

Data-Driven Design Optimization — Surrogate-Based

Deciding about designs no one has measured

Jinkyoo Park · KAIST

Offline, part 1 of 2 — learn the objective from a fixed dataset, then decide what to trust

Where we are — the query is taken away

	LECTURE 4 · <i>OBSERVE</i>	LECTURE 5 · <i>TRUST</i>	LECTURE 6 · <i>GENERATE</i>
the question	which experiment to run next	what to believe about designs no one has measured	how to produce valid, good designs
access to f	one evaluation per round	none — a fixed dataset	none — a fixed dataset
uncertainty is	a reason to look	a reason to be careful	a reason to stay on the data's manifold

The goal is unchanged — a good real design. What is gone is the chance to check one before it is returned.

Learning route — two sessions, one question

Bring: regression and gradient ascent (Lecture 1), the GP posterior and UCB (Lecture 4).

SESSION	WHAT IT DOES
1 · Why it fails	Delete the evaluation → watch the optimiser exploit the surrogate → split the decision loss into two errors → conservatism, with COMs worked to the end
2 · Other grounds for trust	Uncertainty: estimate what the surrogate does not know and decide with it → system knowledge: give it structure or physics → compare the three

Throughout, keep two things apart: **how well a model predicts**, and **how good a decision it makes** when an optimiser is pointed at it.

Lecture 4's loop, with one line deleted

Every black-box method so far closes its loop through the real objective — Bayesian optimisation, and equally a genetic algorithm, CMA-ES, a policy gradient, or gradient ascent on a fitted proxy. Lecture 4's loop:

ONLINE – BAYESIAN OPTIMISATION (LECTURE 4)

```
for t = 1 ... T:
  fit the GP on D_t
  x_{t+1} = argmax_x a_t(x)    ← acquisition, uses σ
  y_{t+1} = f(x_{t+1})        ← query
  D_{t+1} = D_t ∪ {(x_{t+1}, y_{t+1})}
```

OFFLINE (THIS LECTURE)

```
fit a model f_θ on D          ← once
x̂ = argmax_x f_θ(x)          ← no query

return x̂
```

One line deleted, and with it every correction. Online, an over-rated point is queried, found wanting, and the model repairs itself — which is why BO could afford to *seek* uncertainty. Offline, the model is fitted once and never contradicted again.

— SESSION 1 · ACT 1

What breaks offline

Q1. Fit a model of f to the data, then optimise the model. On paper it should work.

Offline model-based optimisation

Q1 What breaks offline?

Q2 Conservatism?

Q3 Uncertainty?

Q4 System knowledge?

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N, \quad y_i = f(x_i) + \epsilon_i$$

$$\text{find } x^* = \arg \max_{x \in \mathcal{X}_{\text{valid}}} f(x) \quad \text{using \textcolor{blue}{only} } \mathcal{D}$$

Evaluating f is expensive or prohibitive — a molecule to synthesise, a protein to express, a chip or an aircraft to build. The source figure is one deleted arrow: data flows into an offline optimiser, a design flows out to superconductors, DNA, proteins and robot bodies, and the return arrow from the design back to the data is struck through — **no additional interactions**.

1 · SURROGATE-BASED — THIS LECTURE

Approximate f from $\mathcal{D}$, then optimise **over the surrogate**. A *forward* model, and a *search*.

2 · GENERATIVE-BASED — LECTURE 6

Learn a representation or distribution of **plausible designs**, then search in it, condition it, or guide it toward better ones.

The bar to beat is the best design you already own

three ways to answer with the same data

RULE	THE DESIGN IT RETURNS	WHAT IT RISKS
Best observed	$\arg \max_{x_i \in \mathcal{D}} y_i$	nothing — and it can never improve
Naive surrogate	$\arg \max_x \hat{f}_\theta(x)$	believing $\hat{f}_\theta$ where no data checked it
Reliable offline optimisation	the maximiser of a score that accounts for out-of-distribution risk	this chapter

Offline optimisation is worth doing only if it beats the best design in $\mathcal{D}$ — and it has to do so without testing the design it returns.

The obvious method — fit a proxy, then climb it

$$\text{Step 1. } \theta^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N (f_{\theta}(x_i) - y_i)^2 \qquad \text{Step 2. } \hat{x} = \arg \max_x f_{\theta^*}(x)$$

Fit a neural network f_{θ} to $\mathcal{D}$ — this is the **surrogate**, or proxy — and then, because f_{θ} is differentiable where the real f was not, run gradient ascent on it, starting from the best design in $\mathcal{D}$.

Step 1 can pass every supervised-learning check. What goes wrong is the interaction: Step 2 searches exactly where Step 1 was **never checked**.

What the benchmark asks for, and how it is scored

the six Design-Bench tasks as used by NEMO, COMs and RoMA (2021)

TASK	DIMENSION	TYPE	DATASET SIZE
Superconductor	81	continuous	21,263
GFP (<i>green fluorescent protein</i>)	238	categorical (20)	5,000
MoleculeActivity	1,024	binary	4,216
HopperController	5,126	continuous	3,200
AntMorphology	60	continuous	25,009
DKittyMorphology	56	continuous	25,009

Three things to keep apart. The *offline data* $\mathcal{D}$ is all a method may use. The *candidates* — 128 designs in the papers of this lecture — are what it returns. The *oracle* scores them afterwards, for the benchmark only, and is never fed back: the physics simulator for the robot tasks, a model fitted to a larger held-out dataset for tasks such as Superconductor. The usual score, the best of the 128, is a benchmark number; deploying one design would still need a way to pick it without the oracle. HopperController has 3,200 designs in 5,126 dimensions: **almost every direction the ascent can take was never checked against anything.**

Two different problems

PROBLEM 1 — VALUES OFF THE DATA · EVERY OFFLINE PROBLEM

The training data constrains the model where it has support. Away from it, a small training loss controls nothing — and the search must go there, because the goal is a design *better than anything in $\mathcal{D}$* . **The failure is the task.**

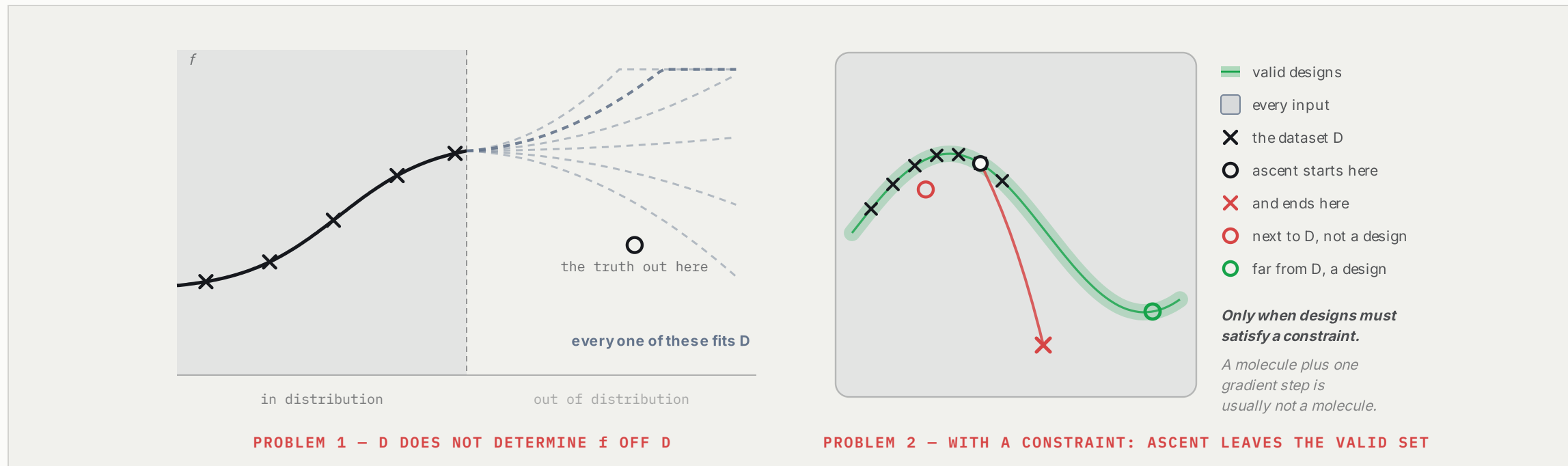
PROBLEM 2 — THE VALID SET · ONLY WITH A CONSTRAINT

When designs must be real molecules, foldable proteins or buildable layouts, only a thin sliver of input space is valid, and ascent can leave it: a design off it is **not merely poor but meaningless**. If every input is a legitimate design, this problem does not exist.

