

Bayesian Optimization

Put the belief to work — act to learn, and to win

Jinkyoo Park · KAIST

The first policy in the course — a rule that turns a belief into an action

Where we are — belief stops observing and starts choosing

PREVIOUS STEP

THIS CHAPTER'S QUESTION

Lectures 2–3 supplied probability models and conditional updates.

Now choose the next expensive measurement.

An unknown objective can be observed at selected inputs. Fit a GP, choose with an acquisition rule, measure and update.

Choose the next experiment from a stated acquisition rule and weigh it against the budget.

Learning route — predict, choose, pay for one measurement

Bring: Gaussian conditioning and regression from Lectures 2–3.

FIRST PASS	WHAT TO DO
Follow the idea	GP update → acquisition choice → evaluate → update again
Work without the solution	Choose the next experiment from a stated acquisition rule and weigh it against the budget.
Return later	Multi-output kernels, and how EHVI is computed beyond two objectives, are references.

For the temperature thread: **predict** → **calculate** → **reveal and check** → **change one condition**. Complete the core calculation before reading the research extensions.

— ACT 1

A belief over an unknown function

Q1. You cannot afford to probe f everywhere. So carry a distribution over the functions it might be.

The setting — unknown, expensive, and every query counts

Lecture 1 assumed we could evaluate the objective and use its mathematical structure. Here a new evaluation is expensive, and a formula or gradient for the true objective is unavailable.

$$x^* = \arg \max_x f(x), \quad f \text{ unknown and expensive}$$

WHAT "UNKNOWN" COSTS YOU

No formula, so no gradient. No gradient, so none of Lecture 1's machinery applies directly — the only thing you may do with f is **ask it a question and receive a number**.

WHAT "EXPENSIVE" COSTS YOU

One evaluation is a wet-lab experiment, a multi-hour CFD run, a clinical trial, a season of a wind farm. You get **tens** of queries, not millions. **Every query must be spent deliberately.**

The setting makes an adaptive loop useful: spend the next expensive evaluation using what earlier evaluations taught us. A fixed experimental design is another option; BO uses feedback to target promising or informative regions. Lecture 5 considers the separate restriction that no new evaluations are allowed.

The surrogate — a distribution, not a fitted curve

Q1 Model an unknown f ?

Q2 What is a GP?

Q3 Where next?

Q4 What lies beyond?

From a handful of evaluations, build a **surrogate**: a cheap stand-in for f that can be evaluated anywhere. The usual move is to fit one curve. The Bayesian move is to keep **all the curves the data has not ruled out**.

A FITTED CURVE GIVES YOU

one number at every x — a prediction. Where the data is dense and where it is absent, it looks equally confident. It cannot tell you where it is guessing.

A DISTRIBUTION GIVES YOU

a number **and a width** at every x . Narrow where data is dense, wide where data is sparse, and returning to the prior width far from everything.

That width is not decoration. It is **the signal we will use to decide where to look next**.

A function on a finite domain is a vector

The idea that makes a distribution over functions ordinary

Let $\mathcal{X} = \{x_1, \dots, x_n\}$ be finite, and let $\mathcal{H}$ be every function from $\mathcal{X}$ to $\mathbb{R}$. One such function is just a list of values —

$$f_0(x_1) = 5, \quad f_0(x_2) = 2.3, \quad f_0(x_3) = -7, \quad \dots, \quad f_0(x_n) = 8$$

— so f_0 is the vector $\mathbf{f} = [f(x_1), \dots, f(x_n)]^\top \in \mathbb{R}^n$, and a probability distribution over $\mathcal{H}$ is nothing more exotic than **a probability distribution over that vector**.

Take the simplest one, $\mathbf{f} \sim \mathcal{N}(\mu, \sigma^2 I)$, and the density over functions is written out in full:

$$p(\mathbf{f}) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2} (f(x_i) - \mu_i)^2\right)$$

The only thing wrong with this is the *independence*: it says knowing $f(x_1)$ tells you nothing about $f(x_2)$, however close they are. Replace $\sigma^2 I$ with a covariance matrix that couples nearby inputs, let $n \rightarrow \infty$, and you have a Gaussian process. Everything after this slide is that one substitution.

Lecture 2's posterior, with the parameter replaced by a function

The handoff from Chapter 2 is not an analogy. It is the *same five steps*, with one symbol swapped.

STEP	LECTURE 2 — BELIEF OVER θ	LECTURE 4 — BELIEF OVER f
Model	a likelihood family for y	$y_i = f(x_i) + \epsilon_i$
Prior	$p(\theta)$ — Beta, Gamma, Normal	$p(f) = \mathcal{GP}(m(\cdot), k(\cdot, \cdot))$
Likelihood	$p(y \mid \theta)$	$p(\mathbf{y} \mid \mathbf{f}) = \mathcal{N}(\mathbf{f}, \sigma_\epsilon^2 I)$
Posterior	$p(\theta \mid y)$ — conjugate, closed form	$p(f \mid \mathcal{D})$ — Gaussian, closed form
Predict	$p(\hat{y} \mid y) = \int p(\hat{y} \mid \theta) p(\theta \mid y) d\theta$	$p(f^* \mid x^*, \mathcal{D}) = \int p(f^* \mid x^*, \mathbf{f}) p(\mathbf{f} \mid \mathcal{D}) d\mathbf{f}$

Chapter 2 ended on the Normal–Normal case and noted that the predictive variance splits into measurement noise you can never remove plus parameter uncertainty you can. **Replace the parameter by an entire function and that same predictive integral is Gaussian-process regression.** Gaussian conjugacy gives the closed-form posterior used here. Other likelihoods are possible but may require approximate inference.

Check — a prior over what

Lecture 2 put a prior on a *parameter*. What does Bayesian optimisation put a prior on?

- ☒ A The whole unknown function f — a distribution over functions, not over numbers
- ☐ B The location of the optimum x^*
- ☐ C The noise in each observation
- ☐ D The budget of evaluations remaining

This is the step up in object. A Gaussian process says: before seeing any data, here is my belief about **every function** that f could be. Each evaluation conditions that belief, and what comes back is not one fitted curve but a posterior over curves — a mean and, crucially, a variance that is small where you have looked and large where you have not.

— ACT 2

The Gaussian process

Q2. A GP prior and Gaussian observation model give a closed-form posterior. The mean, kernel and noise model each express a modelling assumption.

The Gaussian process — a prior over functions

Q1 Model an unknown f ?

Q2 What is a GP?

Q3 Where next?

Q4 What lies beyond?

A **Gaussian process** is a collection of random variables $\{f(x) : x \in \mathcal{X}\}$ such that **every finite subcollection is jointly Gaussian**.

Written out, for any $x_1, \dots, x_n$:

$$\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} m(x_1) \\ \vdots \\ m(x_n) \end{bmatrix}, \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix} \right), \quad f(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot))$$

THE MEAN FUNCTION

$m(x) = \mathbb{E}[f(x)]$ — the prior trend. We use $m \equiv 0$ after centring the outputs for this derivation. The choice matters, especially far from the observed inputs.

THE COVARIANCE FUNCTION

$k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))]$ — how strongly nearby points are tied together. It must make every $\mathbf{K}$ positive semidefinite. **This is where all the modelling lives.**

A function drawn from a GP prior is, in the source's own phrase, *an extremely high-dimensional vector drawn from an extremely high-dimensional multivariate Gaussian*. A finite set of function values has a Gaussian joint distribution. The kernel specifies **covariances**, not the missing-edge conditional independences of a Bayesian network.

Conditioning is the whole of it

Draw $\mathbf{f} = [f_1, \dots, f_{25}] \sim \mathcal{N}(0, \mathbf{K})$ and plot the 25 numbers in order: they look like a smooth curve, because $\mathbf{K}$ made neighbouring entries nearly identical. Read two entries of $\mathbf{K}$ off the diagonal band:

NEIGHBOURS — $\text{corr}(f_1, f_2) = 0.966$

The joint $p(f_1, f_2)$ is a thin cigar. Observe $f_1 = -0.313$ and the conditional $p(f_2 \mid f_1)$ is **a narrow spike** — you have almost measured f_2 without paying for it.

FOUR APART — $\text{corr}(f_1, f_5) = 0.573$

The joint is a fat ellipse. The same observation $f_1 = -0.313$ leaves $p(f_5 \mid f_1)$ **barely narrower than the prior** — that value you still have to buy.

And the mechanism is a single fact of linear algebra, the one the probability appendix calls Property 4:

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}\right) \implies Y_2 \mid Y_1 = y \sim \mathcal{N}(\mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(y - \mu_1), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12})$$

The kernel decides how much one observation is worth at every other point in the domain. Everything the GP does — the narrowing, the interpolation, the uncertainty that grows away from data — is that one formula applied at scale.

A GP update with just two numbers

the source's correlation example, calculated explicitly

Assume f_1, f_2 have zero mean, unit variance and correlation ρ . We observe $f_1 = -0.313$ without noise.

$$\mathbb{E}[f_2 \mid f_1] = \rho(-0.313), \quad \text{Var}(f_2 \mid f_1) = 1 - \rho^2.$$

RELATIONSHIP TO THE MEASURED POINT	CORRELATION	POSTERIOR MEAN	POSTERIOR VARIANCE
Nearby	0.966	−0.302	0.0668
Farther away	0.573	−0.179	0.6717

A stronger correlation transfers more information. The next matrix formula performs this same calculation using many measurements at once.

GP regression, two ways — the setup

p. 15 of the source — a distribution over functions, used as a Bayesian prior

Observe $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, collected as $\mathbf{X}$ and $\mathbf{y}$. Two ingredients, both Gaussian:

PRIOR — OVER THE FUNCTION

$$p(\mathbf{f}) = \mathcal{GP}(m(\cdot), k(\cdot, \cdot))$$

At the n training inputs: $\mathbf{f}_{1:n} \sim \mathcal{N}(\mathbf{m}, \mathbf{K})$, $K_{ij} = k(x_i, x_j)$. From here on $m(\cdot) = 0$ — subtract any known trend first.

LIKELIHOOD — HOW WE SEE IT

$$y_i = f_i + \epsilon_i, \quad \epsilon_i \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma_\epsilon^2)$$

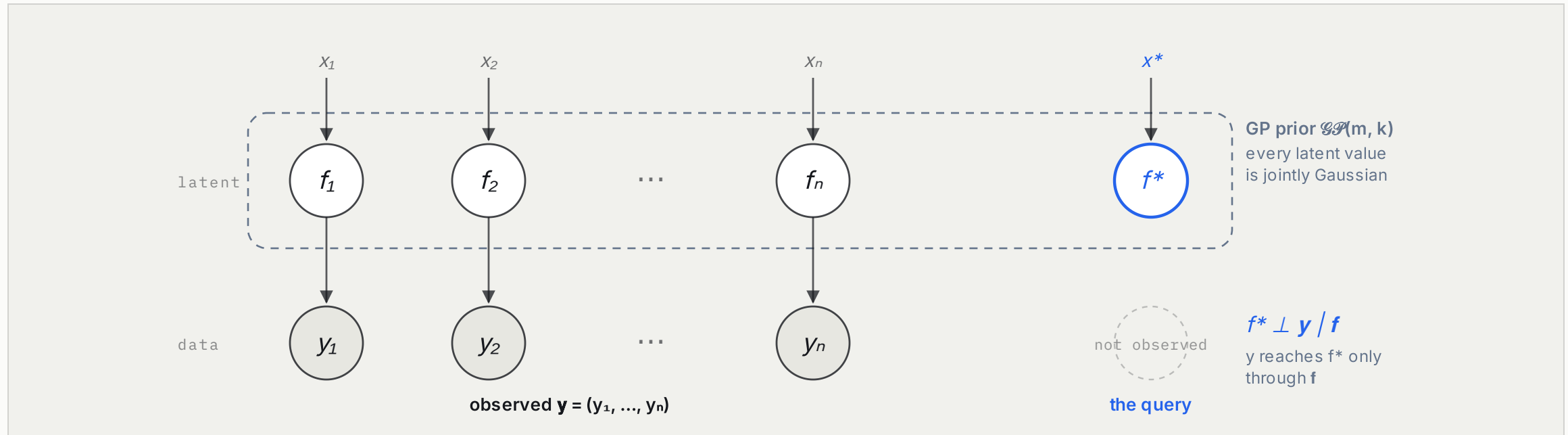
$$p(\mathbf{y} \mid \mathbf{f}) = \mathcal{N}(\mathbf{f}_{1:n}, \sigma_\epsilon^2 \mathbf{I})$$

We want $f^* = f(x^*)$ at a new input x^* . There are two ways to get there, and **they land on the same answer**:

- **A • the Bayesian view** — exactly Lecture 2's posterior predictive, with $\mathbf{f}$ in the place of θ .
- **B • the Gaussian shortcut** — write one joint Gaussian and condition it. This is how it is computed.

A1—the graphical model

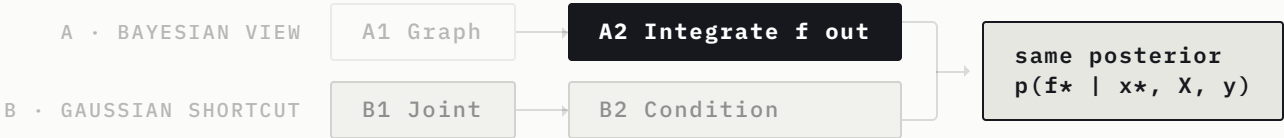
p. 14 of the Lecture 2 source, redrawn — θ becomes the latent function values



The latent values $f_1, \dots, f_n, f^*$ are coupled by the GP prior; each training value is seen once, through noise. The query f^* is never observed. Reading the arrows: $f^* \perp \mathbf{y} | \mathbf{f}$ — the data can reach the prediction only through the function values beneath it.

A2 — integrate the function values out

p. 15 of the source — Lecture 2's posterior predictive, verbatim



LECTURE 2 — A PARAMETER

$$p(\hat{y} \mid y) = \int_{\theta} p(\hat{y} \mid \theta) p(\theta \mid y) d\theta$$

LECTURE 4 — A FUNCTION

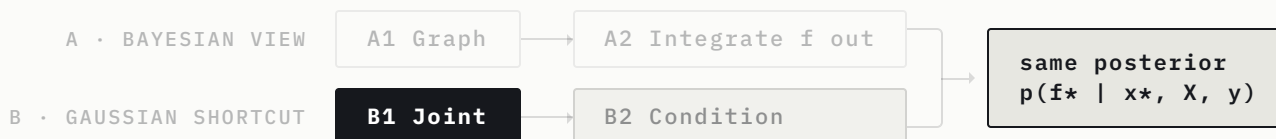
$$p(f^* \mid x^*, \mathbf{X}, \mathbf{y}) = \int_{\mathbf{f}} p(f^* \mid x^*, \mathbf{f}) p(\mathbf{f} \mid \mathbf{X}, \mathbf{y}) d\mathbf{f}$$

FACTOR	WHAT IT IS	WHY IT HAS THIS FORM
$p(\mathbf{f} \mid \mathbf{X}, \mathbf{y}) \propto p(\mathbf{y} \mid \mathbf{f}) p(\mathbf{f})$	the posterior over the function values	Bayes' rule — Gaussian likelihood times Gaussian prior
$p(f^* \mid x^*, \mathbf{f})$	the GP prior read at x^* , given $\mathbf{f}$	$f^* \perp \mathbf{y} \mid \mathbf{f}$ from A1 lets $\mathbf{y}$ drop out

Integrate $\mathbf{f}$ out and what remains depends **only on the data $(\mathbf{X}, \mathbf{y})$ and the query x^*** . Every factor is Gaussian, so the integral closes — route B finds its value without doing it.

B1—the joint of the data and the query

p. 37 of the source — one Gaussian over everything we see and everything we want



Stack the observations and the query value. They are jointly Gaussian — fill in the three blocks:

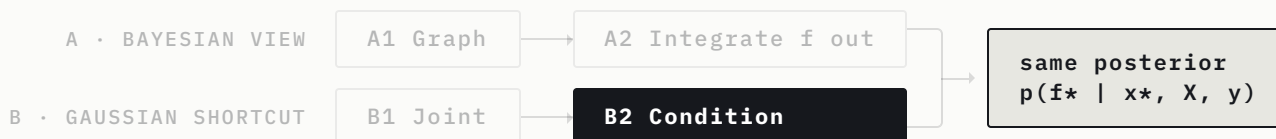
$$\begin{bmatrix} \mathbf{y}_{1:n} \\ f^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma_\epsilon^2 \mathbf{I} & \mathbf{k} \\ \mathbf{k}^\top & k(x^*, x^*) \end{bmatrix}\right)$$

BLOCK	WHERE IT COMES FROM
$\text{cov}(\mathbf{y}, \mathbf{y}) = \text{cov}(\mathbf{f}) + \text{cov}(\boldsymbol{\epsilon}) = \mathbf{K} + \sigma_\epsilon^2 \mathbf{I}$	$\mathbf{y} = \mathbf{f} + \boldsymbol{\epsilon}$, two independent Gaussians — their covariances add
$\text{cov}(\mathbf{y}, f^*) = \text{cov}(\mathbf{f} + \boldsymbol{\epsilon}, f^*) = \mathbf{k}, k_i = k(x_i, x^*)$	the noise is independent of f^* , so it drops out
$\text{var}(f^*) = k(x^*, x^*)$	the prior at the query

Noise sits on the $\mathbf{y}$ - $\mathbf{y}$ block only: we predict the latent $f(x^*)$, not a future noisy reading of it. No separate $p(\mathbf{y})$ is needed here — only this one block of the joint.

B2 — condition on the data

pp. 38–39 of the source — Property 4, applied block by block



PROPERTY 4 — GAUSSIAN CONDITIONAL

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}\right)$$

$$Y_2 \mid Y_1 = y \sim \mathcal{N}(\mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(y - \mu_1), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12})$$

MATCH THE BLOCKS FROM B1

Set $Y_1 = \mathbf{y}_{1:n}$, $Y_2 = f^*$, and both means to zero:

PROPERTY 4	GP
Σ_{11}	$\mathbf{K} + \sigma_\epsilon^2 \mathbf{I}$
$\Sigma_{21} = \Sigma_{12}^\top$	$\mathbf{k}^\top$
Σ_{22}	$k(x^*, x^*)$

Substitute, and the **posterior predictive distribution** falls out:

$$p(f^* \mid x^*, \mathcal{D}) = \mathcal{N}(\mu(x^* \mid \mathcal{D}), \sigma^2(x^* \mid \mathcal{D}))$$

$$\mu(x^* \mid \mathcal{D}) = \mathbf{k}^\top (\mathbf{K} + \sigma_\epsilon^2 \mathbf{I})^{-1} \mathbf{y}_{1:n}, \quad \sigma^2(x^* \mid \mathcal{D}) = k(x^*, x^*) - \mathbf{k}^\top (\mathbf{K} + \sigma_\epsilon^2 \mathbf{I})^{-1} \mathbf{k}$$

The posterior predictive — mean and uncertainty, in closed form

what both routes bought

$$\mu(x \mid \mathcal{D}) = \mathbf{k}^\top (\mathbf{K} + \sigma_\epsilon^2 \mathbf{I})^{-1} \mathbf{y}_{1:n}, \quad \sigma^2(x \mid \mathcal{D}) = k(x, x) - \mathbf{k}^\top (\mathbf{K} + \sigma_\epsilon^2 \mathbf{I})^{-1} \mathbf{k}$$

A • THE BAYESIAN VIEW

Posterior over $\mathbf{f}$, then integrate $\mathbf{f}$ out. Says **what the prediction is**: Lecture 2's posterior predictive over a function.

