(mus[z], sigx) @ f"x_{i}"
```



~10 lines of GenJAX *is* the model; a slice-Gibbs sampler then infers **K from the data — 3 clusters** here, the lone **275 g** bento on its own table. Full script: `genjax_dpmm.py`.

One machine, many models

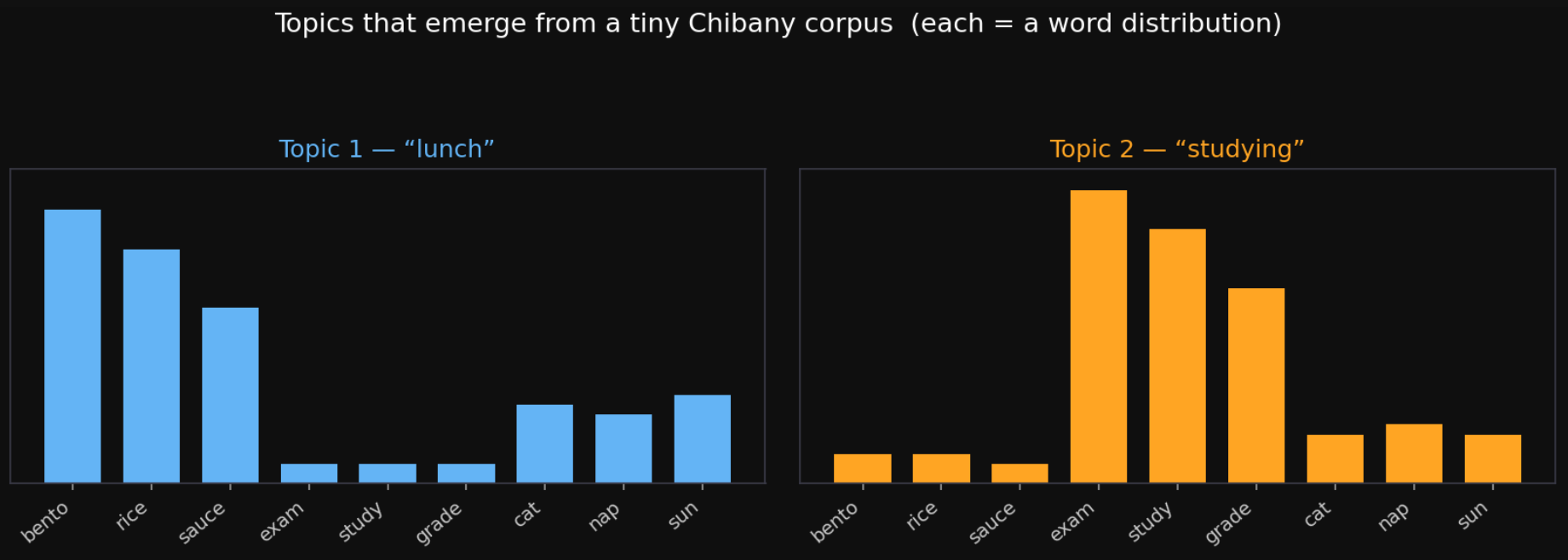
The building-block recipe

The move is always the same:

> *Take a model with a **finite parameter** (a fixed list of clusters / topics / features) and **replace it with a random measure** (a DP, or a cousin) so that list can **grow with the data**.*

- cluster means → **DPMM** *(this lecture)*
- topics → **HDP topic models**
- features → **Indian Buffet Process**
- the **likelihood stays the same** — only the prior over “how many” changes

Topics emerging from a corpus



A **topic** is a distribution over words. Given a tiny corpus, the model discovers that documents mix a **“lunch”** topic (bento, rice, sauce) and a **“studying”** topic (exam, study, grade) — without being told the topics in advance.