VALID		INVALID
near the data	the data themselves	a molecule string with one character changed — when the valid set is thin
far from the data	the design we hope to find	where unchecked ascent ends

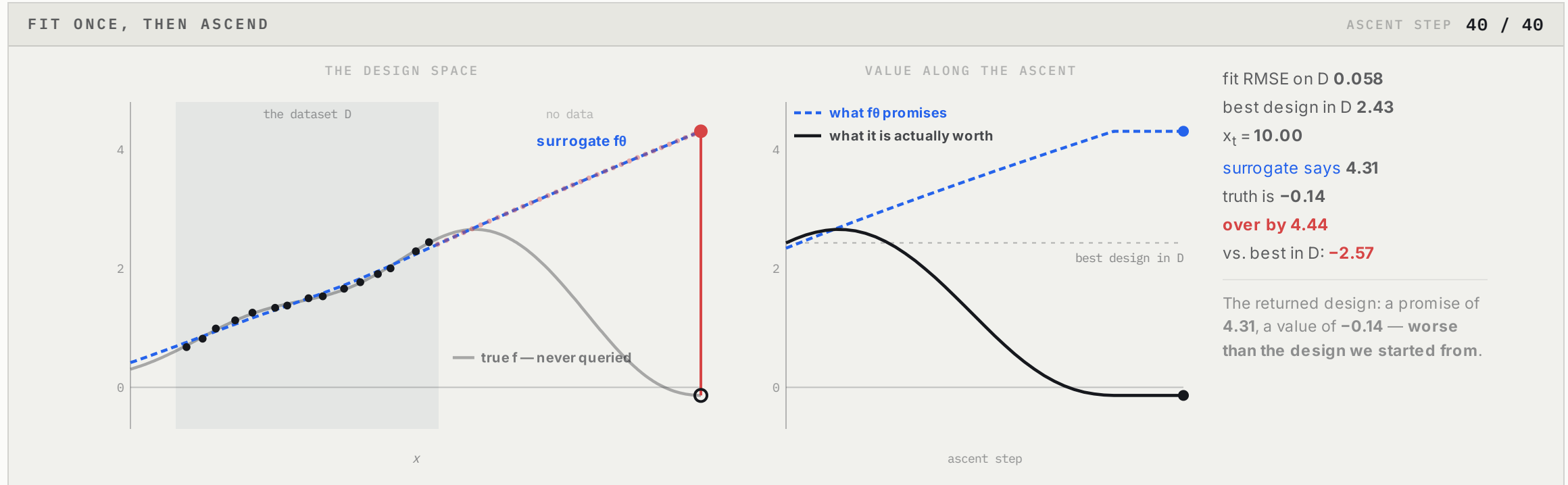
Independent questions. A **known** constraint is Lecture 1's constrained optimisation. An **implicit** one — known only through valid examples — must be learned, and judging it off the data is Problem 1 again, for the constraint. This lecture attacks Problem 1; Lecture 6 learns the valid set.

The two pictures



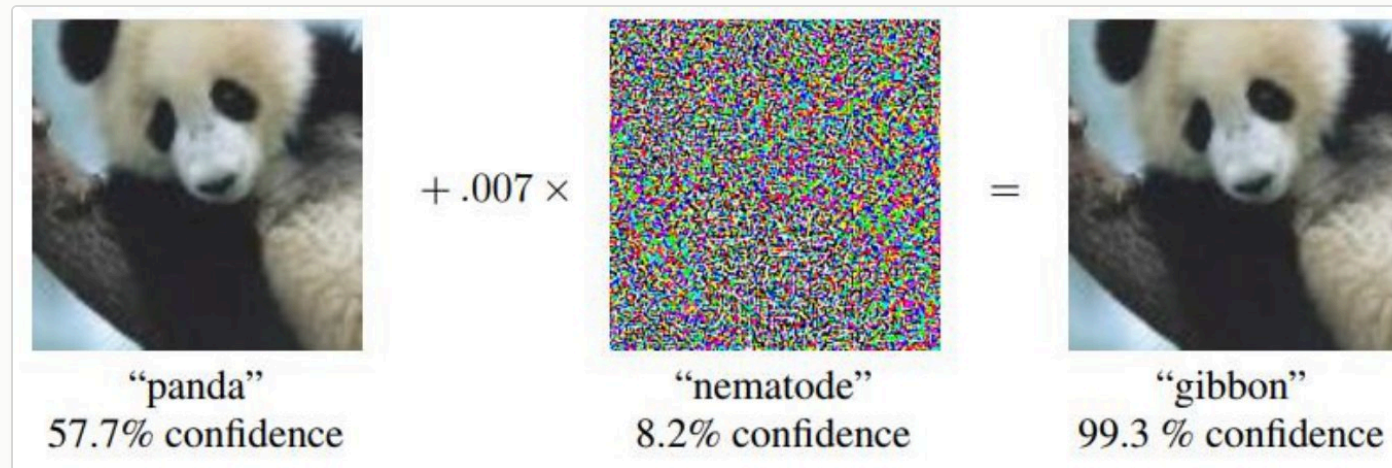
Left, the dataset does not determine f off the data: **every one of those dashed continuations fits D equally well**. Right, only when designs must satisfy a constraint: the valid designs are a thin curve, and ascent from a data point leaves it at once. **A point next to the data can be off the curve, and a point far from it can be on it.**

Watch it happen



Fifteen measured designs, a surrogate that fits them to an RMSE of 0.058, and gradient ascent from the best of them. The faint curve is the true f — drawn so that we can watch; the optimiser never sees it. Press $\rightarrow$ to walk the ascent to $t = 5, 12$ and 40: **it returns -0.14 where the surrogate promised 4.31**, worse than the design it started from.

The optimiser is an adversary



A perturbation of size 0.007, chosen by following a gradient, turns 57.7 % "panda" into 99.3 % "gibbon". (Goodfellow, Shlens & Szegedy, ICLR 2015)

Gradient ascent on $\hat{f}_\theta$ is the same computation aimed at a regression model: it moves the input in whatever direction raises the model's output fastest. In a high-dimensional space with little data, many of those directions lead to errors.

The optimiser does not look for errors. It looks for high predictions — and where there is no data, **high predictions are often errors**.

A good fit can still make a bad decision

the signed decision bound

Let $\hat{x}$ be the design the optimiser returns and x^* the true optimum. Add and subtract $\hat{f}$ at both points:

$$f(x^*) - f(\hat{x}) = \underbrace{[f(x^*) - \hat{f}(x^*)]}_A + \underbrace{[\hat{f}(x^*) - \hat{f}(\hat{x})]}_{\leq 0 \text{ when } \hat{x} \text{ maximises } \hat{f}} + \underbrace{[\hat{f}(\hat{x}) - f(\hat{x})]}_B \leq A + B$$

A — AT THE TRUE OPTIMUM

How much the surrogate **under-rates** the best design. A model too pessimistic about the right answer pays here.

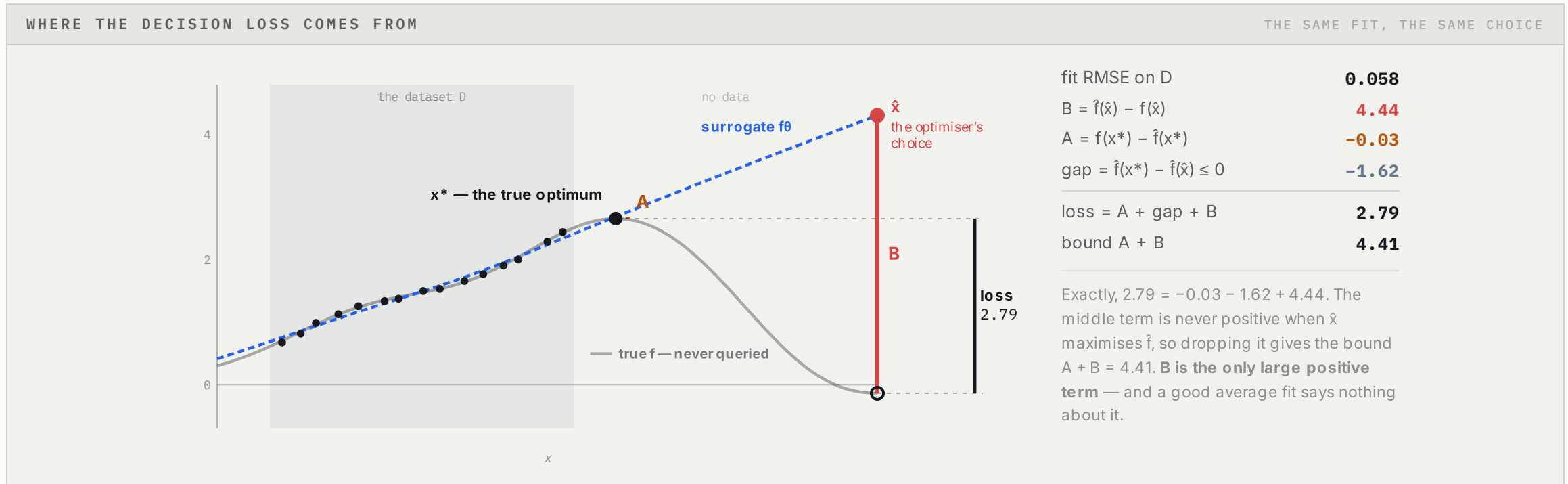
B — AT THE CHOSEN DESIGN

How much the surrogate **over-rates** the design it picked. The optimiser's exploit pays here.