B • THE GAUSSIAN SHORTCUT

One joint, one conditioning. Says **how to compute it**: a single linear solve with $\mathbf{K} + \sigma_\epsilon^2 \mathbf{I}$.

The GP hands us, at every x , both a best guess *and* **how much to trust it** — and that pair is exactly what the next decision needs.

Read the two formulas. The mean is a **linear combination of the observed y 's**, with weights set by the kernel — they need not be positive or sum to one. The variance starts at the prior value $k(x, x)$ and is reduced by $\mathbf{k}^\top (\cdot)^{-1} \mathbf{k}$, large near data and vanishing far from it. For a fixed kernel and noise level, σ^2 **does not depend on y** : where the GP is uncertain is fixed the moment you choose *where* to look.

Check — route A, carried out, lands on route B

two Gaussians and one integral — the algebra closes

Write $\mathbf{S} = \sigma_\epsilon^2 \mathbf{I}$ and $k^{**} = k(x^*, x^*)$. Both factors of A2 are Gaussian:

$$\mathbf{f} \mid \mathbf{y} \sim \mathcal{N}(\mathbf{K}(\mathbf{K} + \mathbf{S})^{-1}\mathbf{y}, \mathbf{K} - \mathbf{K}(\mathbf{K} + \mathbf{S})^{-1}\mathbf{K}), \quad f^* \mid \mathbf{f} \sim \mathcal{N}(\mathbf{k}^\top \mathbf{K}^{-1}\mathbf{f}, k^{**} - \mathbf{k}^\top \mathbf{K}^{-1}\mathbf{k})$$

f^* is a linear map of $\mathbf{f}$ plus independent noise, so its mean passes through and its variance adds:

$$\mu = \mathbf{k}^\top \mathbf{K}^{-1} \mathbf{K}(\mathbf{K} + \mathbf{S})^{-1} \mathbf{y} = \mathbf{k}^\top (\mathbf{K} + \mathbf{S})^{-1} \mathbf{y}$$

$$\sigma^2 = k^{**} - \mathbf{k}^\top \mathbf{K}^{-1} \mathbf{k} + \mathbf{k}^\top \mathbf{K}^{-1} [\mathbf{K} - \mathbf{K}(\mathbf{K} + \mathbf{S})^{-1} \mathbf{K}] \mathbf{K}^{-1} \mathbf{k} = k^{**} - \mathbf{k}^\top (\mathbf{K} + \mathbf{S})^{-1} \mathbf{k}$$

The $\mathbf{k}^\top \mathbf{K}^{-1} \mathbf{k}$ terms cancel, leaving exactly route B. Checked numerically on six random points: the two routes agree to twelve decimal places. Route B is what you implement — it never forms $\mathbf{K}^{-1}$, only one solve with $\mathbf{K} + \mathbf{S}$.

Read the GP formula — what each object means

For n observations and one query input x , $\mathbf{K}$ is an $n \times n$ matrix with $K_{ij} = k(x_i, x_j)$; $\mathbf{k}$ is a length- n vector with $k_i = k(x_i, x)$.

ONE NOISY OBSERVATION

Use prior variances 1, covariance $k(x_1, x) = 0.8$, noise variance 0.25, and observe $y_1 = 1$.

$$\mu(x) = \frac{0.8}{1 + 0.25}(1) = 0.64.$$

$$\sigma_f^2(x) = 1 - \frac{0.8^2}{1.25} = 0.488.$$

A FUNCTION VALUE OR A NEW MEASUREMENT?

The formula predicts the latent $f(x)$.

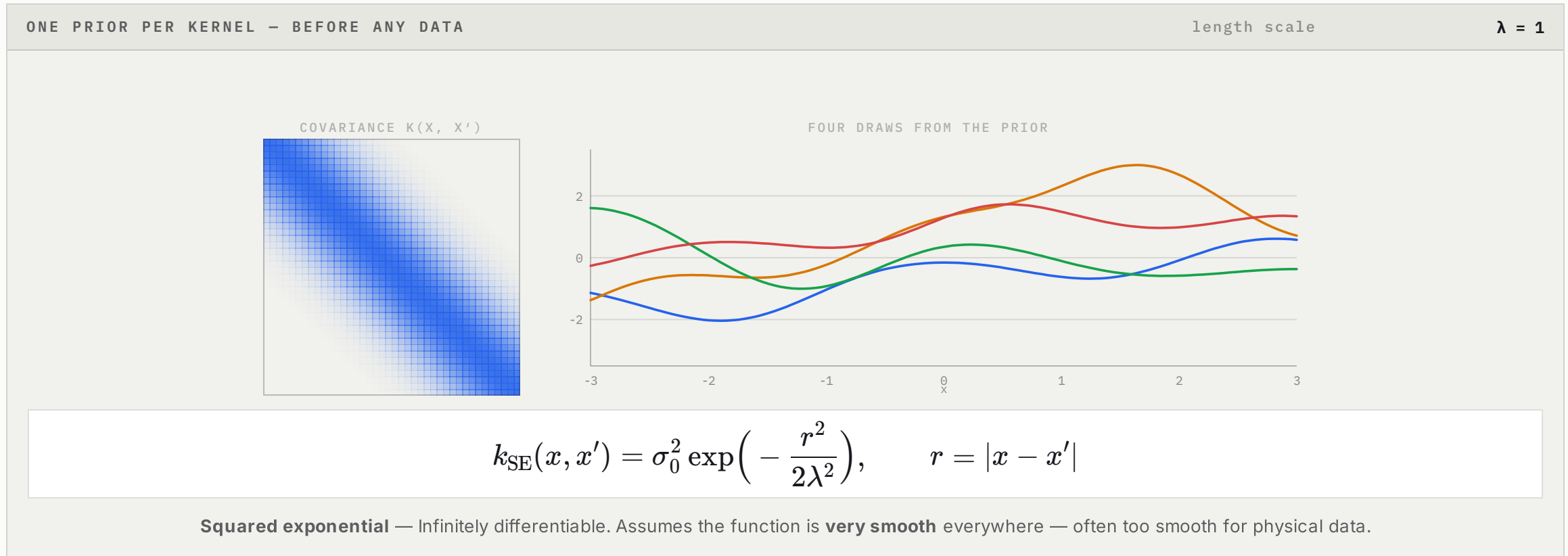
For a future independent noisy measurement $y = f(x) + \epsilon$, add its noise:

$$\sigma_y^2(x) = 0.488 + 0.25 = 0.738.$$

Uncertainty about the function and noise in a new measurement are different. Numerically, solve linear systems with $\mathbf{K} + \sigma_\epsilon^2 I$ rather than explicitly forming its inverse.

The kernel is the assumption

pp. 22–25 of the source — every kernel is a different belief about f , visible before any data



Pick a kernel and move the length scale. Left: the covariance $k(x, x')$ over the input range — a band along the diagonal says *nearby points agree*, stripes say *the pattern repeats*.

Right: four functions the prior considers typical, drawn from **the same random numbers for every kernel**, so what changes is the assumption alone. **ARD** is the SE kernel with one λ_d per input dimension; a large λ_d switches that dimension off.

Fitting the kernel — maximise the marginal likelihood

p. 28 of the source — the hyperparameters are fitted, not chosen

The kernel's knobs $\theta = (\sigma_\epsilon, \sigma_0, \boldsymbol{\lambda})$ are chosen to make the observed data most probable, with $\mathbf{f}$ integrated out. Write $\mathbf{C}_\theta = \mathbf{K}_\theta + \sigma_\epsilon^2 \mathbf{I}$, so that $\mathbf{y} \mid \theta \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_\theta)$:

$$\mathcal{L}(\theta) = \log p(\mathbf{y} \mid \theta) = \underbrace{-\frac{1}{2} \mathbf{y}^\top \mathbf{C}_\theta^{-1} \mathbf{y}}_{\mathcal{F}(\theta) : \text{data fit}} \underbrace{-\frac{1}{2} \log |\mathbf{C}_\theta|}_{\mathcal{S}(\theta) : \text{simplicity}} \underbrace{-\frac{n}{2} \log 2\pi}_{\text{constant}} \implies \theta^* = \arg \max_{\theta} [\mathcal{F}(\theta) + \mathcal{S}(\theta)]$$

DATA FIT $\mathcal{F}(\theta)$ — ALWAYS ≤ 0

How typical the observed $\mathbf{y}$ is under $\mathcal{N}(\mathbf{0}, \mathbf{C}_\theta)$. A flexible kernel — short λ — can bend to any $\mathbf{y}$ and keeps $\mathcal{F}$ near 0; a rigid one — long λ — cannot, and $\mathcal{F}$ falls.

SIMPLICITY $\mathcal{S}(\theta)$

$\log |\mathbf{C}_\theta|$ is the log-volume of datasets the prior spreads itself over. With the minus sign, a kernel prepared for *fewer* datasets — long λ — scores **higher**. Rasmussen & Williams call $+\frac{1}{2} \log |\mathbf{C}_\theta|$ the *complexity penalty*; $\mathcal{S}$ is its negative.

Both terms are **larger-is-better**, but **no θ maximises both**: as λ grows, $\mathcal{S}$ rises and $\mathcal{F}$ falls. So we maximise their **sum**, and θ^* is the best trade-off — not the maximiser of either term. The constant does not depend on θ and never moves θ^* .

Why the two terms must disagree

probability has to sum to one — and Occam's razor falls out of that

A kernel is a claim about which datasets are plausible, and that claim is a **distribution**: its total mass is fixed at 1. So being ready for more costs you on each.

A FLEXIBLE KERNEL — SHORT λ

Prepared for a huge variety of $\mathbf{y}$. It must spread one unit of probability thinly, so the particular $\mathbf{y}$ you saw receives **a small share**.

It can bend to anything, and is rewarded for nothing.

A RIGID KERNEL — LONG λ

Prepared for very few $\mathbf{y}$. If yours is among them it receives **a large share**; if it is not, almost none.

Simplicity pays — right up until it cannot reach the data.

Occam's razor is not imposed here; it falls out of normalisation. $\mathcal{F}$ asks *did it explain my data?*; $\mathcal{S}$ asks *how little else was it prepared to explain?*

In symbols: as λ grows, $\mathcal{S}(\theta)$ rises and $\mathcal{F}(\theta)$ falls. The widget two slides on draws exactly these two curves and their sum $\mathcal{L}(\theta)$.

Why the **marginal** likelihood — and not the likelihood

the question every student should ask of the previous slide

The likelihood $p(\mathbf{y} \mid \mathbf{f}, \sigma_\epsilon) = \mathcal{N}(\mathbf{f}, \sigma_\epsilon^2 \mathbf{I})$ is right there. Why integrate $\mathbf{f}$ away first?

1 • IT CANNOT SEE THE KERNEL

λ and σ_0 do not appear in $p(\mathbf{y} \mid \mathbf{f})$ at all — they live only in the prior $p(\mathbf{f} \mid \theta)$.

Maximising the likelihood cannot choose a kernel.

2 • IT HAS A TRIVIAL OPTIMUM

Maximise over $\mathbf{f}$ as well and the answer is $\mathbf{f} = \mathbf{y}, \sigma_\epsilon \rightarrow 0$: the function threads every noisy point and the likelihood goes to $+\infty$.
Perfect fit, zero information.

3 • THE MARGINAL ASKS THE RIGHT QUESTION

$p(\mathbf{y} \mid \theta) = \int p(\mathbf{y} \mid \mathbf{f}) p(\mathbf{f} \mid \theta) d\mathbf{f}$ averages over every function the prior finds plausible: *how probable was this data under the kernel, before we saw it?*

A kernel that could explain anything spreads its probability thin and scores low; a rigid one cannot reach the data. **The marginal likelihood is Occam's razor, built in** — no validation set required.

It is Lecture 2's **prior predictive** $p(y) = \int p(y \mid \theta) p(\theta) d\theta$ (p. 14 of that deck), with $\mathbf{f}$ in the place of θ — the same object that, in route A, the posterior was normalised by. The next slide shows $\mathcal{F}$ and $\mathcal{S}$ trading off as λ moves.

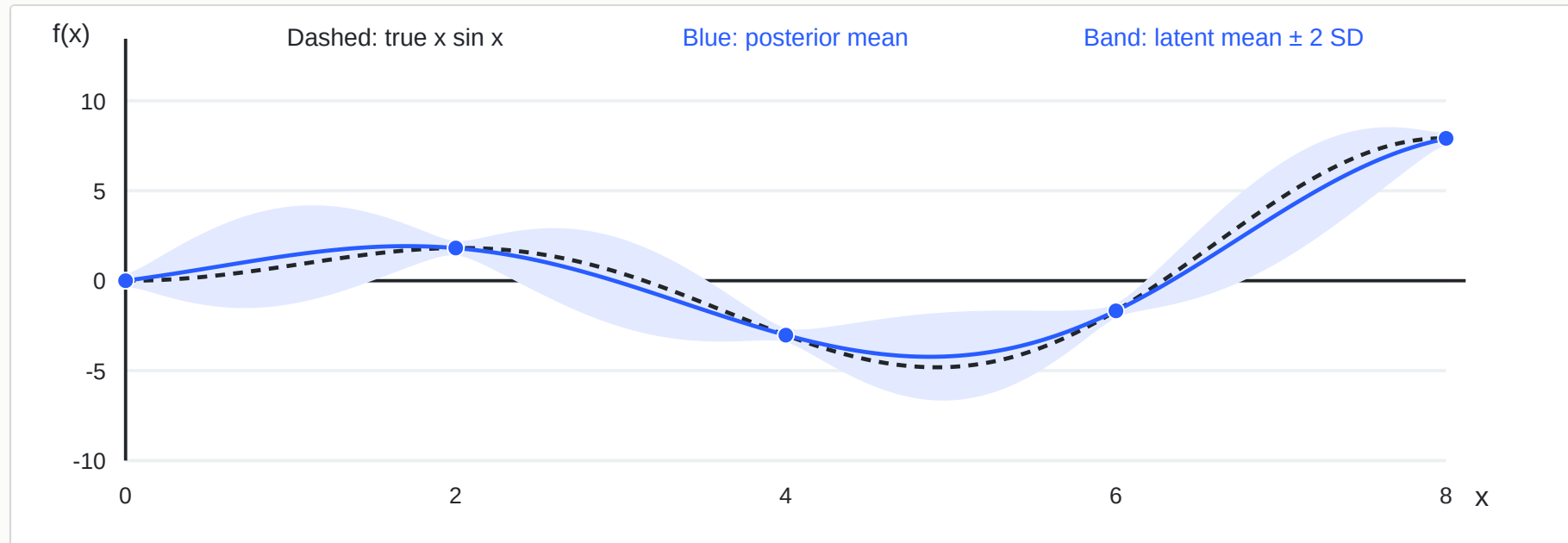
Turn the dial and watch the assumption move



The same seven observations, one kernel, one knob: $\theta = \lambda$, with $\sigma_0 = 1$ and $\sigma_\epsilon = 0.1$ held fixed. Short λ : the posterior spikes at each datum and falls back to the prior between them. Long λ : a near-straight line that cannot bend to the data. Right: $\mathcal{S}$ rises, $\mathcal{F}$ falls, and their sum peaks where neither is at its own maximum — the Occam balance, drawn.

The original regression exercise — fit $x \sin x$ before optimising

Rebuilt illustration with stated settings



True function $f(x) = x \sin x$; five observations at $x = 0, 2, 4, 6, 8$. This redrawing uses observed values equal to $f(x)$, a zero-mean SE prior, amplitude 5, length scale 1.5, and assumed observation-noise standard deviation 0.2.

The line is the posterior mean. The band is **latent-function mean $\pm$ 2 standard deviations** under this model. An interval for a new noisy reading would be wider.

Regression answers **what might the function be?** The next act adds the separate decision: **which input should we measure next?**

Check — what the length scale controls

You shorten a GP kernel's length scale ℓ . What happens to the posterior?

- ☐ A It becomes smoother, and the uncertainty between data points shrinks
- ☒ B It wiggles more, and the uncertainty grows back faster as you move away from a data point
- ☐ C Nothing changes between observations; ℓ only rescales the output
- ☐ D The mean is unaffected; only the observation noise changes

The length scale says how far a datum's influence reaches. Short ℓ means each observation informs only its immediate neighbourhood, so the posterior returns to the prior — high variance — almost immediately beyond it. Long ℓ borrows strength across the whole domain and gives a smooth, confident, and possibly badly wrong fit. It is the single knob that decides how much the model is willing to extrapolate.

— ACT 3

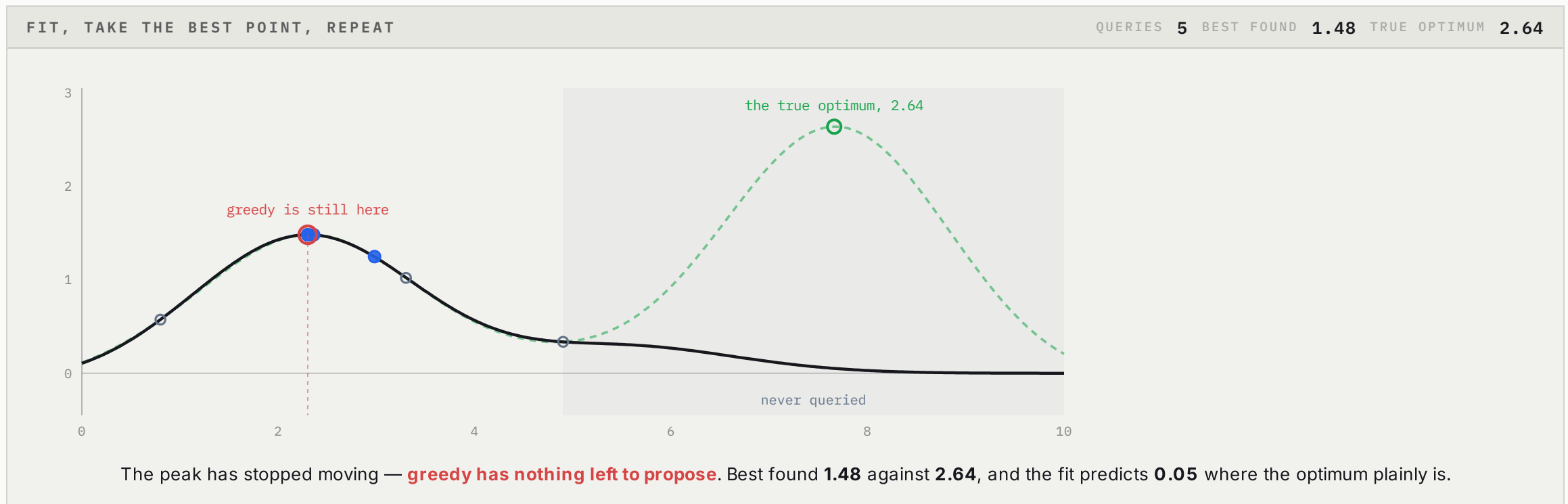
Where to look next

Q3. The posterior gives a best guess and a width at every x . One number has to fuse them, and that number is a rule for acting.