LDA, and its nonparametric version

LDA — *parametric* (Blei, Ng & Jordan 2003)

- each document = a mix of topics
- each topic = a distribution over words
- **K (number of topics) is fixed**
- a *finite-Dirichlet* model

HDP-LDA — *nonparametric* (Teh et al. 2006)

- apply the recipe to LDA
- a shared top-level DP = a **reusable menu of topics**
- number of topics **grows with the data**

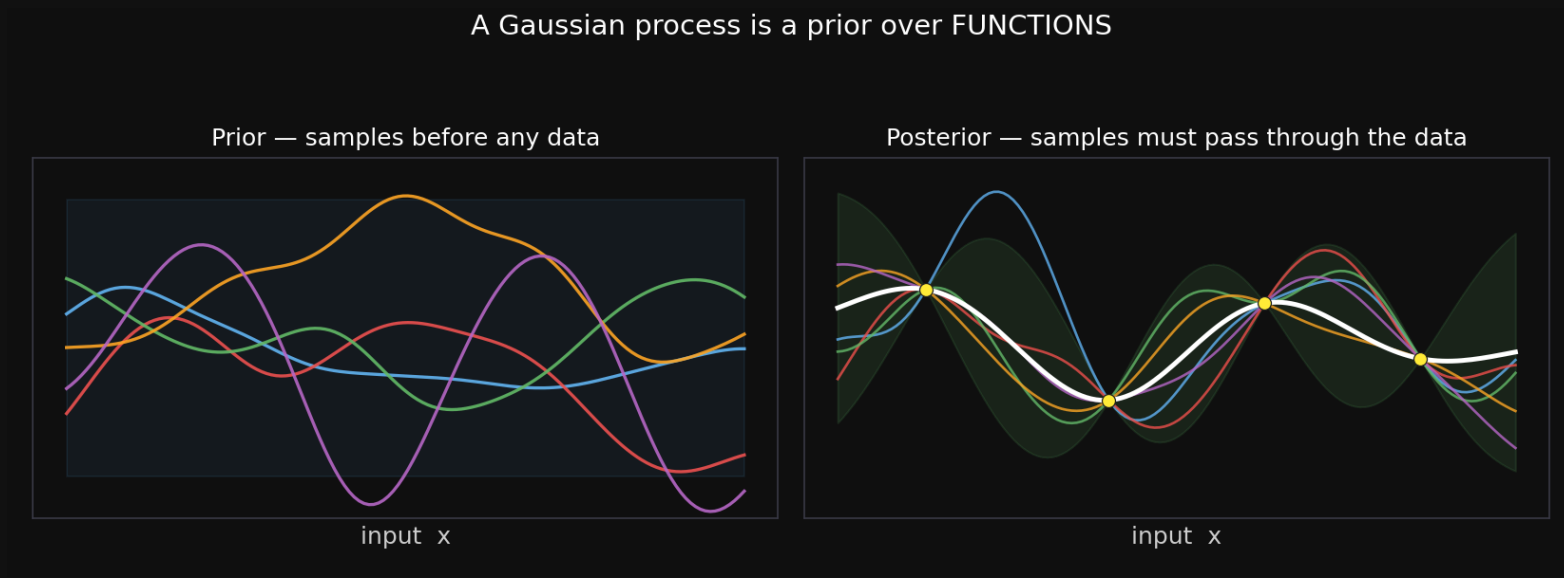
The feature sibling: the IBP

The DP gives each item **one** cluster. What if an item can have **many** features at once (a face has glasses *and* a hat *and* a beard)?

- The **Indian Buffet Process (IBP)** is the feature version: a prior over binary item-by-feature matrices, with the number of features **unbounded** (Griffiths & Ghahramani; the instructor's own line of work).