Validation error is an average over the data. The decision depends on the error at two particular points — **and the optimiser chooses one of them.**

The middle term, the **optimiser gap**, is never positive when $\hat{x}$ maximises $\hat{f}$ — that is all the inequality uses. The exact loss keeps it, so $A + B$ can sit well above the loss. A and B themselves can be negative.

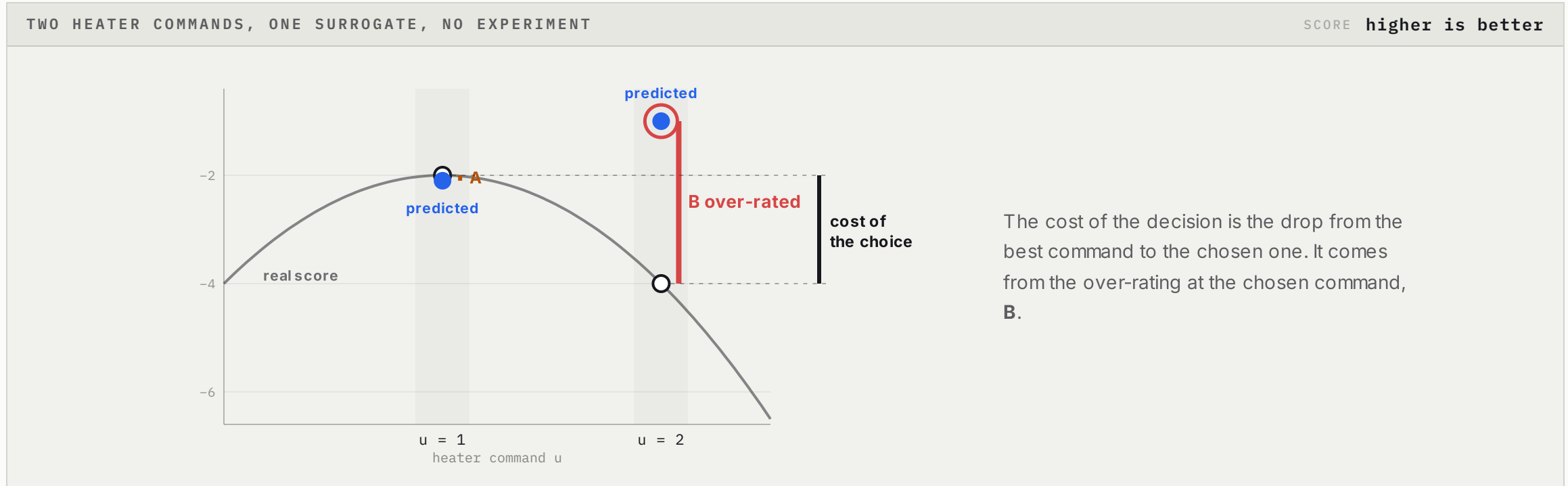
Where the loss comes from, on the same fit



Walk it with $\rightarrow$. The surrogate is **right about the true optimum** — $A = -0.03$ — and wrong about the design it chose, $B = 4.44$. Exactly, $2.79 = A + \text{gap} + B = -0.03 - 1.62 + 4.44$; dropping the gap gives the bound 4.41. **B is the only large positive term**.

Temperature thread — what the surrogate promised, and what it delivered

shared teaching example · predict before revealing



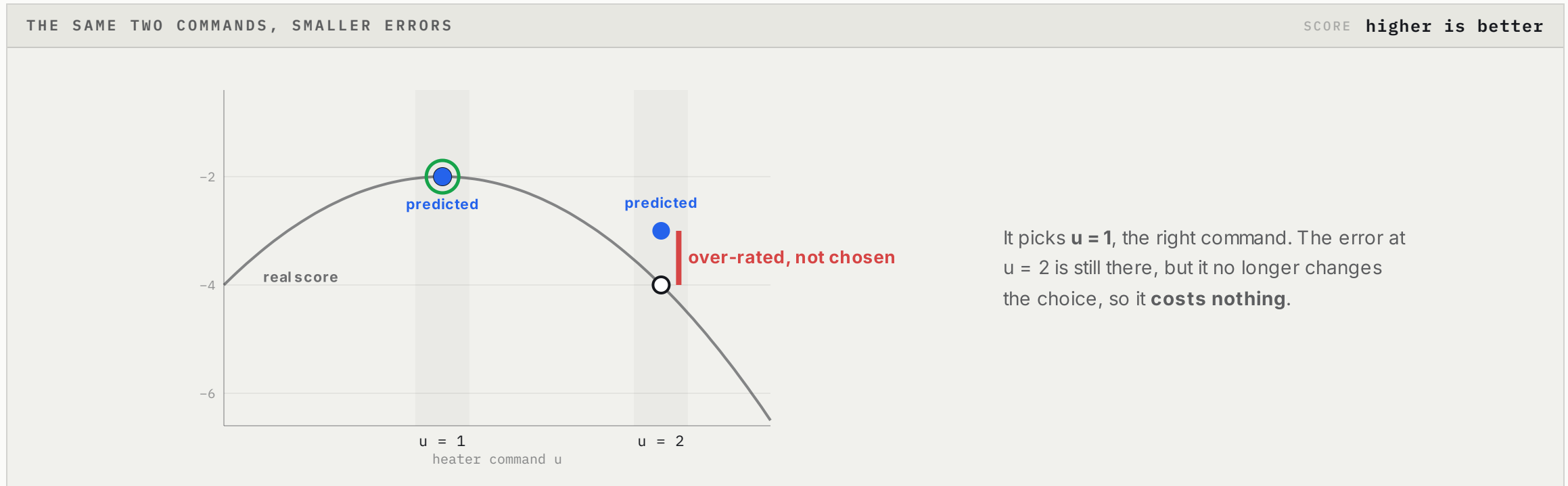
Lecture 0's heater, offline: two commands, one surrogate, no experiment.

Predict: Which command does maximising the surrogate pick — and is it the better one?

It picks $u = 2$, the command it rates higher. Press → to run the heater and see why that is the worse choice.

Try it — the largest error is no longer the selected action

work independently · reveal only after deciding



The same two commands; the surrogate is now exact at $u = 1$ and over-rates $u = 2$ by less.

Which command is chosen now — and does the remaining error at $u = 2$ cost anything?

Check your answer. It picks $u = 1$, the right command. The optimiser maximises truth plus error, not error alone: the error at $u = 2$ is still there, but an error costs the decision only when it **changes the choice**. Press → to see it.

The cure is not a better optimiser

Two repairs suggest themselves first, and both treat the search as the problem.

SEARCH HARDER

A stronger, more thorough optimiser finds a *higher* point of $\hat{f}$ — on this surface, a point still further from the data and still more over-rated. It makes B larger.

Optimisation strength is on the **wrong side** of the problem.

SEARCH LESS

Restricting the search to measured designs caps the result at the best recorded score. A neighbourhood limit can still allow improvement, but it also excludes the novel designs we came for.

The trade is between evidence and novelty.

What has to change is what the surrogate is entitled to say away from the data.

Check — prediction quality versus decision quality

Surrogate S_1 has a lower validation error than surrogate S_2 , yet the design chosen by maximising S_1 is worse. Is that a contradiction?

- ☐ A Yes — a lower validation error must give a better optimised design
- ☐ B No — S_1 must have been overfitted
- ☒ C No — validation error is measured on the data; the decision depends on the error at the design the optimiser selects, usually away from the data
- ☐ D Yes — the optimiser must have failed to converge

Validation error averages over designs like those in $\mathcal{D}$. The decision loss is bounded by $A + B$: the errors at the true optimum and at the chosen design, which the optimiser picks *because* the prediction is high there. A model can win on the first number and lose on the second — the surrogate in *Watch it happen* had an RMSE of 0.058 and $B = 4.44$.

Three ways to give a surrogate grounds for trust

the map for both sessions

CONSERVATISM

Change what $\hat{f}$ says **where the optimiser goes**: a training penalty on the designs it would chase.

COMs (*ICML 2021*) · PRIME · RoMA —
Session 1

UNCERTAINTY

Estimate **what $\hat{f}$ does not know**, and decide with it: $\mu - \kappa\sigma$, Lecture 4's UCB with the sign turned.