Why not just fit a model and take its best point?

the obvious thing to do — done honestly, and watched

An expensive f , three samples, and a regression through them. The regression has a highest point. Spend a query there, refit, and repeat. No uncertainty anywhere — just fit and exploit.



The green dashes are the **true** f — drawn for us, invisible to the fit, which has only the three grey samples. [Click the red peak](#) to spend a query on it, then the new peak, and again. The tall peak on the right stays in plain sight the whole time.

What went wrong — the model had no way to say "I have not looked here"

The greedy peak walks $2.98 \rightarrow 2.35 \rightarrow 2.30$ and then stops. Every later query lands on the same point, and nothing beyond $x = 4.9$ is ever tried.

WHAT IT FOUND

Best value 1.481 at $x = 2.30$, against a true optimum of 2.636 at $x = 7.67$ — **44% short**, and the optimum sits 2.8 away from the furthest point ever queried.

WHY IT NEVER WENT THERE

At $x = 7.67$ the truth is 2.636 and the fit predicts -0.011 . We could see the peak; the fit could not. It does not report *ignorance* about the right half — it reports a **low value**, and a greedy rule has no reason to go and check.

A point estimate cannot distinguish "**I looked, and it is bad**" from "**I never looked.**" Both come back as a small number. Exploration has to be paid for by something the fit alone does not contain.

This is not a failure of the optimiser — the argmax was computed exactly every time. It is a failure of what was handed to the optimiser. The fix is not a better search; it is a score that knows where the model is uncertain, which is the next slide.

The acquisition function — a score you are allowed to optimise

Q1 Model an unknown f ?

Q2 What is a GP?

Q3 Where next?

Q4 What lies beyond?

Given the posterior (μ, σ) , the **acquisition function** $A(x)$ scores how worthwhile it would be to query x next. Then

$$x_{\text{next}} = \arg \max_x A(x)$$

which is a *cheap* inner optimisation over the surrogate — no expensive evaluations, gradients available, run it as long as you like.

TWO IMPULSES

High $\mu(x)$ says *probably good* — spend the query where the model already expects a win. That is **exploitation**.

High $\sigma(x)$ says *worth learning* — spend the query where the model is ignorant, because that is where a surprise can hide. That is **exploration**.

ONE NUMBER

$A(x)$ is whatever function of μ and σ you are willing to defend. The three standard answers differ only in **how they weigh the two impulses**, and the disagreement is real: on the same posterior they point at different places.

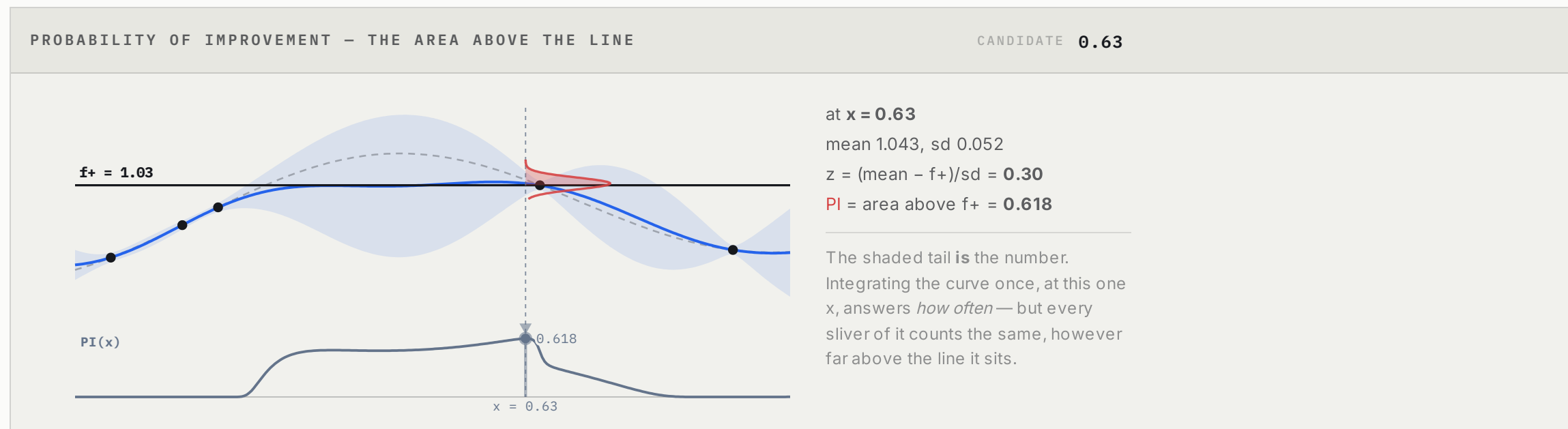
This is the explore–exploit dilemma — the same one inside every reinforcement learner — but here it is not a heuristic. It is an explicit, differentiable, optimisable score, which is why Bayesian optimisation is the cleanest place in the course to meet it.

Where the number comes from — one area at one point

pp. 113–116 of the source, redrawn — probability of improvement

At one input the GP is only a Gaussian, $f(x) \mid \mathcal{D} \sim \mathcal{N}(\mu(x), \sigma^2(x))$. Stand it on its side against the incumbent line and ask what fraction clears:

$$\text{PI}(x) = \Pr(f(x) > f^+) = \int_{f^+}^{\infty} p(f \mid \mathcal{D}) df = \Phi\left(\frac{\mu(x) - f^+}{\sigma(x)}\right)$$



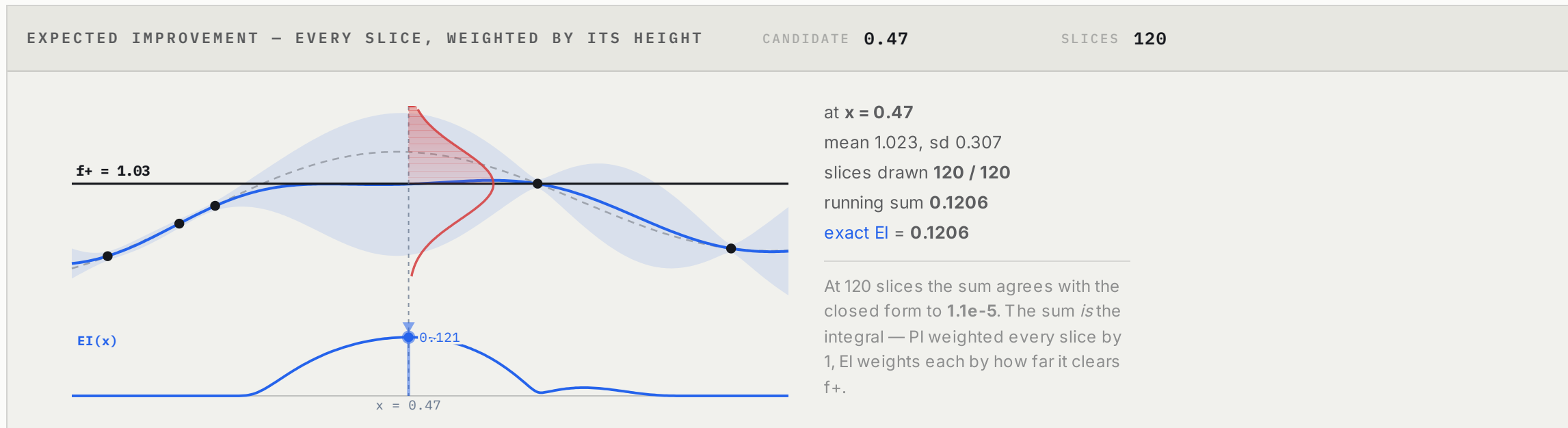
The curve on the dashed column *is* the posterior at that x , sideways; shade it and the number appears. **That one area is one point of the PI curve below** — move the candidate and watch the dot travel along it. And there is the complaint against PI: a sliver a thousandth above the line counts the same as a gain of half a unit, because the integrand is $p(f)$, which has no idea how high f is.

Expected improvement — the same tail, weighted by how far it clears

pp. 118–123 of the source — five printed pages of one accumulating sum

Keep the region; change what a sliver is worth — not 1, but the improvement it delivers.

$$\text{EI}(x) = \mathbb{E}[\max(0, f - f^+)] = \int_{f^+}^{\infty} (f - f^+) p(f | \mathcal{D}) df = \sigma(x) [z \Phi(z) + \phi(z)], \quad z = \frac{\mu(x) - f^+}{\sigma(x)}$$



Each slice contributes **the green length times the red one** — how far above f^+ it sits, times how probable that is. Watch the dot below climb onto $\text{EI}(x)$ as the sum completes: **the integral you are building is that point**. PI weighted every slice by 1 and answered *how often*; EI weights each by $(f - f^+)$, which is why it needs no margin ξ to stop it hugging the incumbent.

Three scores

	RULE	READS AS	LEANS
Probability of improvement	$\text{PI}(x) = \Phi\left(\frac{\mu(x) - f^+ - \xi}{\sigma(x)}\right)$	<i>how likely</i> is any improvement at all	exploit
Expected improvement	$\text{EI}(x) = \mathbb{E}[\max(0, f(x) - f^+)]$	how likely and by how much	balanced
Upper confidence bound	$\text{UCB}(x) = \mu(x) + \kappa \sigma(x)$	optimism, at an explicit price κ	tunable

Here Φ is the standard normal CDF, $\xi \geq 0$ is an improvement margin, and $\kappa \geq 0$ weights uncertainty. In the noiseless case, f^+ is the best observed value. With noise, the incumbent requires a noise-aware definition.

WHY PI PLAYS IT SAFE — AND WHY EI EXISTS

PI asks only *will x beat f^+ ?* — never *by how much*. A point just beside the incumbent, with μ a hair above f^+ and almost no uncertainty, wins almost surely: $\text{PI} \approx 1$, for a gain of almost nothing. An unexplored point that could win big — but whose mean is no better than f^+ — scores at most $\frac{1}{2}$. So PI keeps taking the safe sliver: **exploitation, not exploration**. It ranks bets by the chance of winning and ignores the payout.

Patch: count only wins larger than a margin ξ — a knob to tune: too small and the search stays local, too large and it wanders. **Fix:** EI weights each win by its size ($f - f^+$), so a large, uncertain gain can outscore a certain sliver, with no knob at all.

Choose between two candidate experiments

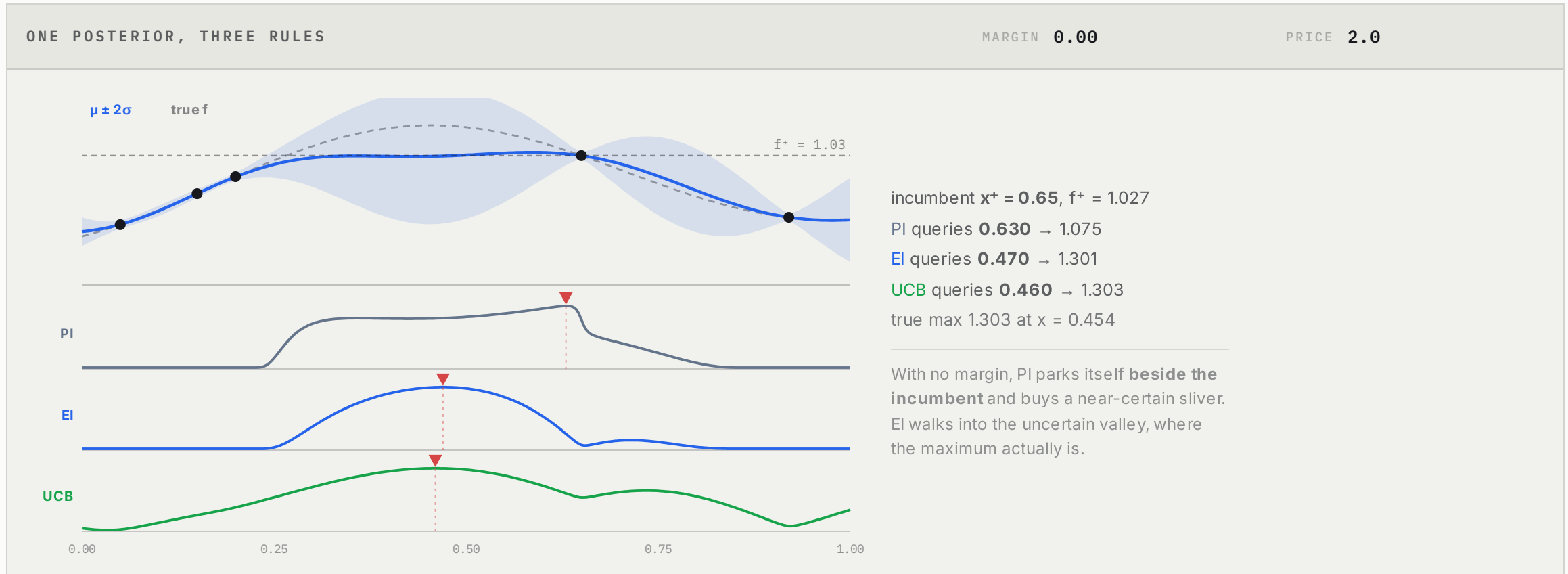
maximisation, incumbent 1, no improvement margin

Suppose the posterior at candidate A is $\mathcal{N}(1.10, 0.05^2)$ and at B is $\mathcal{N}(1.00, 0.50^2)$. For UCB, use $\kappa = 2$.

SCORE	CANDIDATE A	CANDIDATE B	CHOICE
Mean only	1.10	1.00	A
PI	$\Phi(2) \approx 0.977$	$\Phi(0) = 0.500$	A
EI	$0.10\Phi(2) + 0.05\phi(2) \approx 0.1004$	$0.50\phi(0) \approx 0.1995$	B
UCB	$1.10 + 2(0.05) = 1.20$	$1.00 + 2(0.50) = 2.00$	B

A offers a likely small gain. B offers a less certain but potentially larger gain. **The acquisition rule determines which opportunity is worth the next measurement.**

The three rules, disagreeing

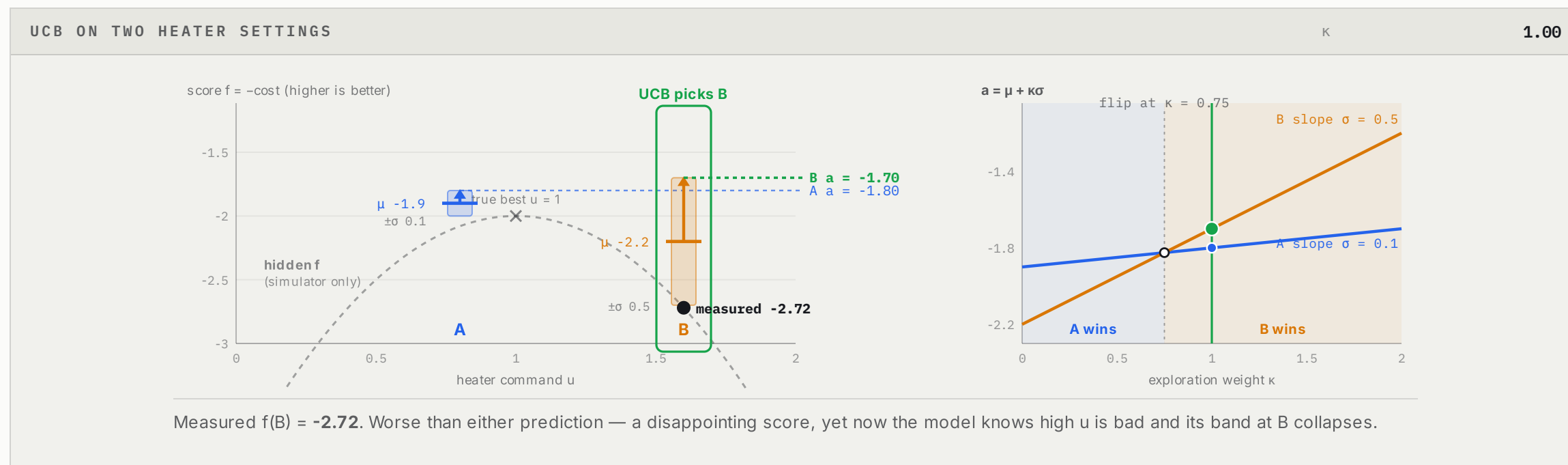


One posterior, five observations, three scores drawn underneath it, each with its own $\arg \max$ marked. At $\xi = 0$, PI points at $x = 0.630$ — hard against the incumbent at 0.65, buying a near-certain sliver. EI points at $x = 0.470$, into the wide-uncertainty valley where the true maximum actually is. Turn ξ up and PI walks out to meet EI; turn κ down and UCB collapses onto the greedy mean. **The knob is the same knob in all three.**

Temperature thread — choose an informative heater experiment

shared teaching example · predict before revealing the calculation

Room at 20 °C, target 22 °C, so the error is $x = -2$. A heater command $u \in [0, 2]$ gives $x' = x + u$ at cost $c = (x')^2 + u^2$. BO **never sees this formula** — it can only run an experiment and read the score $f = -c$. Its GP currently believes:



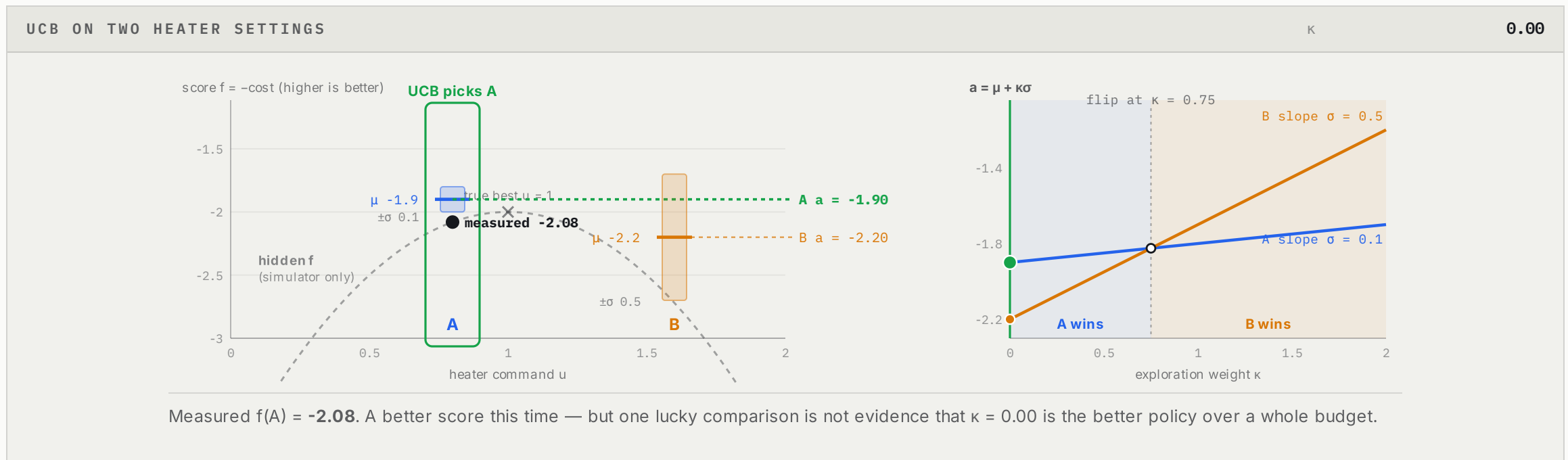
Predict: at $\kappa = 1$, does UCB $a(u) = \mu(u) + \kappa\sigma(u)$ pick **A** (better mean, narrow band) or **B** (worse mean, wide band)? Press → to add each candidate's optimism bonus $\kappa\sigma$, and → again to run the experiment it picks.