- Same spirit, different finite parameter. *(Cousins: HDP-HMM for sequences, Pitman–Yor for power-law vocabularies.)*

Gaussian processes & the modern tail

A prior over functions

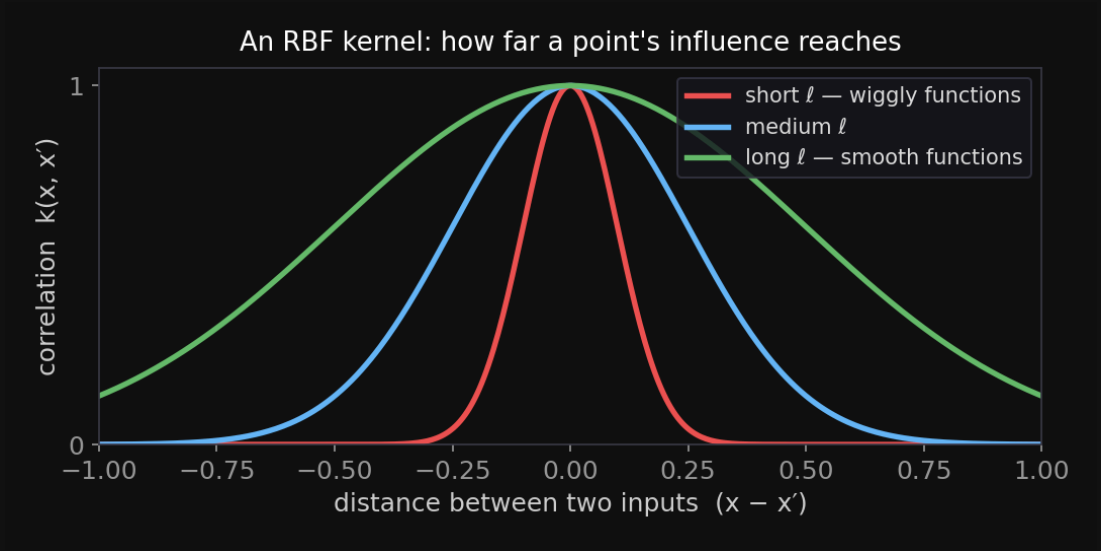


A **Gaussian process (GP)** is a prior over **functions**. Before data, samples wiggle everywhere (left). Condition on a few points and every sample must pass through them; uncertainty shrinks near data, grows far from it (right). Capacity grows with data — nonparametric, for regression.

Kernels — the shape of “nearby”

A GP is defined by its **kernel** $k(x, x')$ — how strongly two inputs’ outputs are correlated. A common choice ties nearby inputs to similar outputs (smoothness).

Callback: this is the **same kernel idea** as the KDE bandwidth in Widget 1 — “how far does the influence of a point reach?”



GP → neural networks

- **Neal (1996):** a one-layer net with infinitely many hidden units, random weights, **is exactly a GP**.
- **NNGP (Lee et al. 2018):** the same holds for *deep* nets at initialization — a deep net = a GP with a particular kernel.
- **NTK (Jacot et al. 2018):** while a very wide net is *trained by gradient descent*, it behaves like a (different) fixed-kernel method.
- **NNGP** = the net *at init* (Bayesian inference is a GP); **NTK** = the net *during training*. Caveat: both omit feature learning.

Nonparametrics in today’s AI

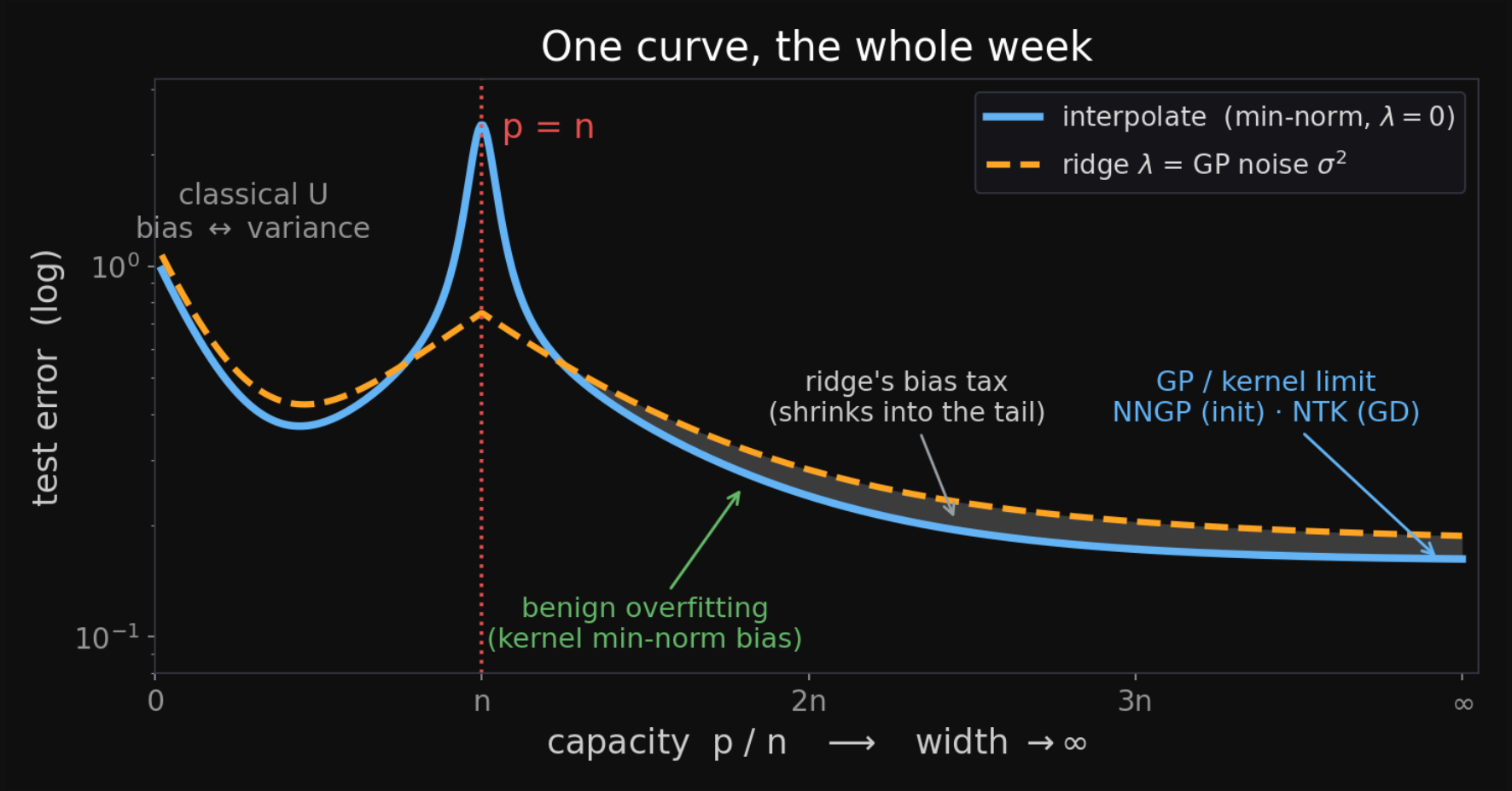
- **Neural processes** (Garnelo et al. 2018): neural nets trained to do GP-like prediction with uncertainty, at scale.
- **Nonparametric memory: RAG** and **kNN-LM** (Lewis et al. 2020; Khandelwal et al. 2020) let a model **look things up** in a store that grows with data — capacity in the *data*, not the weights.