Deep Ensembles (*NeurIPS 2017*) · NEMO —
Session 2

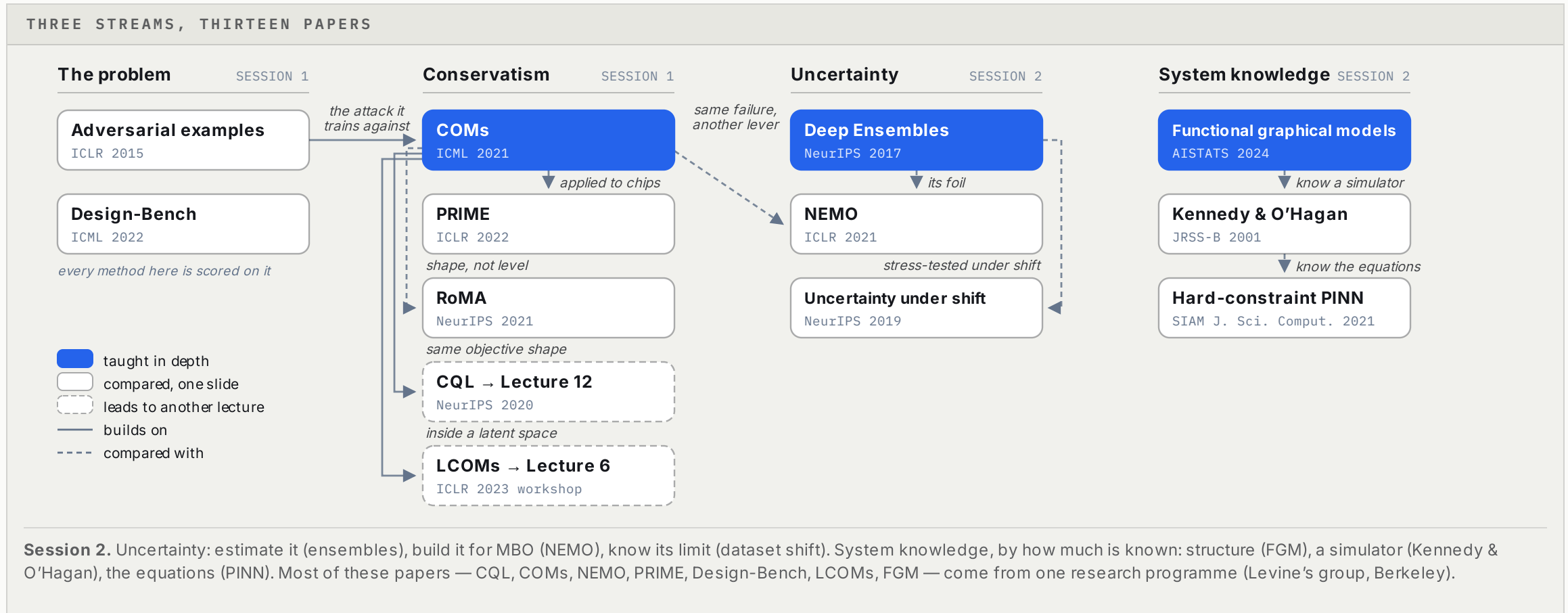
SYSTEM KNOWLEDGE

Give the model **structure or physics**, so that its extrapolation has grounds other than the data.

FGM (*AISTATS 2024*) · Kennedy & O'Hagan —
Session 2

All three try to shrink B ; each brings its own assumptions and can cost something elsewhere — too much conservatism inflates A , as Session 1 will show. They are not rivals: an uncertainty penalty is itself conservative, and a structured model can be trained conservatively.

The papers of this lecture — why each is here



Press → for Session 2's papers; click any paper for what it does, why we read it and how it connects. The same list is a table in the appendix.

— SESSION 1 • ACT 2

Conservatism: train the surrogate against its own optimiser

Q2. If the optimiser will attack the surrogate, let the surrogate see the attack while it is trained.

COMs — what we want, and the obstacle

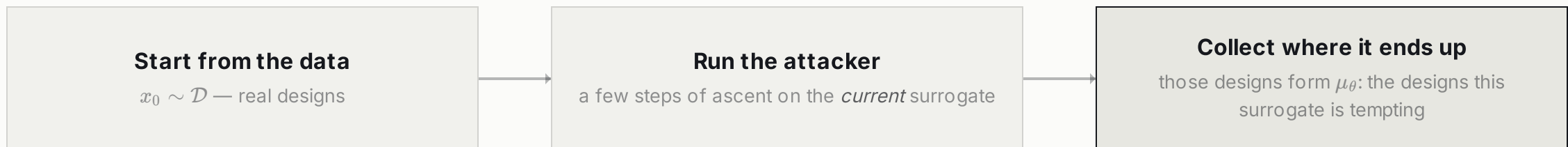
Conservative Objective Models · Trabucco, Kumar, Geng & Levine, ICML 2021



We want a model that **does not over-rate the designs an optimiser would chase** — a small B . The obstacle is knowing which designs those are, and the answer is to generate them by running the attack we fear.

$$\mathbb{E}_{\mu_\theta}[f_\theta] = \frac{1}{M} \sum_{j=1}^M f_\theta(x_T^{(j)}), \quad x_{t+1}^{(j)} = x_t^{(j)} + \eta \nabla_x f_\theta(x_t^{(j)}), \quad x_0^{(j)} \sim \mathcal{D}$$

μ_θ is where T ascent steps from the data **end up**; the loss uses it only through this average.



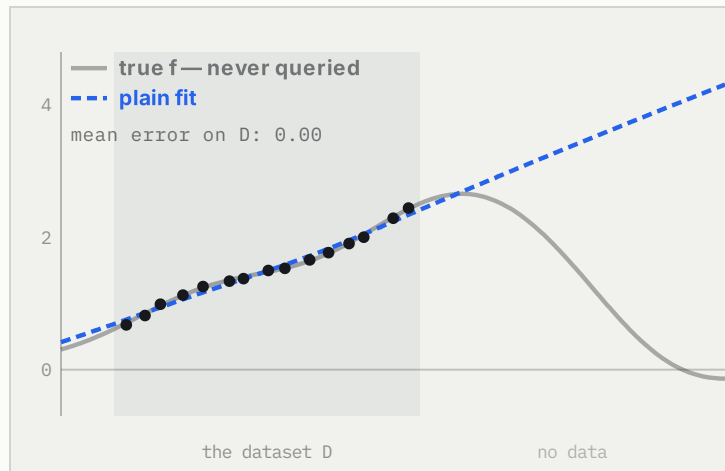
Because f_θ changes at every training step, μ_θ is regenerated as training proceeds — an inner attacker chasing an outer defender, as in adversarial training for robustness.

Three terms, three fits

add a term to the loss and watch the surrogate move · the same fifteen points, $\alpha = 0.3$

(I) FIT THE DATA

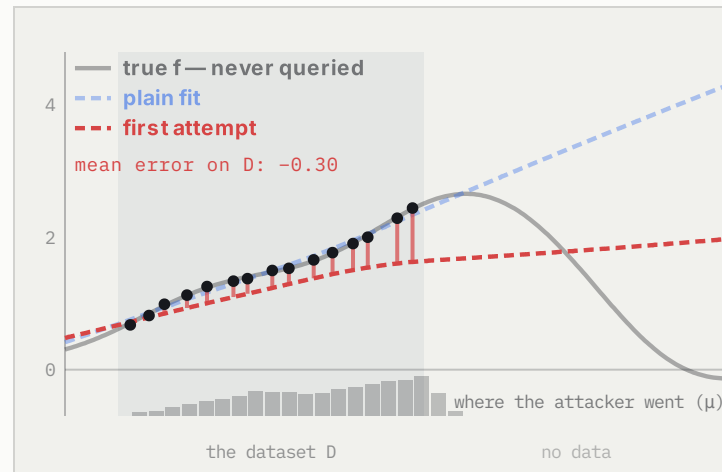
$$\frac{1}{2} \mathbb{E}_{\mathcal{D}} [(f_{\theta} - y)^2]$$



The plain fit: right on the data, and it keeps climbing where there is none.

(II) + PUSH THE ATTACKER'S DESIGNS DOWN

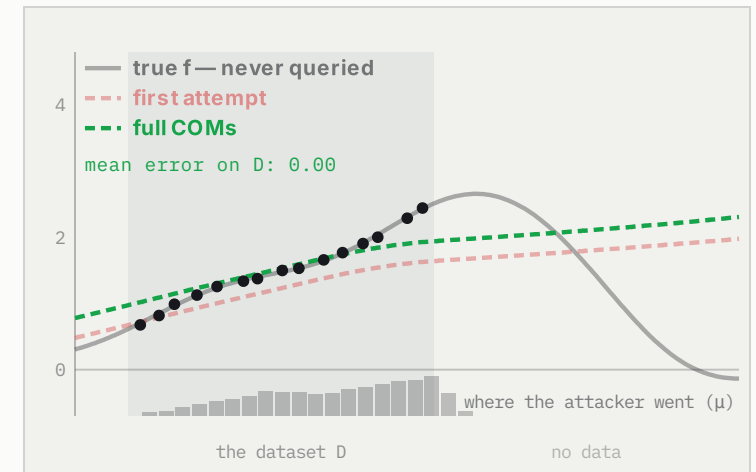
$$\frac{1}{2} \mathbb{E}_{\mathcal{D}} [(f_{\theta} - y)^2] + \alpha \mathbb{E}_{\mu_{\theta}} [f_{\theta}]$$



The cheapest way to lower the attacker's scores is to lower everything: the surface sinks below the very data it was fitted to, by exactly α .