Acquisition value is not predicted reward or measured reward. A disappointing experiment can still provide useful information.

Try it — the exploration weight becomes zero

work independently · reveal only after writing an answer

Same two beliefs, but now $\kappa = 0$. **Which candidate is selected?** And if its measured score turns out better, **does that show $\kappa = 0$ is the better policy?** Write both answers, then press →.



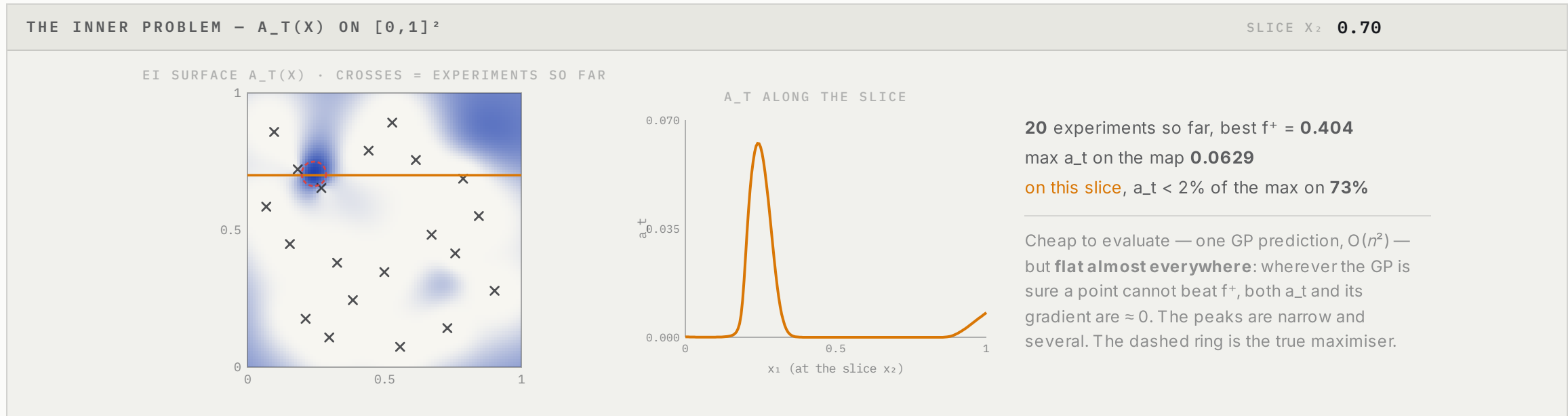
After the reveal, drag κ : each line's slope is its σ , so **the more uncertain candidate gains faster as κ grows**, and the pick flips where the lines cross.

Distinguish a calculation about one acquisition choice from evidence about a whole optimization strategy.

Optimise the acquisition — an optimisation inside the optimisation

source pp. 139–140 · a cheap problem with an awkward shape

Every round of BO ends in an inner problem, $x_{t+1} = \arg \max_{x \in \mathcal{X}} a_t(x)$. One evaluation of a_t is one GP prediction — $O(n^2)$ arithmetic, microseconds — against hours for one experiment. So we can afford **thousands of evaluations of a_t** . What makes the problem awkward is its shape.



A real EI surface: a GP on twenty observations (crosses) in $[0, 1]^2$, colour $\propto \sqrt{a_t}$. Drag the slice to see the profile. EI is ≈ 0 wherever the GP is confident a point cannot beat f^+ — **flat almost everywhere, with a few narrow peaks**.

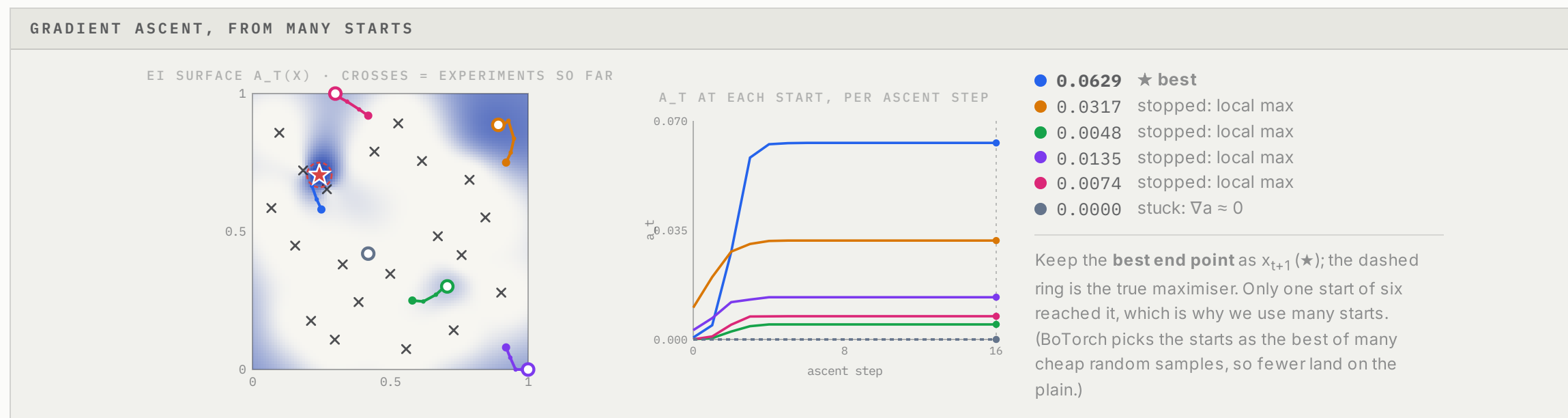
The source deck names three families of solver (p. 140): **gradient methods** (conjugate gradient, BFGS), **Lipschitz-based heuristics** (DIRECT) and **evolutionary algorithms** (CMA-ES). The next three slides run each one on this surface.

Method 1— gradient ascent, from many starts

source p. 141 · Wilson, Hutter & Deisenroth (2018); BoTorch, Balandat et al. (2020)

EI is differentiable, and with $z = (\mu - f^+)/\sigma$ its gradient is closed-form:

$$\frac{\partial \text{EI}}{\partial \mu} = \Phi(z), \quad \frac{\partial \text{EI}}{\partial \sigma} = \phi(z) \quad \implies \quad \nabla_x a_t(x) = \Phi(z) \nabla_x \mu(x) + \phi(z) \nabla_x \sigma(x), \quad x \leftarrow x + \eta \nabla_x a_t(x)$$



Each $\rightarrow$ moves all six starts two ascent steps uphill; the last $\rightarrow$ keeps the best end point as x_{t+1} (★). The dashed ring marks the true maximiser.

Method 2 — DIRECT, dividing rectangles

source pp. 142–146 · Jones, Perttunen & Stuckman (1993); Finkel's user guide (2003)

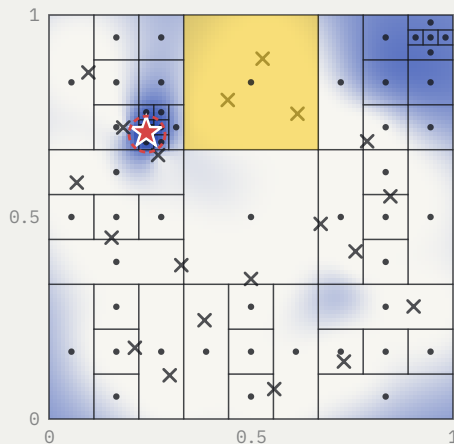
Sample the centre c_j of each box; d_j is its centre-to-corner size. If a_t were K -Lipschitz, box j could hold at most $a_t(c_j) + Kd_j$.

DIRECT divides every box that is best for *some* $K \geq 0$:

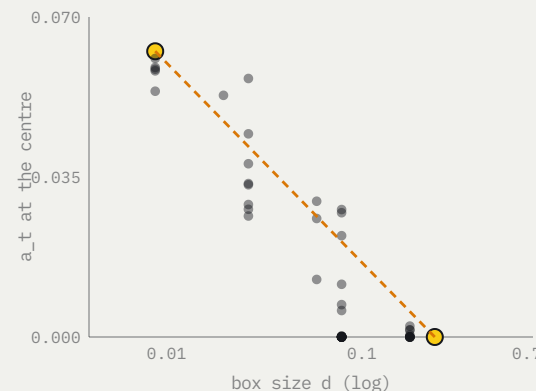
$$\exists K \geq 0 : \quad a_t(c_j) + K d_j \geq a_t(c_i) + K d_i \quad \text{for all boxes } i$$

DIRECT — DIVIDE THE RECTANGLES

EI SURFACE $A_T(X)$ · CROSSES = EXPERIMENTS SO FAR



EVERY BOX: SIZE VS VALUE AT ITS CENTRE



iteration 10 · 57 boxes

57 evaluations of a_t (one per centre)

next to divide 2

best centre **0.0625** = 99% of max

Boxes are finest where a_t is high, and every region was still sampled at least coarsely. No gradient was used — only values — but the count grows fast with dimension.

Each $\rightarrow$ is one iteration: the yellow boxes are the potentially optimal ones, and they are trisected along their longest side. Right: every box as a point (size, value); the selected ones form the upper-right hull.

Method 3 — CMA-ES, moving a Gaussian

source pp. 147–149 · Hansen & Ostermeier (2001); Hansen, [The CMA Evolution Strategy: A Tutorial](#) (2016)

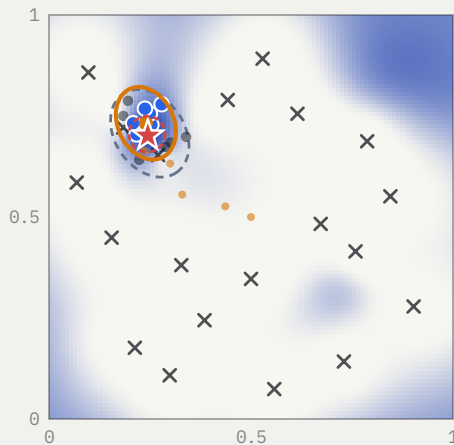
Search with a distribution over inputs instead of a single point — **sample** λ points, **select** the μ best, **re-fit** the Gaussian:

$$\mathbf{x}_k^{(g+1)} \sim \mathbf{m}^{(g)} + \sigma^{(g)} \mathcal{N}(\mathbf{0}, \mathbf{C}^{(g)}), \quad \mathbf{m}^{(g+1)} = \mathbf{m}^{(g)} + c_m \sum_{i=1}^{\mu} w_i (\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)}),$$

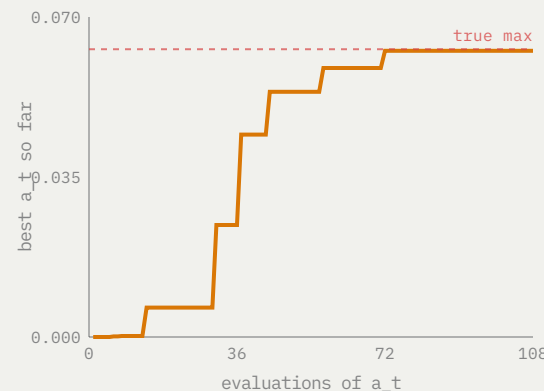
$$\mathbf{C}_{\mu}^{(g+1)} = \sum_{i=1}^{\mu} w_i (\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)}) (\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)})^{\top}$$

CMA-ES — MOVE A GAUSSIAN

EI SURFACE $A_T(x)$ · CROSSES = EXPERIMENTS SO FAR



BEST A_T FOUND VS EVALUATIONS



generation **9** · $\lambda = 12$ samples, $\mu = 6$ kept

108 evaluations of a_t

best so far **0.0626** = 99% of max

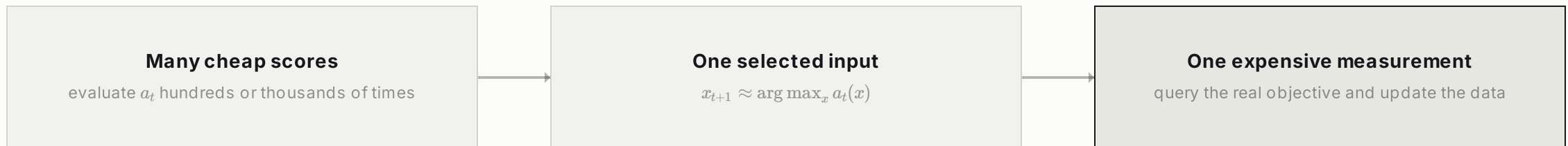
Converged: a small Gaussian on one peak; the best sample is x_{t+1} (★). One Gaussian can also settle on a *lesser* peak — other seeds do here — so practical use adds **restarts**. It searched with a **distribution over inputs**, the idea Lecture 6 turns into a generative model over designs.

Each $\rightarrow$ is one generation: grey ellipse = the current Gaussian (2σ), dots = its $\lambda = 12$ samples, blue = the $\mu = 6$ kept, amber = the re-fitted Gaussian. $x_{i:\lambda}$ is the i -th best sample; w_i are decreasing weights.

Three optimisers, one inner problem

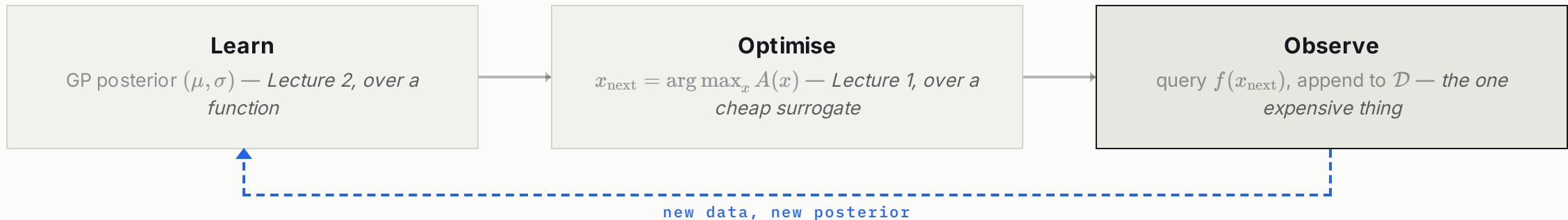
what each spends, and when to reach for it

	GRADIENT, MULTI-START	DIRECT	CMA-ES
uses	values and gradients of a_t	values only	values only (ranks)
global?	only through the starts	yes, by design (bounded box)	partly — one Gaussian, restarts help
cost grows with dimension	mildly	fast	moderately
typical use	the default in BoTorch-style libraries	low-dimensional boxes	non-smooth or awkward a_t



DIRECT and CMA-ES could optimise f itself when f can be evaluated (source p. 140) — but they need hundreds of evaluations, exactly what an expensive f cannot afford. **BO spends those evaluations on a_t instead**, and a locally optimised a_t still costs only a slightly worse query, never an extra experiment.

The loop

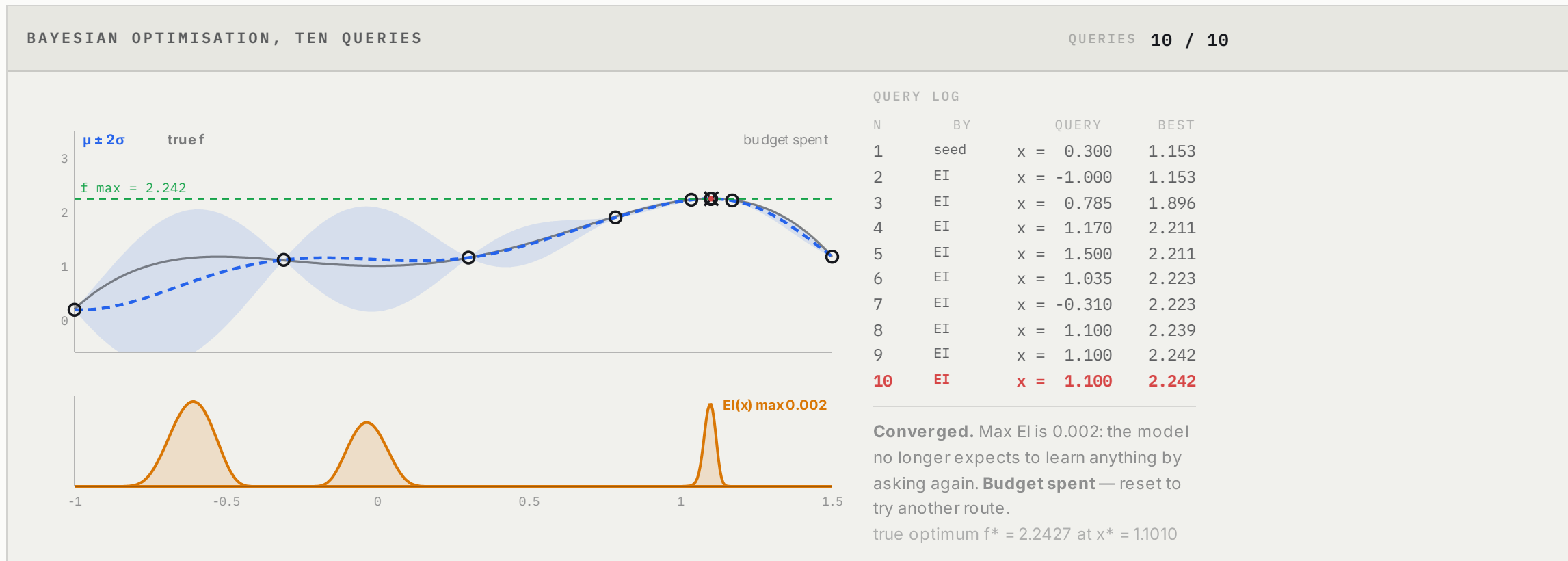


The dashed return arrow is what makes this a lecture rather than a technique. Each pass changes the belief, the changed belief changes the acquisition surface, and the changed surface changes where you look. **Nothing in Part I had a return arrow.**

Note also what the loop does *not* do: it never touches f except at step 3. All the optimisation happens on the surrogate. The whole design is an accounting trick for spending a scarce resource — expensive evaluations — by substituting a cheap one.

Ten queries on a quartic

Example 4.1 · maximise $-1.3x^4 + x^3 + 1.5x^2 + 1$ over $-1 \leq x \leq 1.5$ with noise $\sigma_\epsilon = 0.01$



Press *next query* and watch EI decide — or click either chart to spend a query where *you* think best, and compare it with EI's choice. The second query goes straight to the far boundary $x = -1$ — the mean there is unremarkable, but the uncertainty is enormous, and EI pays to find out. By the eighth the queries have collapsed onto $x = 1.10$, and the EI peak has fallen from 0.48 to 0.002: **the model expects little additional improvement under this acquisition rule.** A small EI is not a proof that the true global optimum has been found. The true maximum is $x^* = 1.1010$, $f^* = 2.2427$.