- Through-line: *let capacity grow with the data* — the idea scaling laws gesture at.

The one picture

Parametric → nonparametric → BNP

- We asked: **how complex should the model be?** The Bayesian-nonparametric answer is *don't fix it — put a prior over all complexities and let the data decide.*
- **One machine** — a prior over partitions / measures / functions — reused everywhere: clusters (DPMM), topics (HDP), features (IBP), functions (GP).
- The concentration α (or the GP kernel) is the dial that says *how eagerly* to grow.

It all comes home: one curve, the whole week



The grey gap is **ridge's bias tax**: a little **bias** kills the **variance** spike at $p = n$.

One question, three answers — the GP holds all three

- **GP = kernel ridge.** posterior mean $= k_*^\top (K + \sigma^2 I)^{-1} y$ — the noise σ^2 ***is* the ridge** λ (the explicit twin from Block 1).
- **Kernel = inductive bias.** the lengthscale sets your spot on the bias–variance curve; the **marginal likelihood** picks it $\rightarrow$ *let the data decide* (the BNP thesis).
- **Wide net = GP.** NNGP (at init) · NTK (under GD): the **second-descent floor is the kernel limit**, and benign overfitting is the kernel's min-norm bias.
- **Three answers, one question.** regularize (**ridge**) · grow capacity (**BNP**) · prior over functions (**GP**) — and the GP contains all three.

Where this leaves us

- Bias-variance is **half** the story; double descent + implicit/explicit priors complete it — and that *prior* is the thread into Bayesian nonparametrics.
- BNP: **let the model grow with the data**, under a prior with one interpretable dial.
- Next week: **contemporary ML** — where these ideas (priors, capacity, scale) meet large models.
- *Play with the four widgets — that's the homework that isn't graded.*