(III) + HOLD THE DATA UP

$$\frac{1}{2} \mathbb{E}_{\mathcal{D}} [(f_{\theta} - y)^2] + \alpha \mathbb{E}_{\mu_{\theta}} [f_{\theta}] - \alpha \mathbb{E}_{\mathcal{D}} [f_{\theta}]$$

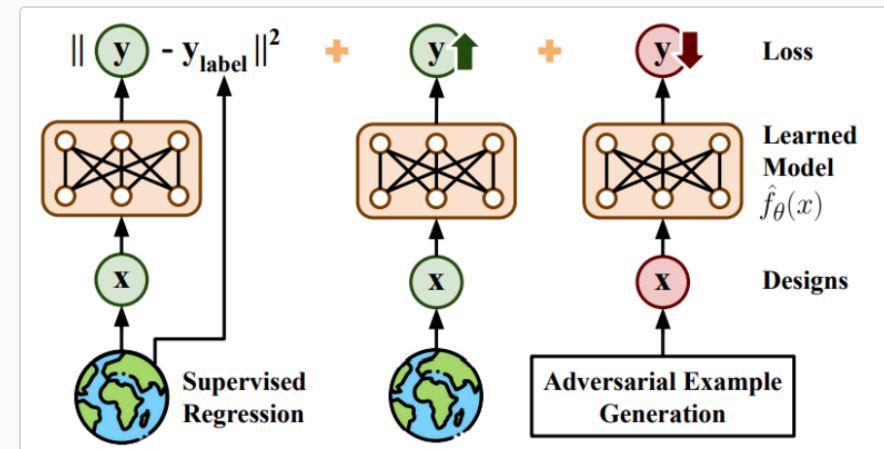


A constant shift now moves both averages alike, so sinking no longer pays: lifted back by α inside the data, the push stays **where the attacker went**.

The loss, term by term

$$L(\theta) = \underbrace{\frac{1}{2} \mathbb{E}_{(x,y) \sim \mathcal{D}} [(f_{\theta}(x) - y)^2]}_{\text{(i) fit the data}} + \alpha \left(\underbrace{\mathbb{E}_{x \sim \mu_{\theta}} [f_{\theta}(x)]}_{\text{(ii) push the attacker's designs down}} - \underbrace{\mathbb{E}_{x \sim \mathcal{D}} [f_{\theta}(x)]}_{\text{(iii) hold the data up}} \right)$$

- **(i)** ordinary regression — be right about the designs we measured;
- **(ii)** prevent over-estimation out of distribution;
- **(iii)** prevent under-estimation in distribution.



The source's picture of the same sum: regression on real designs, a push up on real designs, a push down on designs from adversarial generation.

COMs as an algorithm — train against an attacker, then search slowly

ALGORITHM 1 — TRAIN THE CONSERVATIVE SURROGATE

```

initialise  $f_\theta$ ; pick  $\eta, \alpha$ 
repeat:
  sample  $(x_0, y) \sim D$ 
   $x_K \leftarrow K$  ascent steps from  $x_0$  on  $f_\theta$ 
   $\mu \leftarrow$  empirical distribution of the  $x_K$ 
   $L = \frac{1}{2} E_D (f_\theta(x_0) - y)^2$ 
     $+ \alpha E_\mu[f_\theta] - \alpha E_D[f_\theta]$ 
   $\theta \leftarrow \theta - \lambda \nabla_\theta L$ 

```

Left — learn. Ordinary regression on $\mathcal{D}$ plus the penalty of the last two slides. The attacker is re-run on the *current* f_θ every round, so the surrogate is flattened exactly where it was tempting. Out comes one model, $\hat{f}_{\theta^*}$.

ALGORITHM 2 — SEARCH IT, SLOWLY

```

 $x_0 = \operatorname{argmax}_{\{(x,y) \in D\}} y \quad \leftarrow$  the best design we own
for  $t = 0 \dots T-1$ :  $\leftarrow T = 450$  in the paper
   $x_{t+1} = x_t + \eta \nabla_x L_{\text{opt}}(x_t)$ 
   $\leftarrow L_{\text{opt}}: \hat{f}$  with a brake, next slide
return  $\hat{x} = x_T$ 

```

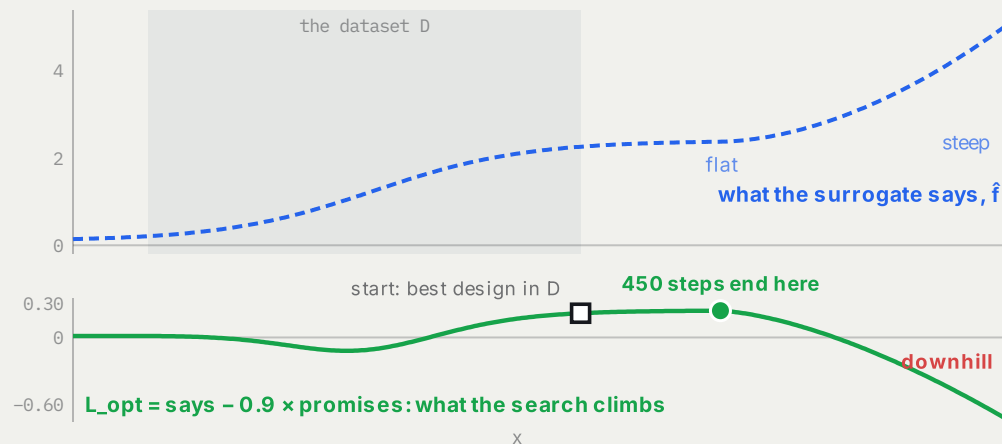
Right — search. Start from the best design we own and take T small steps uphill — not on $\hat{f}_{\theta^*}$ itself but on L_{opt} , a braked copy of it that distrusts what the surrogate promises one step further on.

Training flattens the surrogate on a distribution of challenging designs near the data. The final search is deliberately slow and short, so that **within its budget it does not get far from where conservatism was learned**.

The brake — what Algorithm 2 climbs

a trust region, as in Lecture 1: do not trust the model further than one step

$$L_{\text{opt}}(x) = \underbrace{\hat{f}_{\theta^*}(x)}_{\text{what the surrogate says here}} - \beta \underbrace{\hat{f}_{\theta^*}(x + \eta \nabla_x \hat{f}_{\theta^*}(x))}_{\text{what it promises one trial step up}}, \quad \beta = 0.9$$



P, flat: keeps **0.23** — a tenth of what it says.

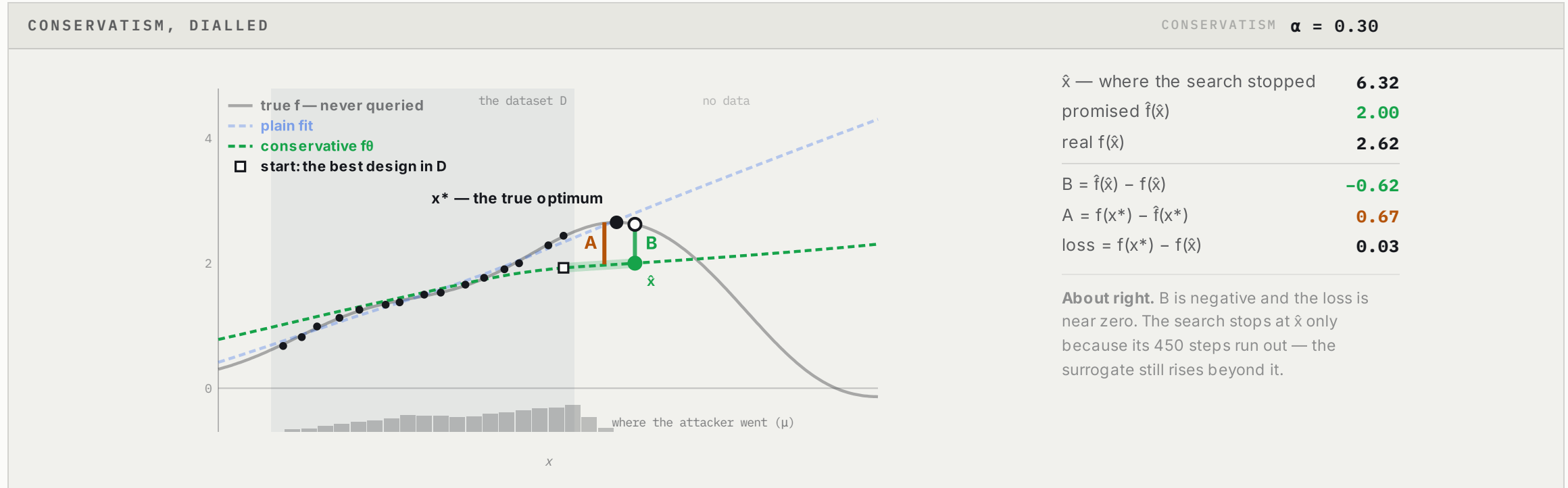
Q, steep: keeps **-0.24** — the promise is subtracted.

What the search climbs. Flat and high wins; the steep edge, where plain ascent would run, is a valley. 450 steps from the best design end at $x = 6.9$, on the plateau.