The lecture's own run took eleven queries and stopped at $x = 1.11$, $y = 2.24$ — the same answer, to the precision the noise allows.

Two representative BO applications — the same loop, different experiments

EGO for expensive engineering functions · practical BO for ML hyperparameters

	JONES, SCHONLAU & WELCH: EGO (1998)	SNOEK, LAROCHELLE & ADAMS (2012)
Decision x	an engineering design	a model's hyperparameters
One expensive observation	a simulation or physical evaluation	train the model and measure validation performance
Shared idea	GP surrogate and expected improvement	GP-based acquisition and repeated experiments
Additional emphasis	spend a small evaluation budget well	kernel/hyperparameter treatment, evaluation duration and parallelism

Teaching experiment: give grid search, random search and BO the same initial observations and evaluation budget. Plot the best observed value after each query; if evaluation costs differ, also compare elapsed cost. Keep a separate test set for the final ML evaluation.

A smooth posterior plot is a regression result. **A better recommendation under the same experimental budget** is an optimization result.

[EGO](#) · [Practical Bayesian Optimization](#)

This loop is a policy — the first in the course

THE BANDIT, LECTURE 4

$$\pi : [(a_1, r_1), \dots, (a_{t-1}, r_{t-1})] \rightarrow a_t$$

Find π^* maximising $\mathbb{E}[\sum_t r_t]$.

BAYESIAN OPTIMISATION, LECTURE 4

$$\pi : [(x^1, y^1), \dots, (x^{n-1}, y^{n-1})] \rightarrow x^n$$

A common BO objective is a good **final recommendation** $\hat{x}_T$:
maximise $\mathbb{E}[f(\hat{x}_T)]$ after T queries.

Both are rules mapping **history to the next action**. Their goals can differ: cumulative reward values every trial; final-design optimisation values the recommendation after the trials.

An acquisition rule and an RL exploration rule both turn current information into the next action. Their stored objects differ: a GP posterior describes uncertainty about a function; a standard Q -table estimates expected returns and is not itself a posterior distribution. In Lecture 8, an exploration rule acts on those estimates.

Discuss — do the trials count, or only the answer?

Same policy form, two objectives. Decide before the next slide.

SIX PROBLEMS

1. Tuning a network's hyperparameters on a GPU cluster
2. Choosing which headline each visitor to a news site sees
3. Assigning doses to patients in a clinical trial
4. Screening alloy compositions, one furnace run each
5. Setting yaw angles on a wind farm that is selling its power
6. Tuning a walking robot's gait on the real hardware

THREE QUESTIONS

1. **Which objective?** $\mathbb{E}[\sum_t r_t]$ or $\mathbb{E}[f(\hat{x}_T)]$ — and what in the problem decides it?
2. **Should exploration change?** More or less of it — and early or late in the budget?
3. **What do you hand over at the end?** The last query, the best y observed, or $\arg \max_x \mu(x)$?

Answers — the objective sets how much to explore

simple regret $f^* - f(\hat{x}_T)$ versus cumulative regret $\sum_t (f^* - f(x^t))$

	ONLY THE ANSWER COUNTS	EVERY TRIAL COUNTS
Problems	1 hyperparameters · 4 alloys	2 headlines · 3 doses · 5 live wind farm
A poor query costs	one unit of budget — only $\hat{x}_T$ is deployed	budget and its reward — each trial is a payoff, or a patient
Exploration	cheap; still worth it at the last query	repaid only over the remaining horizon — explore early, exploit late
Suited rules	El, knowledge gradient, entropy search	UCB (GP-UCB), Thompson sampling
Hand over	$\arg \max_x \mu(x)$, not the luckiest noisy y	nothing separate — the queries were the product

Problem 6 is a hybrid: only the final gait matters, but a query that topples the robot is not free. Safe BO (e.g. SafeOpt) keeps the final-answer objective and restricts every query to where the posterior is confidently safe.

The objective, not the model, sets the exploration schedule — and no rule is best at both. [Bubeck, Munos & Stoltz \(2009\)](#): the smaller a strategy's cumulative regret, the larger its lower bound on simple regret.

Check — the acquisition function's job

An acquisition function turns a GP posterior into the next query. Why not simply query the point with the highest posterior *mean*?

- ☐ A Because the mean is biased upward wherever data is scarce
- ☐ B Because the mean has no maximum when the domain is continuous
- ☒ C Because that is pure exploitation — it never visits the regions the model admits it knows nothing about, where the true optimum may sit
- ☐ D Because the posterior mean is not differentiable, so it cannot be maximised

Maximising the mean trusts the model exactly where the model is least entitled to be trusted. Every acquisition function is a rule for trading the mean against the **variance**, and that trade is the seed of the explore–exploit problem that returns as ϵ -greedy in Lecture 8. Lecture 5 shows what happens to a design pipeline that forgets this and optimises the surrogate alone.

— ACT 4

Beyond, toward RL

Q4. Strip the function away and the loop is a bandit. Add a context and it is a contextual bandit. Add dynamics and it is reinforcement learning.

Bayesian optimisation is a bandit with infinitely many arms



Put a slot machine under every point of the domain. Pulling arm x pays $f(x)$ plus noise; the payouts are unknown; you have a budget of pulls. That is the n -armed bandit with $n = \infty$ — and the reason it is not hopeless is the **kernel**, which makes pulling one arm tell you about its neighbours.

WHAT THE BANDIT CONTRIBUTES

The word *policy*, and the trade-off in its bare form: **acquiring new information** against **capitalising on the information already held**. A Bayesian finite-bandit model can carry a posterior for each arm; the GP additionally couples rewards across input locations.

WHAT BO CONTRIBUTES

Structure. A kernel lets an observation inform other inputs. This can reduce the number of required evaluations when the smoothness assumptions fit the problem; there is no universal small-query guarantee.

The lineage runs both ways. The bandit's Bayesian form is Chapter 2 exactly: after w wins and l losses, arm i has posterior $\text{Beta}(1 + w_i, 1 + l_i)$ and mean $\rho_i = \frac{w_i+1}{w_i+l_i+2}$ — the pseudo-count update from Lecture 2, now serving as **a belief state that an action will change**.

The finite-bandit calculation underneath the loop

Evaluative feedback and incremental learning

A pull reveals the reward of **the chosen arm**, not the reward every alternative would have given. Suppose arm A has three observed rewards 1, 0, 1, so $Q_3(A) = 2/3$.

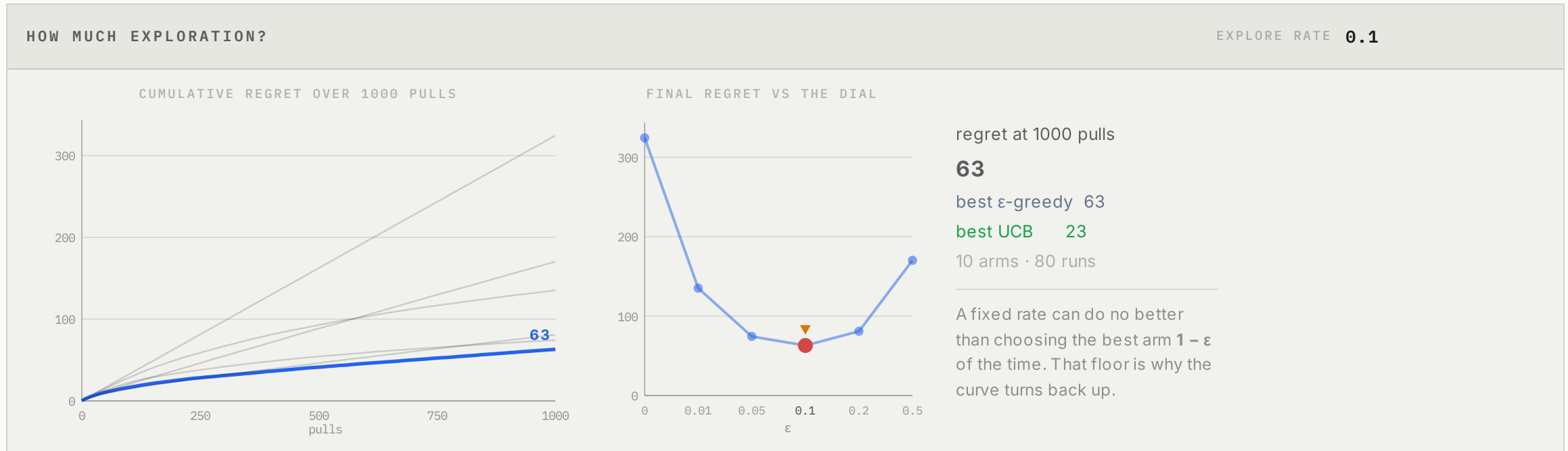
After one more reward of 0,

$$Q_4(A) = Q_3(A) + \frac{1}{4}(0 - Q_3(A)) = \frac{1}{2}.$$

CHOICE RULE	WHAT IT USES	REMAINING ISSUE
Greedy / optimistic initial values	largest estimated reward / initially hopeful estimates	early luck can mislead; optimism is not a permanent exploration guarantee
ϵ -greedy / softmax	uniform exploration / reward-based randomisation	exploration depends on the rate or temperature
UCB	mean plus an uncertainty bonus	confidence assumptions and reward scale matter

A constant step size tracks changing rewards by forgetting old observations; $1/k$ computes a sample average in a stationary problem. Preference and pursuit updates are in the appendix.

How much exploration is the right amount?



Ten arms, unknown payout probabilities, a thousand pulls, cumulative regret on the vertical axis. Pure greed ($\epsilon = 0$) locks onto whichever arm happened to pay first and never recovers. Constant thrashing ($\epsilon = 0.5$) pays a fixed toll on every round. [Compare the fixed exploration rates with UCB in this simulated run](#) — because it explores where the uncertainty actually is, rather than at random. That is the whole argument for $\mu + \kappa\sigma$, made without a Gaussian process anywhere in sight.

From a function to a context

Often the right action depends on a **context** c revealed just before each decision — the best price given the season, the best treatment given the patient, the best yaw angles given the wind direction. The object we want is no longer a point but a map:

$$x^* = \pi^*(c) = \arg \max_x f(x; c)$$

HOW THE GP ABSORBS A CONTEXT

Put a kernel on the context too and multiply:

$$k((x, c), (x', c')) = k_X(x, x') \cdot k_C(c, c')$$

which couples function values at similar inputs and contexts.

Similar values do not guarantee similar argmax locations, especially when two peaks nearly tie.

WHAT WE HAVE JUST BUILT

A rule that takes past (c, x, y) data, maintains a belief over the function, and chooses the next action for the observed context. The objective may be cumulative reward or the quality of a final recommendation.

With a context-arrival model and a horizon, the decisions can be formulated over a belief state. A myopic acquisition rule is not generally the optimal policy of that planning problem.

And it costs something up front. In the professor's own wind-farm study, the contextual learner starts at 0.79 average power efficiency against a greedy controller's 0.93, crosses it at roughly 2 500 iterations, and finishes ahead at 0.94. [This reported run illustrates a short-term exploration cost and a later benefit](#). Other problems and exploration rules need not show the same pattern.

Unknown constraints — feasibility is also an experiment

source pp. 176–178 · Gardner et al. (2014)

Lecture 1 wrote constraints as known functions: you could check $c(x)$ before spending anything. Here the constraint is **as expensive to evaluate as the objective**, and you learn whether x was feasible only after running it.

$$\max_x f(x) \quad \text{subject to} \quad c(x) \leq \lambda$$

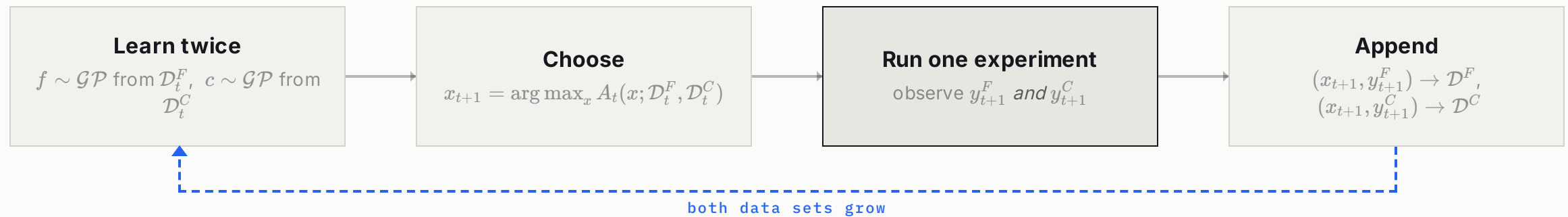
SETTING IN THE SOURCE	MAXIMISE $f(x)$	CONSTRAINT $c(x) \leq \lambda$
Tuning an approximate ML model	speed (shorter test time)	accuracy must match the exact model
Chemical experimental design	process yield	unwanted by-product below a threshold
CPU micro-architecture	processor speed	power usage within a fixed budget

In each case **one experiment returns both numbers**: running x yields $y^F \approx f(x)$ and $y^C \approx c(x)$ together. So the data set grows in pairs, and BO needs a belief about each.

A good predicted objective is not enough. **Feasibility is also uncertain**, and an acquisition score is not a guarantee that every tested design is safe.

Two surrogates, one loop

source p. 179 · the BO loop with a second GP



OBJECTIVE GP

$f(x) \mid \mathcal{D}^F \sim \mathcal{N}(\mu_F(x), \sigma_F^2(x))$ answers: *how much could this design improve performance?* It is the GP of Act 2, unchanged.

CONSTRAINT GP

$c(x) \mid \mathcal{D}^C \sim \mathcal{N}(\mu_C(x), \sigma_C^2(x))$ answers: *how likely is this design to be feasible?* Same machinery, its own kernel hyperparameters θ^C .

Only the acquisition A_t is new — it must read **both** posteriors. The source builds it by editing EI.

Constrained improvement — two edits to EI

source pp. 180–181 · from $I(x)$ to $\Delta(x) \cdot I(x)$

Recall $\text{EI}(x) = \mathbb{E}[I(x) \mid \mathcal{D}]$ with $I(x) = \max(0, f(x) - f(x^+))$. Constraints enter through two edits:

1 · THE INCUMBENT MUST BE FEASIBLE

A high but infeasible result is not a design you could ship, so it is no benchmark. x^+ becomes the best **feasible** observation:

$$f^+ = \max\{y_i^F : y_i^C \leq \lambda\}$$

2 · INFEASIBLE POINTS IMPROVE NOTHING

$$I_C(x) = \Delta(x) I(x), \quad \Delta(x) = \begin{cases} 1 & c(x) \leq \lambda \\ 0 & \text{otherwise} \end{cases}$$

Before the experiment $\Delta(x)$ is unknown — a **Bernoulli** variable, and for $Y \sim \text{B}(p)$, $\mathbb{E}[Y] = 1 \cdot p + 0 \cdot (1 - p) = p$. Its parameter is read off the constraint GP:

$$\mathbb{E}[\Delta(x)] = \text{PF}(x) = P(c(x) \leq \lambda \mid \mathcal{D}^C) = \int_{-\infty}^{\lambda} p(c(x) \mid \mathcal{D}^C) dc = \Phi\left(\frac{\lambda - \mu_C(x)}{\sigma_C(x)}\right)$$

Expected constrained improvement — PF × EI

source p. 182 · where independence is used

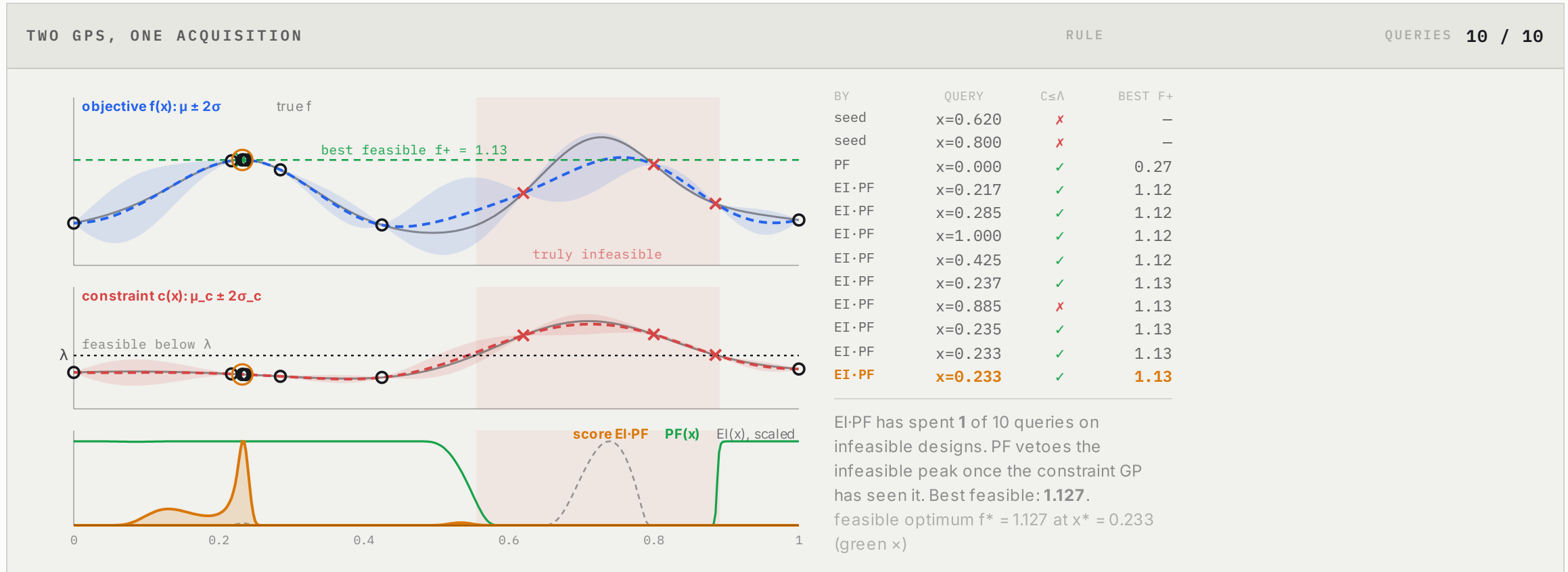
$$\text{EI}_C(x) = \mathbb{E}[\Delta(x) I(x) \mid \mathcal{D}^F, \mathcal{D}^C] \stackrel{\text{indep.}}{=} \underbrace{\mathbb{E}[\Delta(x) \mid \mathcal{D}^C]}_{\text{PF}(x)} \underbrace{\mathbb{E}[I(x) \mid \mathcal{D}^F]}_{\text{EI}(x)}$$

$\Delta(x)$ depends only on $c(x)$ and $I(x)$ only on $f(x)$. With **two separate GPs**, the posteriors of $c(x)$ and $f(x)$ are independent at each x , so the expectation factorises. Both factors are closed-form: the constrained score costs one extra GP prediction.

CANDIDATE	$\text{EI}(x)$	$\text{PF}(x)$	$\text{EI}_C(x)$
A — promising but risky	2.0	0.20	0.40
B — modest but safe	1.0	0.90	0.90

B has less potential improvement but more **expected feasible improvement**. PF acts as a soft veto: $\text{PF} \rightarrow 0$ silences any EI, and $\text{PF} \rightarrow 1$ gives back ordinary EI.