A sketch, not the toy; η enlarged so the trial step shows.

To second order, $L_{\text{opt}} \approx (1 - \beta) \hat{f} - \beta \eta \|\nabla \hat{f}\|^2$: a tenth of the climb, minus a penalty on steepness. $T = 450$ steps is a budget, not convergence.

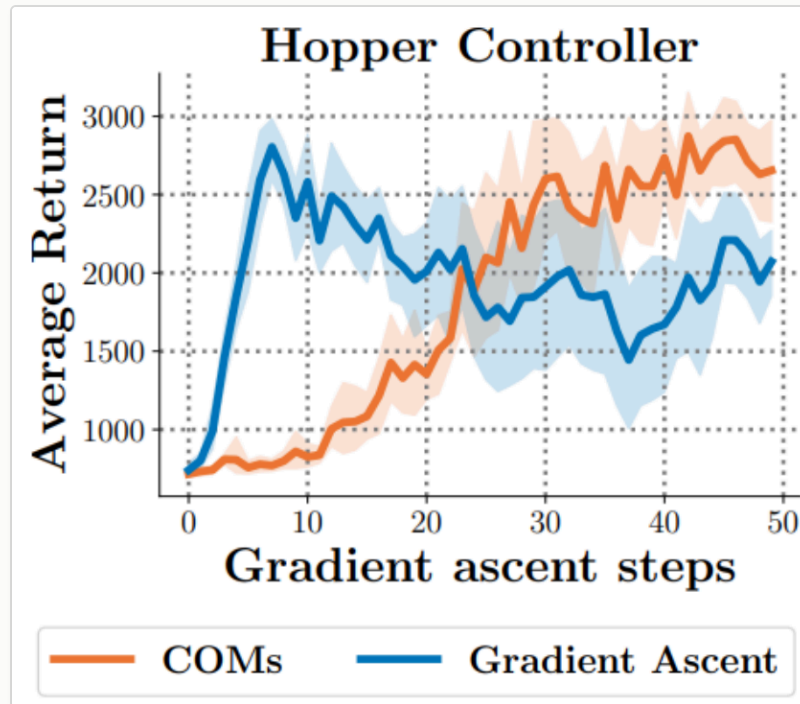
Turning the dial



The same picture as Act 1: at $\hat{x}$ the dot is what the surrogate **promised**, the ring what $\hat{x}$ is **really worth**. The search always takes **the same 450 slow steps** from $\square$; only the penalty α changes. Too little and it runs to the edge (B large); too much and it stops short of x^* (A large).

The same shape on a real task

COMs against naive gradient ascent · HopperController, 5,126 dimensions



Average return of the design after each ascent step, as the source deck shows it. The published version (Trabucco et al., 2021, Fig. 3) runs to 450 steps and shows the same shape.

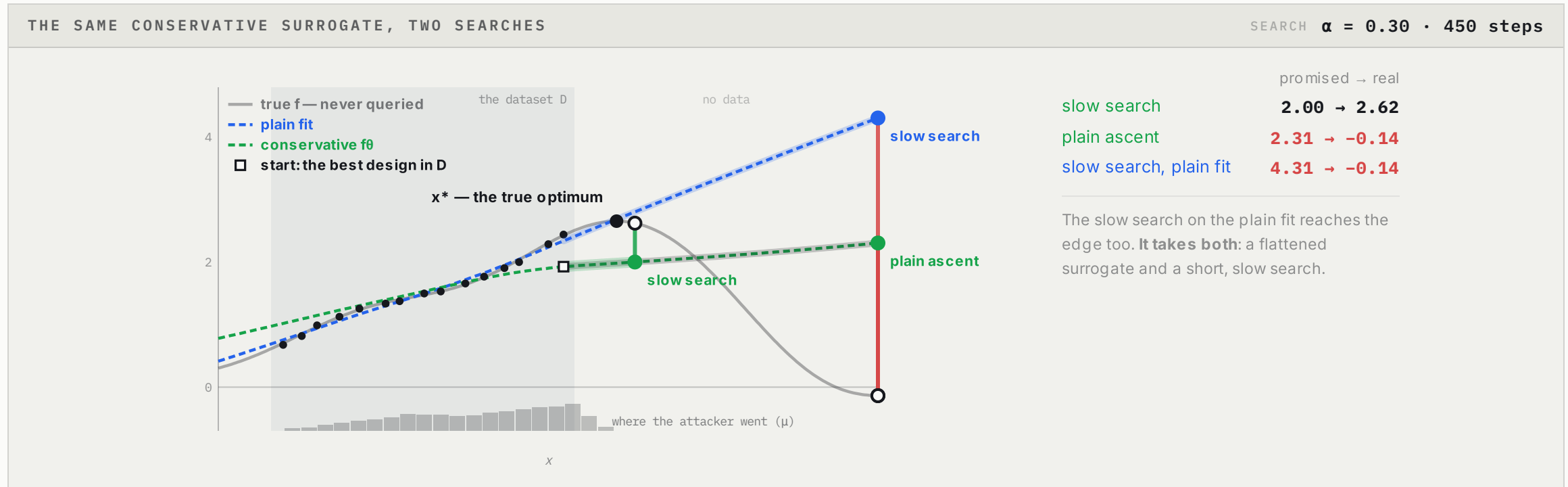
Naive ascent improves at first — near 2,800 after about seven steps in this figure — then deteriorates as further steps carry it further from the data.

COMs climbs slowly and stays high over the 450 steps — as in the toy, where a flattened surrogate and a slow search keep the design near the data for that budget.

The penalty does not make the search smarter. It makes later steps **stop being rewarded for leaving the data**.

Conservatism is learned where the attacker went

so the final search must not go far from there

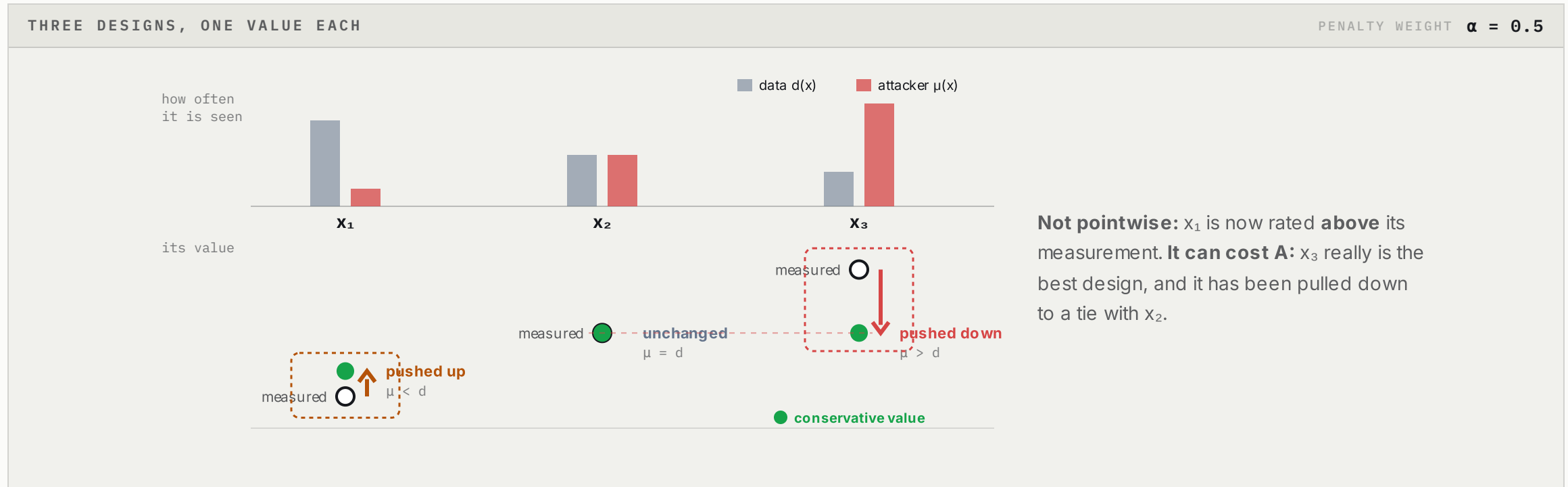


$\alpha = 0.3$: after 450 slow steps the search is next to the true best, still rising slowly. Press → : plain ascent on the same surrogate runs to the far edge, where **nothing pushed it down**. Press → again: the slow search on the steep plain fit.

COMs is the pair: a surrogate flattened where the attacker went, and a short, slow search on it.

What conservatism promises, and what it does not

a lower bound on average over the attacker's designs — not design by design



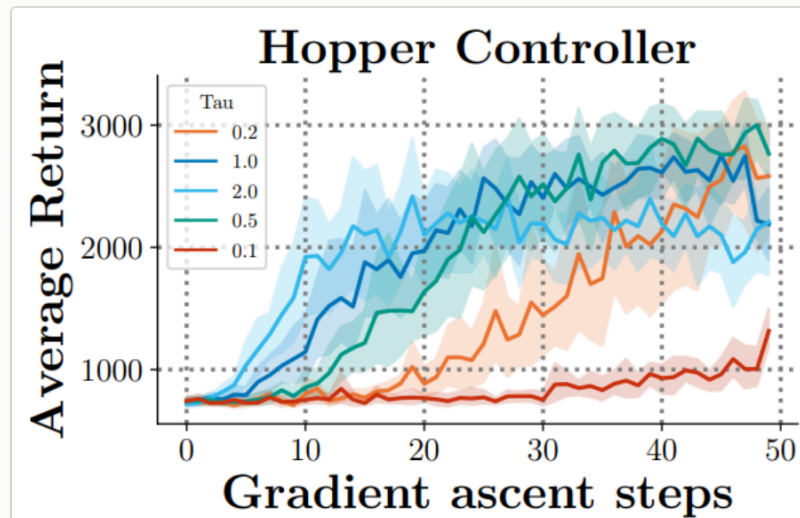
Three designs, one value each; grey is how often the data sees it, red how often the attacker does. Press $\rightarrow$ to apply $\hat{f} = y - \alpha(\mu/d - 1)$, and again for what follows.

Averaged over the attacker's designs the surrogate **sits below the truth** — the paper's Proposition 1; averaged over the data it is unbiased. Design by design it can sit above, and it can pull the best design down: the cost A .

Conservatism is a dial — make it a budget

choosing α is hard, so state the conservatism in the objective's own units

$$\theta^* = \arg \min_{\theta} \frac{1}{2} \mathbb{E}_{\mathcal{D}} [(f_{\theta}(x) - y)^2] \quad \text{s.t.} \quad \mathbb{E}_{x \sim \mu_{\theta}} [f_{\theta}(x)] - \mathbb{E}_{x \sim \mathcal{D}} [f_{\theta}(x)] \leq \tau$$



HopperController, five budgets. (Trabucco et al., 2021)

α 's right value depends on the scale of y ; τ is a **budget** on how far the surrogate may rate the attacker's designs above the data. It bounds a gap between two of the surrogate's *own* averages — not its error, which is never observed offline. With the objective normalised first, the paper uses $\tau = 0.05$ on every task, and α becomes the constraint's Lagrange multiplier.

$\tau = 0.1$ barely moves for forty steps — conservative enough to stay home. $\tau = 1$ and $\tau = 2$ climb fastest. Neither end is free.

Turning a penalty into a constraint with an interpretable budget is a move this course makes more than once — Lecture 1's trust region, and TRPO's bound on the policy step in Lecture 10.

The same disease, one rung up the course

an analogy Lecture 12 takes up

COMS – A CONSERVATIVE OBJECTIVE

$$\theta^* = \arg \min_{\theta} \left\{ \frac{1}{2} \mathbb{E}_D[(f_{\theta}(x) - y)^2] + \alpha (\mathbb{E}_{\mu}[f_{\theta}] - \mathbb{E}_D[f_{\theta}]) \right\}.$$

An optimiser exploits f_{θ} at **designs outside the data**.

CQL – A CONSERVATIVE VALUE

$$Q^* = \arg \min_Q \left\{ \frac{1}{2} \mathbb{E}_D[(Q - \mathcal{B}^{\pi} \hat{Q})^2] + \alpha \mathbb{E}_{s \sim D} \left[\log \sum_a e^{Q(s,a)} - \mathbb{E}_{a \sim \hat{\pi}_{\beta}(\cdot|s)}[Q(s,a)] \right] \right\}.$$

A policy exploits Q at **actions outside the data**.

Fit the data · push down what the optimiser would chase · hold up what the data contains. The problem and the shape of the objective are shared; the targets, the optimiser and the guarantees are not.

Check — what conservatism costs

In COMs, raise α far past its useful range. Which term of the decision bound grows, and why?

- ☐ A B — the surrogate becomes optimistic everywhere
- ☐ B The training error on $\mathcal{D}$ — term (iii) stops holding the data up
- ☐ C Neither — more conservatism is always safer
- ☒ D A — the surrogate under-rates good designs the attacker visits, so the search stops short of them

Conservatism **trades B for A** . Too little and the optimiser exploits the error; too much and the model refuses to rate anything it has not already seen, which is a very safe way to be useless. In the toy, $\alpha = 1.3$ gives $A = 1.41$ and a returned design worse than the best in $\mathcal{D}$. Writing the conservatism as a budget τ makes the trade visible; it does not remove it.

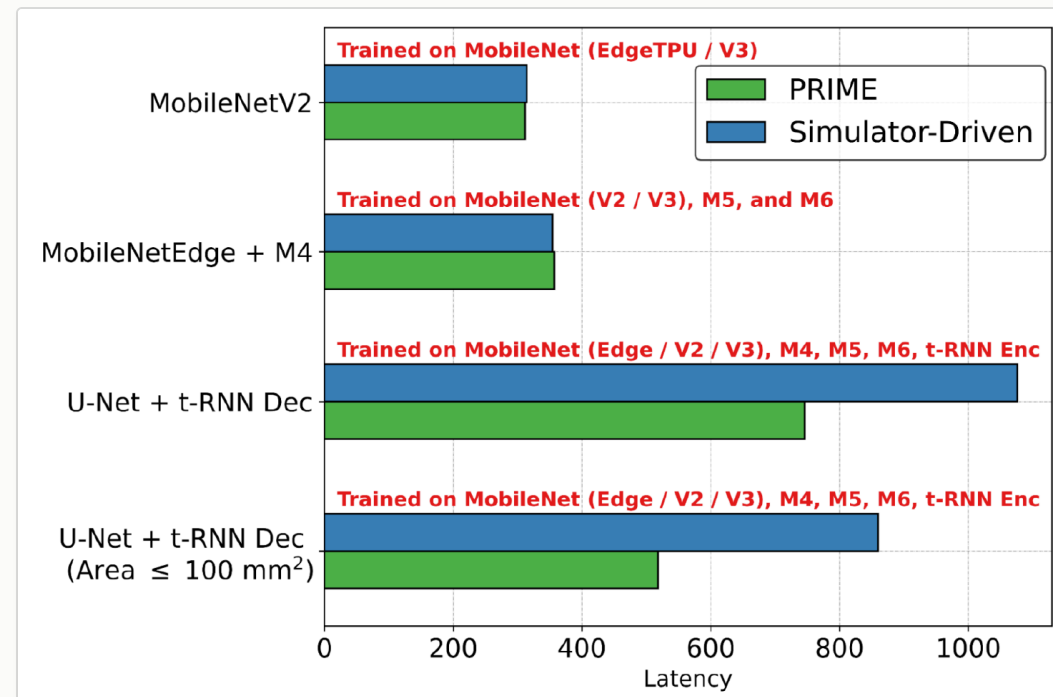
PRIME — a conservative surrogate for hardware design

Kumar, Yazdanbakhsh, Hashemi, Swersky & Levine · ICLR 2022

$$\theta^* = \arg \min_{\theta} \mathbb{E}_{\mathcal{D}} [(f_{\theta}(x) - y)^2] - \alpha \mathbb{E}_{x^- \sim \text{Opt}(f_{\theta})} [f_{\theta}(x^-)] - \beta \mathbb{E}_{x' \sim \mathcal{D}_{\text{infeasible}}} [f_{\theta}(x')]$$

Here f is a **latency**, which is minimised — so conservative means pushing the predicted latency *up* at the designs the optimiser finds, and at designs the log records as **infeasible**. Training over many target applications lets one model serve unseen ones.

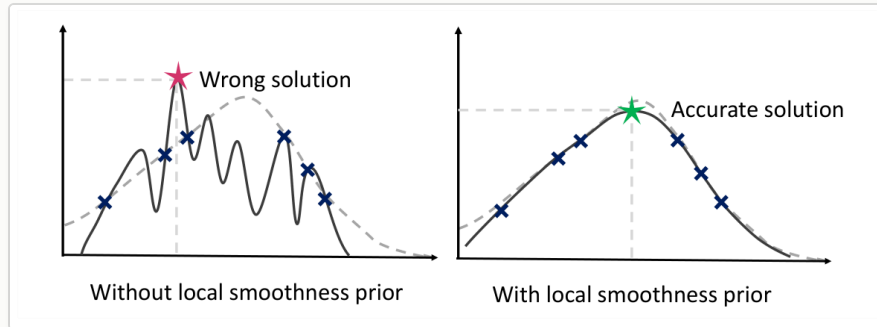
The search uses only a log of past simulations; the baseline calls the simulator as it searches.



Latency of the best accelerator found (lower is better). (Kumar et al., 2022)

RoMA — make the surrogate smooth where the optimiser stands

Yu, Ahn, Song & Shin · NeurIPS 2021



The jagged surrogate's tallest spike is a **wrong solution**; the smooth one peaks on the truth. (Yu et al., 2021, Fig. 2)

Its assumption: over-estimates come from a network that is **not smooth**, with spurious spikes between and beyond the data. So it changes the surrogate's *shape* near each candidate, not its *level*.