Constrained BO, run — EI × PF against plain EI



Press → to spend one query. Top: objective GP; middle: constraint GP and threshold λ ; bottom: PF (green), EI (dashed) and **the score actually maximised** (amber). Red shading is where $c > \lambda$ — drawn for us, unknown to the algorithm. Switch to *EI alone* to replay the same budget with a rule that ignores c .

Several constraints, and no feasible point yet

source p. 183 · and what the formula leaves open

For constraints $c_i(x) \leq \lambda_i, i = 1, \dots, m$, only PF changes; $\text{EI}_C = \text{PF} \cdot \text{EI}$ stays.

$$\text{PF}(x) = P(c_1(x) \leq \lambda_1, \dots, c_m(x) \leq \lambda_m \mid \mathcal{D}^C) \approx \prod_{i=1}^m P(c_i(x) \leq \lambda_i \mid \mathcal{D}^C)$$

The product is exact when each c_i has its own independent GP. For correlated constraints, use their **joint** posterior instead.

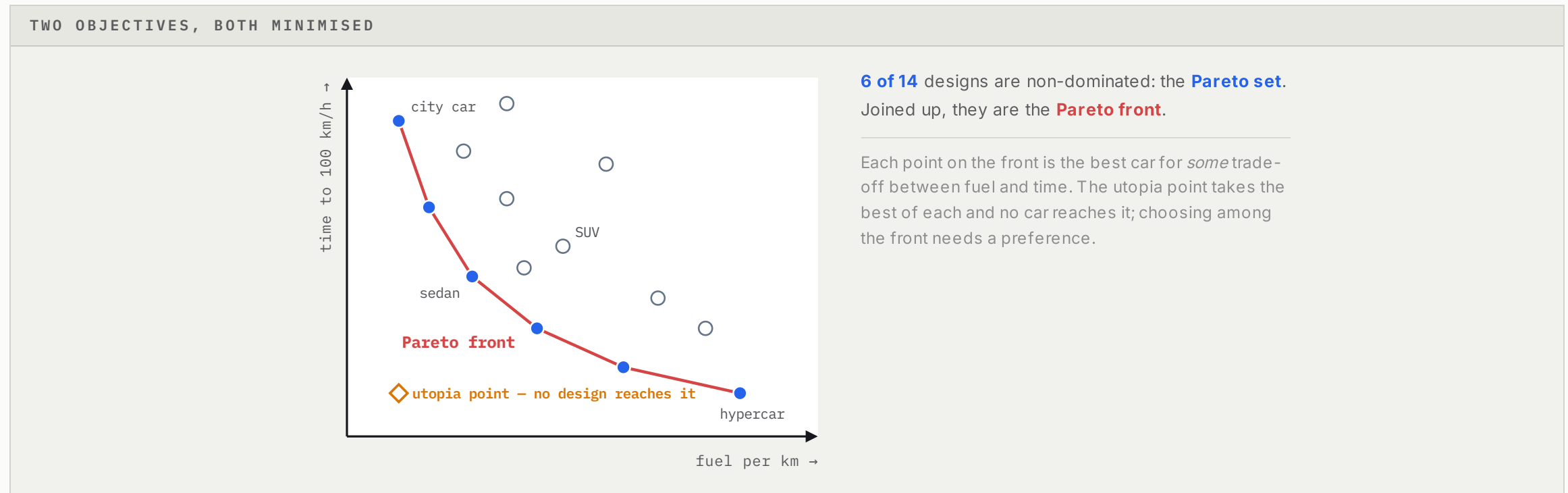
SITUATION	WHAT THE FORMULA SAYS	WHAT TO DO
No feasible observation yet	f^+ is undefined, so EI is too	maximise $\text{PF}(x)$ alone until one feasible point is found
Constraint is cheap or known	nothing to learn about c	check it directly and search only the feasible set (Lecture 1)
An infeasible trial is unacceptable	EI_C only <i>discourages</i> risky queries	<i>safe</i> BO restricts queries to a high-confidence safe set (Sui et al., 2015)

The source closes (pp. 184–185) with Gardner et al.'s 2-D tests: uniform sampling spreads its budget everywhere, plain BO piles queries onto the infeasible optimum, and **constrained BO concentrates on the best feasible region** — the pattern the widget shows in 1-D.

Several objectives — a Pareto set replaces one best number

The fastest car and the most economical car are different cars

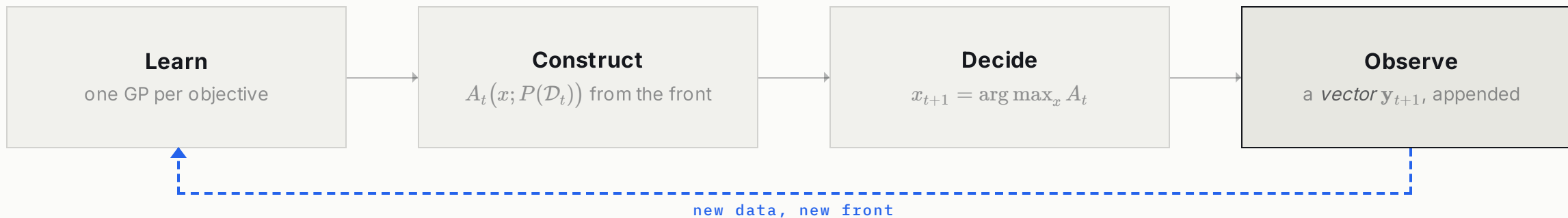
With m objectives the unknown is a vector, $\mathbf{f}(x) = (f_1(x), \dots, f_m(x))$, and no single value can play f^+ . In this section **both objectives are minimised** — less fuel, less time — so better lies toward the lower left.



Press → to walk it; click any design to test it. $\mathbf{y}'' \succ \mathbf{y}'$, **dominates**: no worse in every objective, strictly better in one. **Pareto set**: $P(Y) = \{\mathbf{y}' \in Y : \nexists \mathbf{y}'' \in Y, \mathbf{y}'' \succ \mathbf{y}'\}$.

Multi-objective BO — the same loop, with a front where f^+ was

One GP per objective; only the acquisition function is new



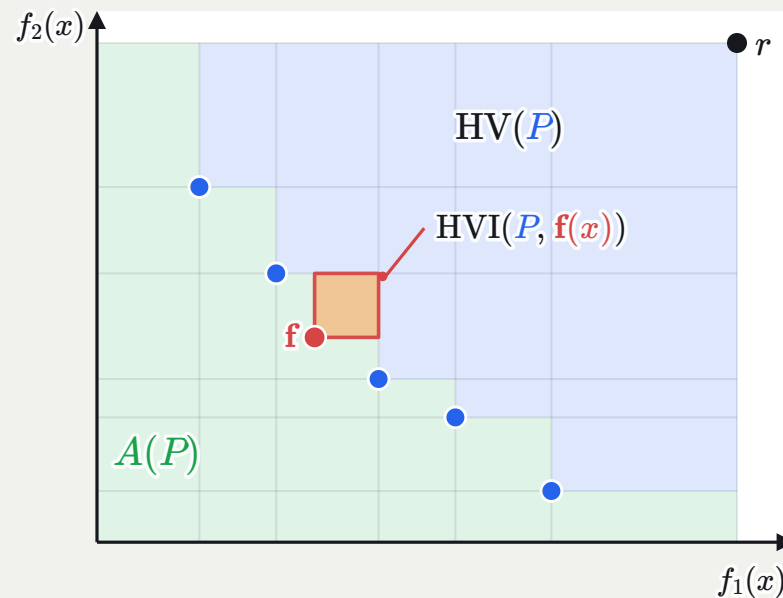
	ONE OBJECTIVE	SEVERAL OBJECTIVES
The best so far	the incumbent f^+	the Pareto front $P(\mathcal{D}_t)$
What one outcome improves	how far $f(x)$ beats f^+	the hypervolume it adds to P , $\text{HVI}(P, \mathbf{f}(x))$
Chance of any improvement	PI	PHVI
Improvement, on average	EI	EHVI
A cheaper stand-in	—	$\text{HVPI} = \text{HVI}(P, \mu) \times \text{PHVI}$

Hypervolume — one number for a whole front

What the front dominates, what one outcome adds, and where that gain is positive

HYPERVOLUME, AND WHAT ONE OUTCOME ADDS

$$\text{HVI}(P, \mathbf{f}(x)) > 0 \text{ for } \mathbf{f}(x) \in A(P), \quad \text{HVI}(P, \mathbf{f}(x)) = 0 \text{ otherwise}$$



$$\mathbf{f}(x) \in A(P) \rightarrow \text{HVI} = 1.00 > 0$$

$A(P)$ is the non-dominated region: an outcome landing there moves the front; one landing in the blue does nothing. Drag $\mathbf{f}$ into the blue and watch the orange vanish.

Press $\rightarrow$ for each piece, and **drag $\mathbf{f}$** anywhere. The reference point r is a deliberately poor corner, fixed before the search, that fences the area off. An outcome dropped into the blue is dominated and adds nothing.

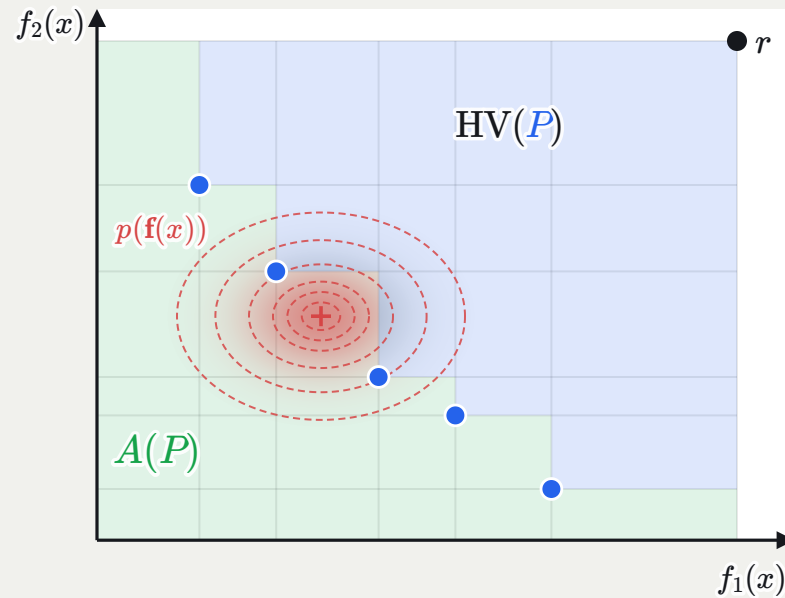
PHVI — will the outcome move the front at all?

Probability of improvement, with “beats f^+ ” replaced by “lands in $A(P)$ ”

PHVI — HOW LIKELY IS ANY GAIN IN HYPERVOLUME

POSTERIOR WIDTH $\sigma_1 = 0.75$

$$\text{PHVI}(P, x, r) = \int_{\mathbf{f} \in A(P)} p(\mathbf{f}(x)) d\mathbf{f}$$



mean $\mu(x) = (3.50, 3.50)$

width $\sigma(x) = (0.75, 0.54)$

PHVI = mass in $A(P)$ = **0.820**

Red mass lands in $A(P)$ and moves the front; grey mass lands where P already dominates. PHVI is the red fraction. Widen σ and watch it fall.

Press → to shade the belief that lands in $A(P)$, then to count 160 draws. Drag the mean, widen σ . The product form assumes one independent GP per objective.

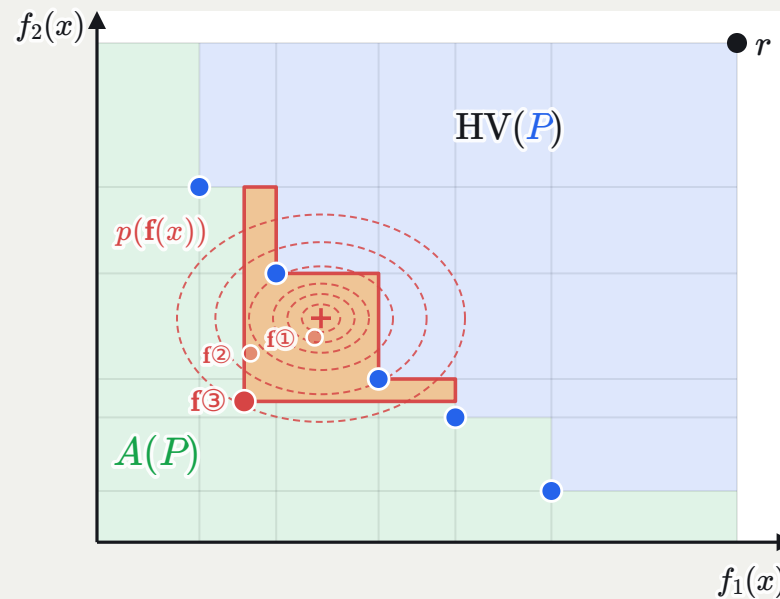
EHVI — how far will it move the front, on average?

Expected improvement for a set — every possible outcome, weighted by the hypervolume it adds

EHVI — THE GAIN, AVERAGED OVER EVERY OUTCOME

POSTERIOR WIDTH $\sigma_1 = 0.75$

$$\text{EHVI}(P, x, r) = \mathbb{E}[\max\{0, \text{HVI}(P, \mathbf{f}(x), r)\}] = \int_{\mathbf{f}(x) \in A(P)} \text{HVI}(P, \mathbf{f}(x), r) p(\mathbf{f}(x)) d\mathbf{f}$$



mean $\mu(x) = (3.50, 3.50)$, width $\sigma(x) = (0.75, 0.54)$

① (3.40, 3.20): $\text{HVI} = 1.00 \cdot p(\mathbf{f})/p(\mu) = 0.85$

② (2.40, 2.95): $\text{HVI} = 3.04 \cdot p(\mathbf{f})/p(\mu) = 0.20$

③ (2.30, 2.20): $\text{HVI} = 5.29 \cdot p(\mathbf{f})/p(\mu) = 0.02$

③ = $0.50 \times 3.35 + 1.60 \times 2.00 + 1.20 \times 0.35$

Outcome ③ clears two Pareto points at once. Rare, but worth five times ①. EHVI keeps both ingredients of EI: **how likely** and **how far**.

Press → for three possible outcomes ①–③ and a dominated one ④, then 160 draws sized by the hypervolume each adds — their average is EHVI.

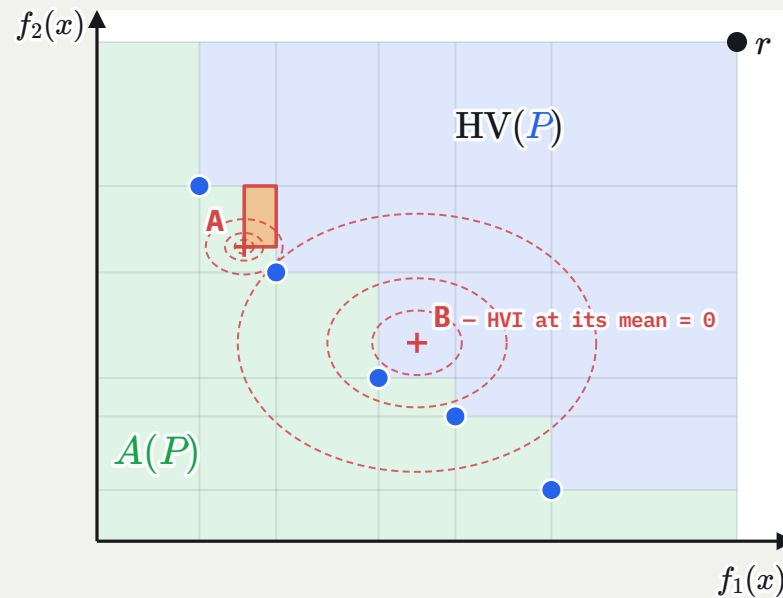
HVPI — the improvement at the mean, times PHVI

A cheaper stand-in for EHVI, and the candidate it cannot see

HVPI — THE GAIN AT THE MEAN, TIMES PHVI

POSTERIOR WIDTH **set per candidate**

$$\text{HVPI} = \text{HVI}(P, \mu) \times \text{PHVI} : \quad \text{HVI}(P, \mu_B) = 0 \Rightarrow \text{HVPI}_B = 0 \text{ for every } \sigma_B$$



	PHVI	EHVI	HVPI
A $\sigma=0.30$	0.954	0.487	0.453
B $\sigma=1.40$	0.428	0.615	0.000
picks	A	B	A

A is a near-certain small gain in the notch. **B**'s mean is dominated, so HVI at its mean is 0 and **HVPI can never choose it** — yet its wide posterior reaches deep into $A(P)$, and EHVI prefers it. The same split as PI against EI.

Press → for the rectangle at the mean, the product, EHVI beside it — then two candidates: **A**, a near-certain small gain; **B**, a wide belief whose mean is dominated.

The bridge — one table, four lectures

MODEL KNOWN — ONLY EXPLOITATION		MODEL UNKNOWN — EXPLORE VS EXPLOIT
Single state (<i>optimum action</i>)	optimisation (<i>Lec 1</i>)	bandit · Bayesian optimisation (<i>Lec 4</i>)
Multiple states (<i>optimum policy</i>)	dynamic programming (<i>Lec 7</i>)	contextual bandit → reinforcement learning (<i>Lec 8, 10</i>)

Read across the bottom row and you have Part IV. Read down the right column and you have this course's central axis. The one missing ingredient between a contextual bandit and full RL is **state dynamics**: in BO your action does not change the world it is asking about. **Let the action move the state, and the bandit becomes reinforcement learning.**

What carries over intact is the explore–exploit machinery of this lecture. What is added is a **value function**, to account for the future an action unlocks and not merely the reward it returns. That is Lecture 7.

Check — what BO assumes it may do

Bayesian optimisation is a loop: fit, choose, evaluate, repeat. Which assumption does the next lecture take away?

- ☐ A That f is smooth enough for a GP prior
- ☐ B That the domain is low-dimensional
- ☐ C That evaluations are noiseless
- ☒ D That you may query f at a point of your choosing and get an answer back

BO earns its sample efficiency by choosing where to look — it needs an **oracle** it can call. Lecture 5 removes exactly that: a fixed dataset, gathered by someone else, and no way to ask a new question. The loop collapses to a single pass, and every safeguard the loop provided has to be rebuilt from inside the model.

— FRONTIER

Beyond the textbook loop — three walls, and the research that climbs them

The loop of Acts 1–3 quietly assumed a **kernel you could choose**, **inputs few enough to cover** and **observations few enough to invert**. Real problems break all three — and each break opened a research stream, ours included.

Three walls between the textbook loop and real problems

Each stream replaces one exact GP calculation with something learned or approximated

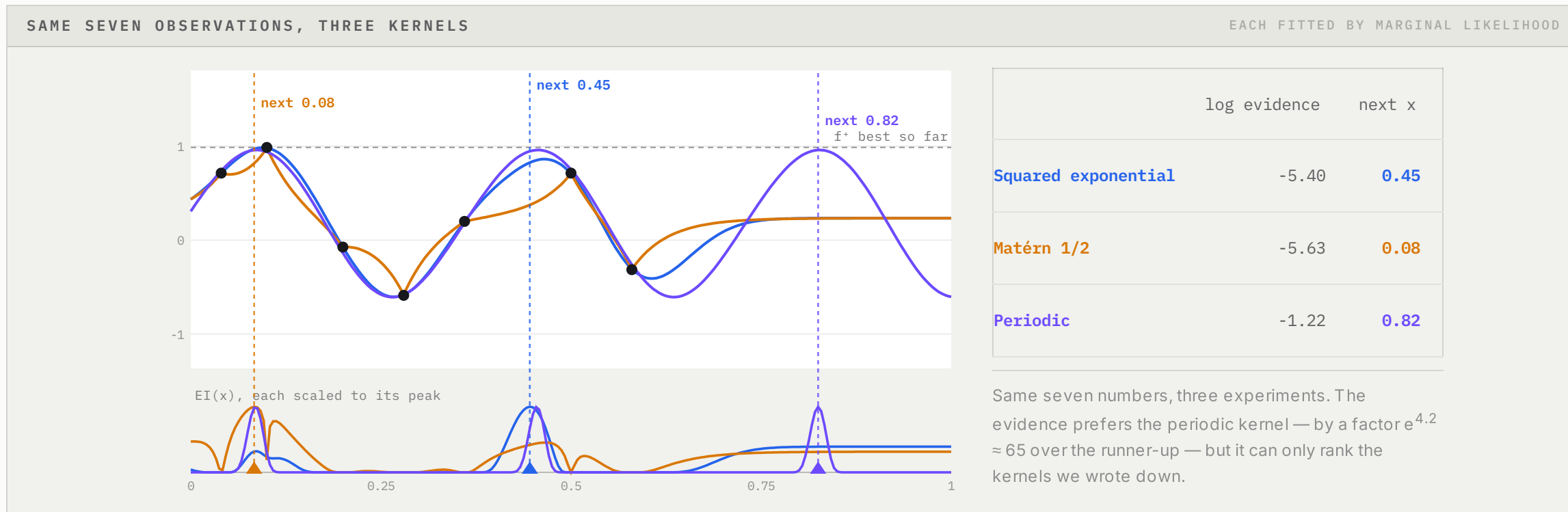
THREE WALLS, AND THE STREAMS THAT CLIMB THEM				
	2006 - 12	2013 - 17	2018 - 21	2022 - 26
Kernel the prior is a guess	Average over θ 12	Spectral mixture 13 Kernel grammar 13 Deep kernel 16 Kernel by BO 16	Neural process 18	Prior-fitted net 22 ④ SM kernel, learned 22 ⑥ Bayesian ConvDeepSets 23 ⑦ Offline meta-BBO 24
Dimension data needed grows as 10^d		Random embedding 13 Learned embedding 14 Additive GP 15 ② Bayesian Ascent 16	Latent-space BO 18 TuRBO 19 ③ CBOTR 20 SAASBO 21	⑧ DiBO 25 ⑨ Latent constrained 25
Data $O(n^3)$ for every refit	Pseudo-inputs 06 Random features 07 Variational sparse GP 09	SVGP 13 DNGO 15 KISS-GP 15 ① Real-time sparse GP 15	GPU solvers 18 Ensemble BO 18	↔ SM kernel, learned 22 ↔ Bayesian ConvDeepSets 23 ↔ DiBO 25 ⑤ GP-emission HMM 22

Numbered in time order, our lab's papers form one thread: from wind-farm control (2015–16) to diffusion-based search (2025). The last slide of this part walks it.

Press → for each wall; click any paper for what it does. **Filled** chips are our lab's papers; a **dashed** chip is a paper that answers a second wall too.

Wall 1—the kernel is a guess, and the guess picks the next experiment

Same seven observations; SE, Matérn-1/2 and periodic kernels, each fitted by marginal likelihood



Press → for each kernel, then all three together. The log evidence $\log p(\mathbf{y} \mid k, \hat{\theta})$ ranks the kernels you tried — it cannot propose one you did not.

Wall 1— average the prior, learn it, or transfer it

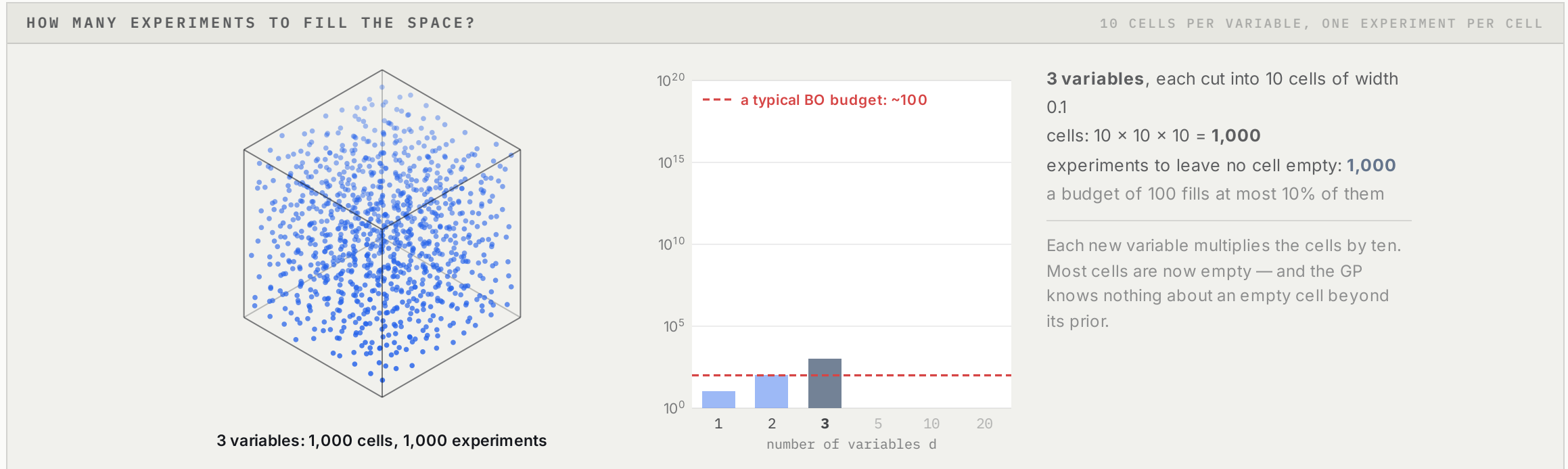
Three ways around a kernel picked by hand

STREAM	IDEA	REPRESENTATIVE WORK	OUR LAB
Average or choose	integrate over θ ; search kernel structures by evidence	Snoek et al. (2012) · Duvenaud et al. (2013) · Malkomes et al. (2016)	
Learn a richer kernel	a spectral mixture can represent any stationary kernel; a network can learn the features	Wilson & Adams (2013) · Wilson et al. (2016)	spectral-mixture kernels learned by random Fourier features and variational inference — Jung, Song & Park (ICML 2022)
Transfer from related tasks	meta-learn the prior: neural processes, prior-fitted networks, pre-trained GPs	Garnelo et al. (2018) · Müller et al. (2022) · Wang et al. (2024)	Bayesian ConvDeepSets with a task-dependent stationary prior — Jung & Park (AISTATS 2023) · offline meta-BBO for traffic lights — Yun, Lee et al. (KDD 2024)

A richer kernel moves the guess from the kernel's **form** into its **parameters** — and many parameters overfit a few points. Taming that is what the 2022 paper does; borrowing other tasks' data is what transfer does.

Wall 2 — too many inputs: the data needed to fill the space grows as 10^d

Cut each variable into 10 cells of width 0.1 — every cell needs at least one experiment



Press → to add variables. One variable has 10 cells, two have 100, three 1 000: each new variable multiplies the count by ten. To leave no cell empty you need at least one experiment per cell, and a typical BO budget fills only the first two.

Wall 2 — assume structure, stay local, or search a learned space

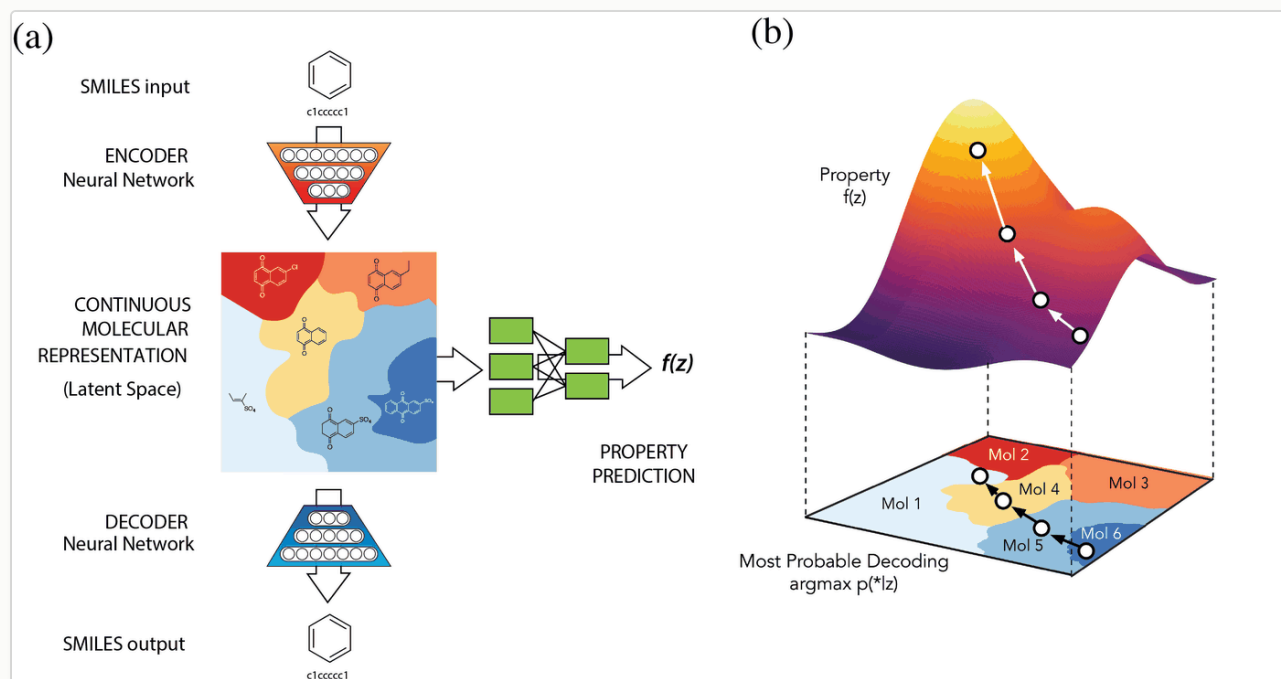
Four ways to make many inputs behave like a few

STREAM	IDEA	REPRESENTATIVE WORK	OUR LAB
Assume structure	f is a sum of low-dimensional parts, or depends on a few directions	Wang et al. (2013) · Kandasamy et al. (2015) · Eriksson & Jankowiak (2021)	
Stay local	model and search only inside a trust region, grown on success and shrunk on failure	Eriksson et al., TuRBO (2019)	Bayesian Ascent — Park & Law (IEEE TCST 2016) · contextual CBOTR — Park (2020)
Search a learned space	optimise in the latent space of a generative model	Gómez-Bombarelli et al. (2018), next slide	latent-space posterior inference with constraints — Om, Sim, Yun et al. (NeurIPS 2025 workshop)
Sample, don't maximise	draw candidates from a posterior: (good designs) $\times$ exp(acquisition)	generative optimisers — Lecture 6	DiBO — Yun, Om et al. (ICML 2025), next slide

There is no universal "BO works in any dimension" guarantee. Each stream buys tractability with an assumption — **few relevant directions, a good local start, a faithful latent space** — so pick the one your problem keeps.

Latent-space BO — turn molecules into points a GP can search

Gómez-Bombarelli et al., ACS Central Science 2018 — the classic of the "search a learned space" stream



(a) Encoder, continuous latent space and decoder, with a property predictor $f(\mathbf{z})$ trained jointly. (b) Climb $f(\mathbf{z})$ in the latent space and decode each point back to a molecule.

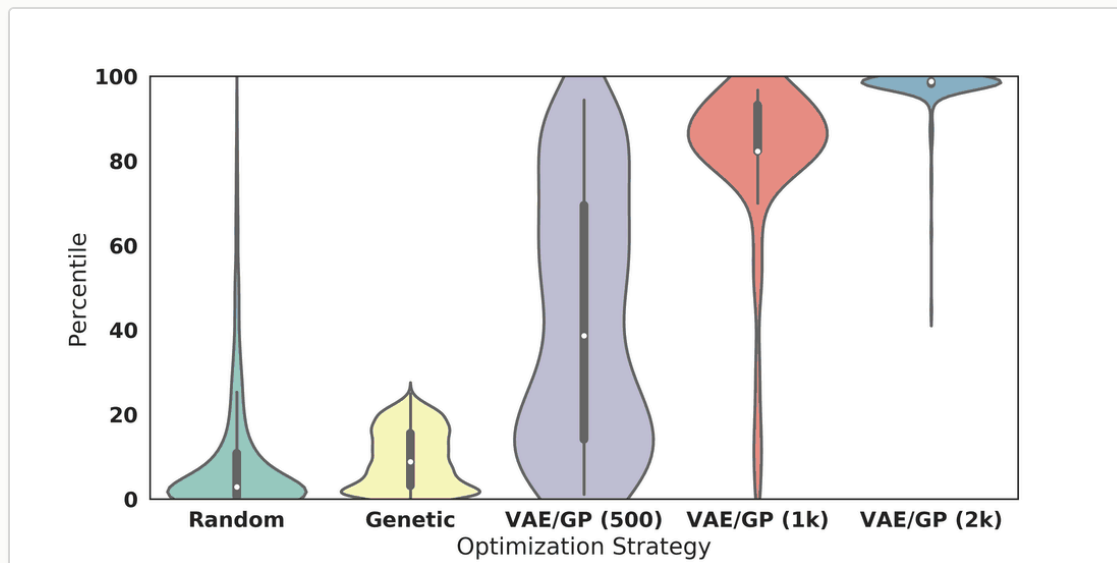
Figure 1 of the paper.

THE RECIPE

1. **Encode.** A VAE trained on 250 000 drug-like molecules (ZINC) maps each SMILES string to a point $\mathbf{z} \in \mathbb{R}^{196}$ — and back.
2. **Organise.** The predictor $f(\mathbf{z})$, trained with the VAE, lays the space out by property: similar molecules, similar values.
3. **Search.** Fit a GP to 2 000 encoded molecules, optimise in $\mathbf{z}$, decode, test.

Latent-space BO — what the search found, and what it assumes

Objective $5 \times \text{QED} - \text{SAS}$: drug-likeness, minus how hard the molecule is to make



Percentile reached from the bottom 10% of ZINC: random search (400 iterations), a genetic algorithm, and latent-space GP optimisation (200 iterations) with the GP trained on 500, 1 000 or 2 000 molecules. Figure 4a of the paper.

WHAT IT ASSUMES

The search is only as good as the space: nearby points must decode to **valid, similar** molecules. Yet only about 74 % of random latent points decode to a valid molecule — the space has "dead areas".

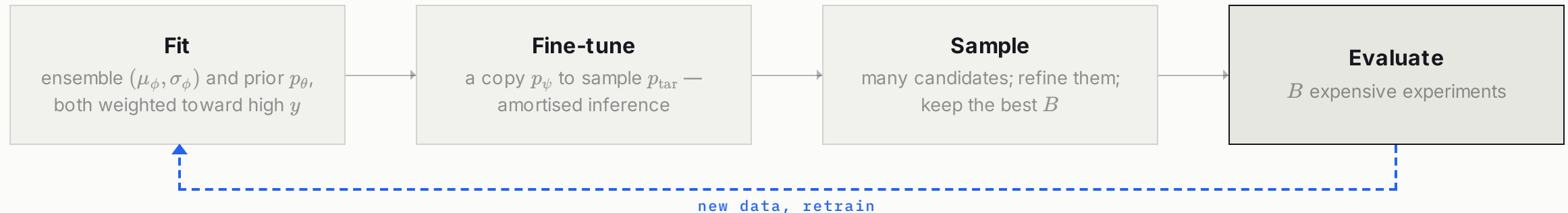
Later work fixed the decoder (grammar and graph VAEs) or reshaped the space while searching (weighted retraining). Lecture 6 builds the VAE itself.

Latent-space BO trades the curse of dimensionality for one assumption: [a faithful latent space](#). Our lab's DiBO, next, replaces the arg-max by posterior inference; its 2025 follow-up runs that inference in a learned latent space, with constraints.

DiBO — sample a posterior instead of maximising the acquisition

Our lab's answer to Wall 2 — and to Wall 3, since no step inverts an $n \times n$ matrix

$$p_{\text{tar}}(\mathbf{x}) \propto \underbrace{p_{\theta}(\mathbf{x})}_{\text{diffusion prior over good designs}} \cdot \exp\left(\beta \underbrace{[\mu_{\phi}(\mathbf{x}) + \gamma \sigma_{\phi}(\mathbf{x})]}_{\text{UCB of a deep ensemble (Act 3)}}\right)$$

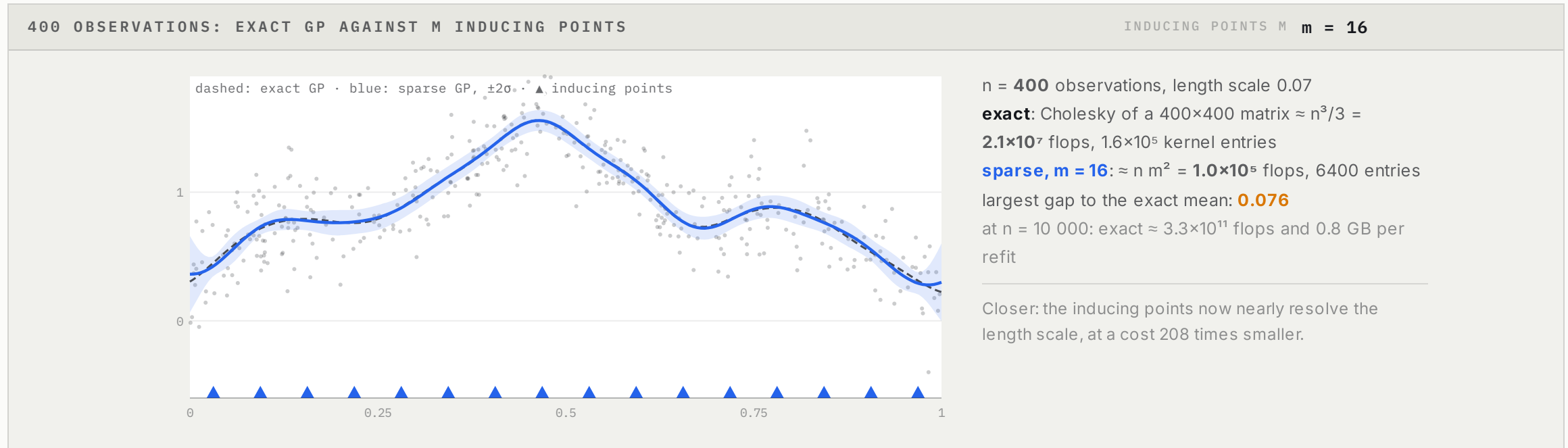


Tested on four synthetic functions with 200 and 400 inputs (10 000 evaluations) and on HalfCheetah-102D, Rover-100D and DNA-180D (2 000 evaluations), against TuRBO, LA-MCTS, MCMC-BO, CMA-ES and generative baselines.

Nothing in this loop is a GP — yet UCB's **exploration bonus survives, as the likelihood**. Lecture 6 builds the diffusion model it rests on.

Wall 3 — too many observations: $O(n^3)$, and a few inducing points

400 noisy observations; the exact GP against a sparse GP built on m inducing points



Press $\rightarrow$ to add inducing points ($\blacktriangle$), or drag m . An exact refit factorises an $n \times n$ matrix; the sparse one, after a single pass over the data, only an $m \times m$ one.

Wall 3 — summarise the data, approximate the kernel, or stop using a GP

Four ways to keep the surrogate cheap as observations pile up

STREAM	IDEA	REPRESENTATIVE WORK	OUR LAB
Summarise the data	m inducing points stand in for n observations: $O(nm^2)$	Snelson & Ghahramani (2006) · Titsias (2009) · Hensman et al. (2013)	real-time sparse GP regression — Park et al. (BigData 2015)
Approximate the kernel	M random features: Bayesian linear regression, $O(nM^2)$	Rahimi & Recht (2007) · Wilson & Nickisch (2015)	random-feature spectral-mixture kernels — Jung, Song & Park (ICML 2022) · GP-emission HMMs (JCGS 2022)
Split the space	a local GP per region or trust region	Wang et al. (2018) · Eriksson et al. (2019)	Bayesian Ascent and CBOTR stay local too
Stop using a GP	networks that carry uncertainty	Snoek et al. (2015) · Lakshminarayanan et al. (2017) · Garnelo et al. (2018)	ConvDeepSets (AISTATS 2023) · DiBO's ensemble (ICML 2025)

Refitting is part of the wall: re-estimate the hyperparameters **on a schedule**, and update the posterior in between.

Our lab's thread — one real problem, followed through the walls

From a wind farm in 2015 to diffusion-based search in 2025

OUR LAB'S THREAD THROUGH THE THREE WALLS				
	2006 - 12	2013 - 17	2018 - 21	2022 - 26
Kernel the prior is a guess				④ SM kernel, learned 22 ⑥ Bayesian ConvDeepSets 23 ⑦ Offline meta-BBO 24
Dimension data needed grows as 10^d		② Bayesian Ascent 16	③ CBOTR 20	⑧ DiBO 25 ⑨ Latent constrained 25
Data $O(n^3)$ for every refit		① Real-time sparse GP 15		→ SM kernel, learned 22 → Bayesian ConvDeepSets 23 → DiBO 25 ⑤ GP-emission HMM 22

Our lab's papers in the order they appeared: ① Real-time sparse GP (2015) · ② Bayesian Ascent (2016) · ③ CBOTR (2020) · ④ SM kernel, learned (2022) · ⑤ GP-emission HMM (2022) · ⑥ Bayesian ConvDeepSets (2023) · ⑦ Offline meta-BBO (2024) · ⑧ DiBO (2025) · ⑨ Latent constrained (2025). Press → to walk them.

Press → to walk the papers in time order. Each step replaced an exact GP calculation with something **learned** — so each inherits Lecture 5's warning: **a learned surrogate's errors become the optimiser's target.**

— CLOSING

Closing

Part II is complete: belief built, belief structured, belief put to work.

Where we are — Part II complete

MODEL - BASED		DATA - DRIVEN
Static, single	optimisation (<i>Lec 1 ✓</i>)	Bayesian stats · network (<i>Lec 2–3 ✓</i>) → Bayesian optimisation (<i>Lec 4 ✓</i>)

What this lecture hands on is **the acquisition policy — the seed of an RL policy — and the bandit**. Two roads lead out from it.

- **Lectures 5–6** — what if you **cannot query f at all**, and hold only a fixed dataset? Offline design optimisation, forward and inverse. Bayesian optimisation minus the loop.
- **Lectures 7–10** — what if the action **moves the world**? The loop stays, a value function is added, and the bandit becomes reinforcement learning.

Honest limits, before you go: dense exact GP training costs $O(n^3)$, and high-dimensional acquisition optimisation can be difficult. The practical range depends on evaluation budget, effective dimension, kernels and approximations. Latent-space and neural surrogates push on both walls, and the first of them, learning a low-dimensional representation to optimise inside, is Lecture 6.

A known objective can be optimised; an unknown,
expensive one must be learned while it is
optimised.

A Gaussian process for the belief, an acquisition function to weigh learning against winning, and a loop that — quietly, three lectures before the words appear — is the first reinforcement learner in the course.

Questions?

The acquisition function is a policy over a belief state. Everything Part IV does is add dynamics to that sentence.

— APPENDIX

Backup slides

Derivations and the bandit toolkit, kept out of the narrative.

Reading guide — from an expensive experiment to the next measurement

one main idea to explain, one comparison, one application

ROLE	READ OR REVISIT	QUESTION TO ANSWER
Core	Frazier, <i>A Tutorial on Bayesian Optimization</i> (2018)	How do GP regression and an acquisition rule form one loop?
Compare	Jones, Schonlau & Welch, <i>Efficient Global Optimization of Expensive Black-Box Functions</i> (1998)	Why evaluate expected improvement instead of only the predicted optimum?
Apply	Snoek, Larochelle & Adams, <i>Practical Bayesian Optimization of Machine Learning Algorithms</i> (NeurIPS 2012)	How does the same loop select expensive model-training hyperparameters?

Understand one GP update and one acquisition choice first. The original traffic and wind cases then add context; multi-output, constrained and multiobjective BO extend the same loop.

Backup 1—the GP posterior, from one Gaussian fact

The fact. If $\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}\right)$, then

$$Y_2 \mid Y_1 = y \sim \mathcal{N}(\mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(y - \mu_1), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12})$$

The application. Take $Y_1 = \mathbf{y}_{1:n}$ (observed, with σ_ϵ^2 on the diagonal) and $Y_2 = f(x)$ (the value we want). Then $\Sigma_{11} = \mathbf{K} + \sigma_\epsilon^2 \mathbf{I}$, $\Sigma_{21} = \mathbf{k}^\top$, $\Sigma_{22} = k(x, x)$, and with $m \equiv 0$:

$$\mu(x \mid \mathcal{D}) = \mathbf{k}^\top (\mathbf{K} + \sigma_\epsilon^2 \mathbf{I})^{-1} \mathbf{y}, \quad \sigma^2(x \mid \mathcal{D}) = k(x, x) - \mathbf{k}^\top (\mathbf{K} + \sigma_\epsilon^2 \mathbf{I})^{-1} \mathbf{k} \quad \blacksquare$$

Three readings. The mean is a **linear combination** of observed y values; its weights need not form an average. The variance shrinks near data and returns to the prior $k(x, x)$ far from it. For fixed hyperparameters, the variance depends on the inputs and noise level, not directly on $\mathbf{y}$ — which is why a GP can plan an experiment before running it. The cost is the factorisation: $O(n^3)$ once, $O(n^2)$ per prediction, one reason dense exact GPs become expensive as the dataset grows.

Backup 2 — kernels, and hyperparameters by marginal likelihood

Squared exponential. $k(x, x') = \sigma_0^2 \exp(-\frac{1}{2}\|x - x'\|^2/\lambda^2)$ — stationary, infinitely differentiable. **Matérn $\frac{3}{2}$:** $\alpha(1 + \sqrt{3}r)e^{-\sqrt{3}r}$; **Matérn $\frac{5}{2}$:** $\alpha(1 + \sqrt{5}r + \frac{5}{3}r^2)e^{-\sqrt{5}r}$, with $r = \|x - x'\|_2/l$ — finitely differentiable, rougher, usually more realistic. **ARD:** one λ_d per dimension, $k = \sigma_0^2 \exp[-\frac{1}{2} \sum_d ((x_d - x'_d)/\lambda_d)^2]$, and a large λ_d means slow variation along dimension d on the studied range. **Algebra:** $k_1 + k_2$ is the covariance of a sum of independent GPs; $k_1 k_2$ is a valid covariance, but multiplying GP sample paths does not generally produce a GP — so $\text{Lin} + \text{Per}$ is *periodic with a trend*, $\text{Lin} \times \text{Per}$ is *growing amplitude*.

Fitting $\theta = (\sigma_\epsilon, \sigma_0, \lambda)$. Marginalise the latent $\mathbf{f}$ away and maximise what is left:

$$\theta^* = \arg \max_{\theta} \log \int p(\mathbf{y} \mid \mathbf{f}, \theta) p(\mathbf{f} \mid \theta) d\mathbf{f} = \arg \min_{\theta} \left[\frac{1}{2} \mathbf{y}^\top (\mathbf{K}_\theta + \sigma_\epsilon^2 \mathbf{I})^{-1} \mathbf{y} + \frac{1}{2} \log |\mathbf{K}_\theta + \sigma_\epsilon^2 \mathbf{I}| \right] = \arg \max_{\theta} [\mathcal{F}(\theta) + \mathcal{S}(\theta)]$$

The two terms pull opposite ways as the length scale grows. On the seven-point example of the Act 2 widget, the simplicity term $\mathcal{S} = -\frac{1}{2} \log |\mathbf{K} + \sigma_\epsilon^2 \mathbf{I}|$ — the negative of Rasmussen & Williams' *complexity penalty*, so that larger is better — climbs monotonically from -0.04 at $\lambda = 0.05$ to $+12.2$ at $\lambda = 10$, because a rigid model is *rewarded* for being prepared for little, while the data-fit term $\mathcal{F} = -\frac{1}{2} \mathbf{y}^\top (\mathbf{K} + \sigma_\epsilon^2 \mathbf{I})^{-1} \mathbf{y}$ falls from -0.75 near $\lambda = 0.45$ to -58.9 at $\lambda = 10$. Their sum $\mathcal{F} + \mathcal{S}$ peaks at $\lambda \approx 0.85$. That is an Occam's razor you did not have to write down, and it is a Lecture 1 optimisation nested inside the Lecture 4 loop.

Backup 3 — Expected Improvement, in closed form

With $f(x) \sim \mathcal{N}(\mu, \sigma^2)$ and incumbent f^+ , define $I = \max(0, f(x) - f^+)$. Integrate:

$$\text{EI}(x) = \int_{f^+}^{\infty} (f - f^+) p(f \mid \mathcal{D}) df = \sigma(x) \left[\underbrace{\frac{\mu - f^+}{\sigma} \Phi(z)}_{\text{exploit}} + \underbrace{\phi(z)}_{\text{explore}} \right], \quad z = \frac{\mu - f^+}{\sigma}$$

with Φ, ϕ the standard normal CDF and PDF. Adding a margin ξ gives the general form $\text{EI} = (\mu - f^+ - \xi)\Phi(Z) + \sigma\phi(Z)$, $Z = (\mu - f^+ - \xi)/\sigma$, and at $\sigma = 0$ use $\text{EI} = \max(0, \mu - f^+ - \xi)$.

Reading the two terms. $(\mu - f^+)\Phi(z)$ is large where the mean already beats the incumbent; $\sigma\phi(z)$ is large where the uncertainty is high, even if the mean is unremarkable. **Why EI and PI differ.** PI integrates the *density* above the line — it counts whether an improvement happens. EI integrates the density *weighted by how far above the line it lands* — it counts how big. The basic EI formula needs no explicit exploration margin, although variants use one. On the illustrative Act 3 posterior it moves the query from $x = 0.630$ (worth 1.075) to $x = 0.470$ (worth 1.301, against a true maximum of 1.303).

Backup 4 — the bandit toolkit, three lectures early

The source lecture develops the finite-armed bandit in full before reaching BO. Every rule below reappears in Part IV.

RULE	FORM	REAPPEARS AS
Action value	$Q_t(a) = \frac{r_1 + \dots + r_{k_a}}{k_a}$, greedy $a_t = \arg \max_a Q_t(a)$	the Q -function, Lecture 8
Incremental update	$Q_{k+1} = Q_k + \alpha_k [r_{k+1} - Q_k]$, with $\alpha_k = \frac{1}{k+1}$ or a constant	$new \leftarrow old + step \times (target - old)$ — every RL update, and a constant α gives exponential recency weighting
ε-greedy	$\pi(a) = 1 - \varepsilon + \frac{\varepsilon}{ A }$ if $a = a^*$, else $\frac{\varepsilon}{ A }$	Lecture 8's exploration rule, unchanged
Softmax	$\pi_t(a) = e^{Q_t(a)/\tau} / \sum_b e^{Q_t(b)/\tau}$	the Boltzmann policy; $\tau \rightarrow 0$ recovers greedy
Preference rules	reinforcement comparison $p_{t+1}(a_t) = p_t(a_t) + \beta[r_t - \bar{r}_t]$; pursuit, keeping both Q_t and π_t	preference learning anticipates policy gradients; see the distinction in the later backup slide
UCB	$a_t = \arg \max_i (\mu_i + \sqrt{2 \ln t / n_i})$	$\mu(x) + \kappa \sigma(x)$, this lecture, continuous

For a stationary finite bandit with sufficient sampling, sample averages converge to action means. Under uniform fixed- ε exploration and a unique best arm, its selection probability approaches $1 - \varepsilon + \varepsilon/|A|$, so exploration still incurs a continuing cost.

Backup — the multi-output covariance, with dimensions visible

Kronecker product and the latent-factor construction

For D outputs measured at the same N inputs, stack $\mathbf{f} = [\mathbf{f}_1^\top, \dots, \mathbf{f}_D^\top]^\top$ in **output-major order**.

$$K_{\text{ICM}} = B \otimes K_X, \quad K_{\text{LMC}} = \sum_q B_q \otimes K_q.$$

MATRIX	SIZE
$B_q = A_q A_q^\top$	$D \times D$
$K_q = [k_q(x_i, x_j)]$	$N \times N$
Stacked covariance	$DN \times DN$

Each independent latent group contributes one PSD covariance; adding them preserves positive semidefiniteness. Independent output-specific noise adds $\text{diag}(\sigma_1^2, \dots, \sigma_D^2) \otimes I_N$.

For different input sets per output, use the corresponding covariance entries; the simple complete-grid Kronecker layout need not remain available.

Backup — preference and pursuit are different bandit updates

Keep their relationship to policy gradients precise

METHOD	STORED OBJECT	UPDATE IDEA
Reinforcement comparison	preferences $h(a)$ and a reward baseline $\bar{r}$	increase the chosen arm's preference when its reward exceeds the baseline
Pursuit	action-value estimates $Q(a)$ and probabilities $\pi(a)$	move probabilities toward the currently greedy action a^*

For pursuit, $\pi_{t+1} = (1 - \beta)\pi_t + \beta e_{a^*}$ with $0 < \beta < 1$. This preserves nonnegative probabilities summing to one.

The exact softmax bandit policy gradient instead updates **every** preference:

$$h_b \leftarrow h_b + \alpha(r - \bar{r}) [\mathbf{1}\{b = a\} - \pi(b)].$$

Reward-relative preference learning anticipates Lecture 10, but the source's chosen-arm-only reinforcement-comparison rule is **not** the full softmax policy-gradient formula.

Backup — the source's computational exercises

GPflowOpt-era lab material

ORIGINAL EXERCISE	LEARNING TASK RETAINED HERE
Branin / polynomial functions	distinguish a local search result from the global best; plot best measured value versus query count
Goldstein and other test functions	compare acquisitions using the same starting observations and evaluation budget
Airline-passenger regression	compare initial and fitted kernel hyperparameters; evaluate held-out predictions
Modified Binh multiobjective problem	identify the Pareto set and compare hypervolume before and after a query

The source's package installation screens are historical implementation material. The mathematical exercises do not depend on that specific library. Record the function, bounds, initial points, noise model, budget, and random seed before comparing methods.

An experiment compares **the same problem under controlled settings**. A good-looking posterior plot alone does not measure optimisation performance.

Backup — three ways to combine Gaussian processes and neural networks

These are different model families

SOURCE DIRECTION	CONSTRUCTION	WHAT CHANGES
Deep Gaussian process	compose latent GP layers	nonlinear latent transformation; inference is generally approximate
Deep kernel learning	$k_{\theta}(x, x') = k(\phi_{\theta}(x), \phi_{\theta}(x'))$	a neural network learns the features used by a GP
Conditional neural process	encode a context dataset and predict distributions at query inputs	learn a conditional predictor; it is not automatically a GP posterior

The source also connects variational autoencoders to latent GP models and low-dimensional search. Lecture 6 develops the latent-variable machinery.

“Uses a neural network” does not specify the uncertainty model. Ask what is random, what is fitted, and which posterior calculation is exact or approximate.

— EXTENSION

Extensions — shared outputs and hypervolume acquisition

Read after completing the main route.

More than one output — share information through a latent function

Intrinsic coregionalisation

Suppose one design has two measured responses. A simple shared latent model is $u \sim \mathcal{GP}(0, k)$, $f_1(x) = u(x)$ and $f_2(x) = 2u(x)$.

$$\text{cov} \left(\begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix}, \begin{bmatrix} f_1(x') \\ f_2(x') \end{bmatrix} \right) = \underbrace{\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}}_B k(x, x').$$

OBJECT	WHAT IT RELATES
Scalar kernel $k(x, x')$	different input locations
Output matrix B	different responses
Independent observation noise	uncertainty in each measurement; added separately

Measuring one response can inform another if **the cross-output covariance model is appropriate**. Predicting several outputs is distinct from deciding how to trade off several objectives.

ICM, SLFM and LMC — change which latent patterns are shared

Retain the model hierarchy

Write each output as a linear combination of independent latent GPs. Their covariance always has the form

$$\text{cov}(f_d(x), f_{d'}(x')) = \sum_q (B_q)_{dd'} k_q(x, x').$$

MODEL	RESTRICTION	WHAT THE EXTRA FREEDOM BUYS
ICM	one shared input kernel, $B k(x, x')$	all output relationships share one spatial pattern
SLFM	several kernels, each $B_q = a_q a_q^\top$	each latent factor can have its own length scale
LMC	several kernels and PSD matrices B_q	several shared factors can use each kernel

For D outputs, every B_q is $D \times D$. With R_q latent factors, $B_q = A_q A_q^\top$ has rank at most $\min(D, R_q)$.

The source's many covariance derivations implement one rule: **independent latent contributions add their covariances**. The appendix gives the stacked matrix.

Computing EHVI — more objectives, correlated outputs, batches, contexts

Where the two-objective picture of Act 4 needs more machinery

SITUATION	WHAT CHANGES
Two objectives, independent GPs	exact: split $A(P)$ into strips and integrate each in closed form — differentiable too (Yang et al., 2019)
Three or more objectives	$A(P)$ splits into boxes whose number grows quickly with m ; Monte Carlo is the usual fallback
Correlated objectives	draw $\mathbf{f}(x)$ jointly from a multi-output GP — the covariance of the previous two slides
A batch of q experiments	qEHVI: Monte Carlo over the joint outcome of all q at once (Daulton, Balandat & Bakshy, 2020)
A context c_{t+1} seen first	contextual MOBO: build the front from $\mu(x, c_{t+1})$, then choose x_{t+1} for that context

Multiple outputs describe **what is predicted**; multiple objectives describe **what is valued**. Every variant keeps the same hypervolume improvement and changes only how its expectation is computed.