STAGE 1 — TRAIN IT SMOOTH

$$L(\theta) = \max_{\tilde{\theta} \in B(\theta)} \mathbb{E}_{(x,y) \sim D, \delta \sim \mathcal{N}(0,\sigma)} \left[(f(x + \delta; \tilde{\theta}) - y)^2 \right]$$

Noise on the inputs and the worst weights nearby: the fit must hold on a neighbourhood of each point, not at the point alone.

STAGE 2 — RE-SMOOTH AT EACH STEP

$$\theta_t = \arg \min_{\tilde{\theta} \in B(\theta)} \left\| \nabla_x f(x^{(t)}; \tilde{\theta}) \right\|_2 + \alpha [f(x^{(t)}; \tilde{\theta}) - f(x^{(t)}; \theta_{t-1})]^2$$

Flat at the candidate (the gradient term) without moving its prediction there (the second). Then one ascent step on $f(\cdot; \theta_t)$.

Adjacent to COMs rather than the same idea: COMs lowers the scores the optimiser would reach; RoMA removes the slopes that would carry it there.

Session 1— what conservatism fixed, and what it left

MITIGATED

B , the optimiser's exploit — on the designs the training attacker visits, with a dial (α) or a budget (τ) to set how hard, and a short, slow search that does not get far from there.

LEFT

- the protection is tied to **the region its attacker explored** — and to **the search's budget**;
- too much of it inflates A ;
- it never says **how much the model knows**, so it cannot tell a gap in the data from a region that is truly poor.

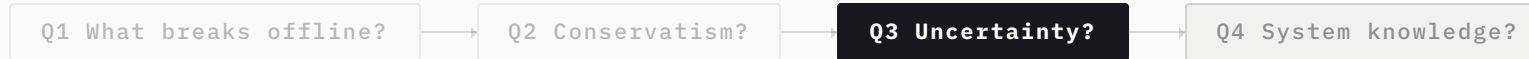
COMs controls the optimiser's optimism. Session 2 asks two questions it cannot answer: can we tell **which predictions are actually uncertain**, and can **knowledge of the system** make the model trustworthy to begin with?

— SESSION 2 • ACT 3

Uncertainty: decide with what the model does not know

Q3. Conservatism controls the optimiser's optimism. Can the surrogate instead tell us *where* it does not know — and can the decision use that?

Session 2 — where Session 1 left off



WHAT CONSERVATISM DID

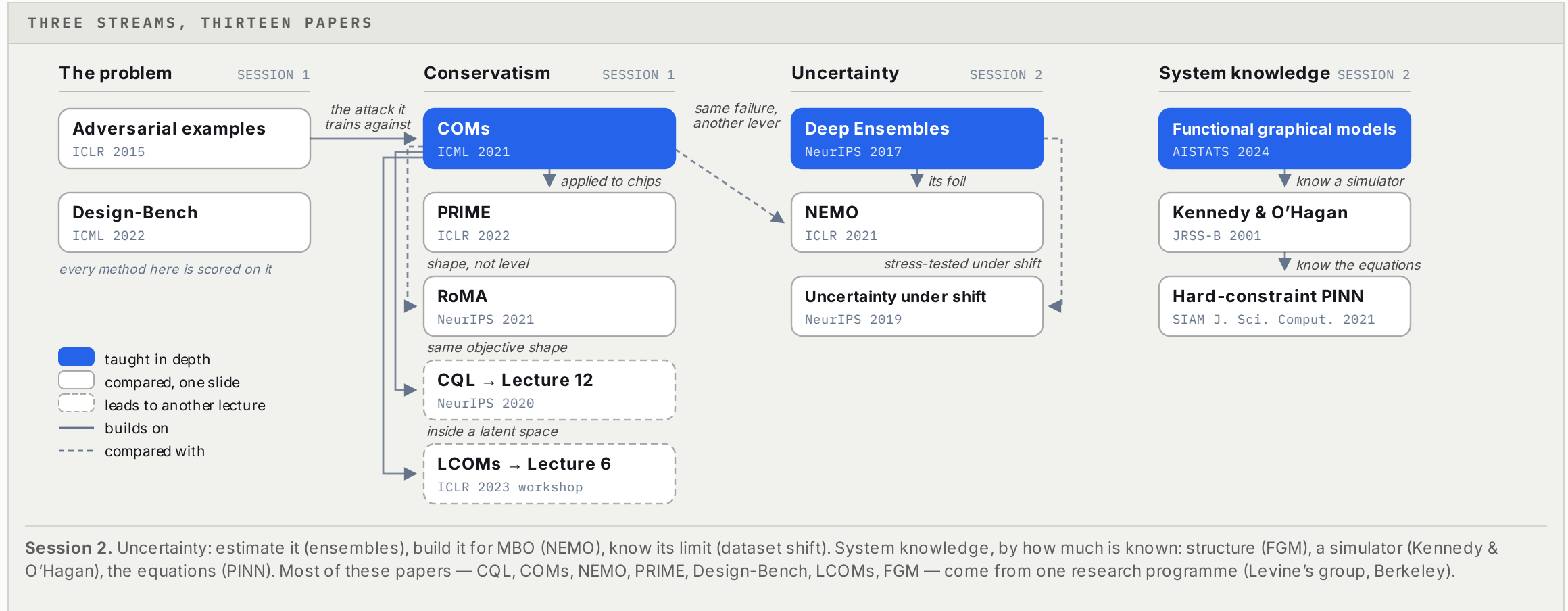
It lowered the scores the optimiser would reach — B shrank — on the designs its own attacker visited, and a slow search kept the final design there.

WHAT IT COULD NOT TELL US

How much the model knows. A penalty treats a gap in the data and a genuinely poor region alike: both are simply rated low.

Two answers today. **Uncertainty:** estimate what the surrogate does not know, and decide with it. **System knowledge:** give the surrogate grounds to extrapolate that the data alone cannot.

The papers of Session 2



The two right-hand columns are today's. Click a paper for what it does, why we read it and how it connects.

Uncertainty is Lecture 4's tool, with the sign turned

ONLINE — BAYESIAN OPTIMISATION (LECTURE 4)

$$x_{t+1} = \arg \max_x \mu(x) + \kappa \sigma(x)$$

Optimism. Go where the band is wide: if the model is wrong there, the next measurement says so and the band collapses.

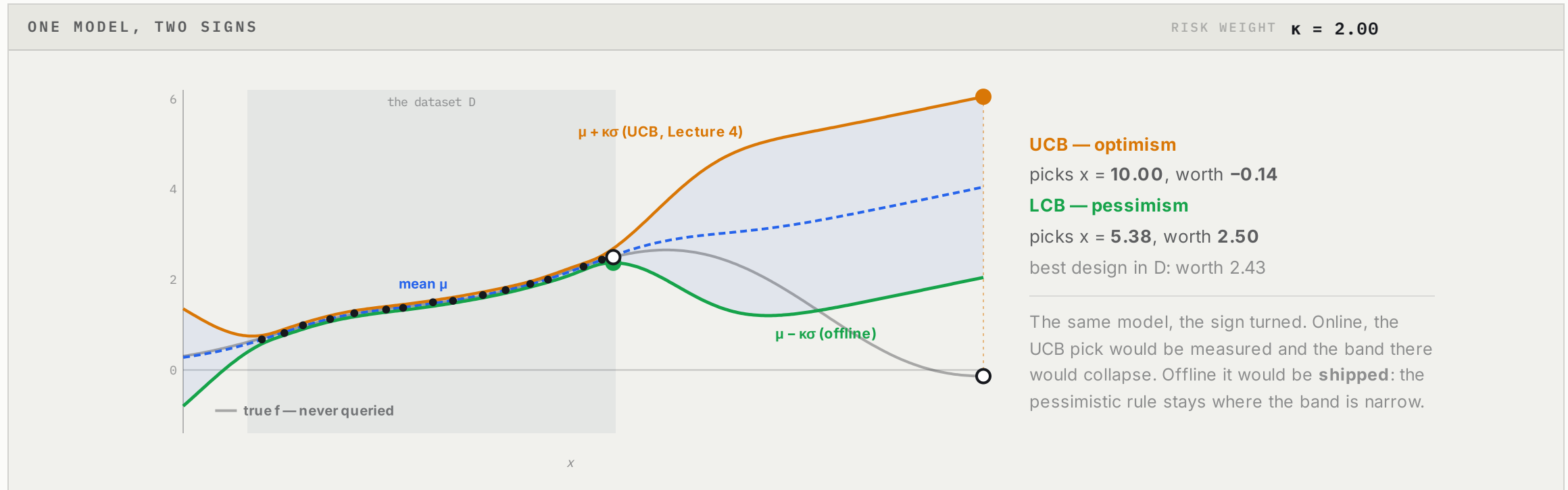
OFFLINE — THIS LECTURE

$$\hat{x} = \arg \max_x \mu(x) - \kappa \sigma(x)$$

Pessimism. Stay where the band is narrow: no measurement will ever correct a wrong guess, so a wide band is a risk, not an opportunity.

The same two numbers, μ and σ . What changed is whether the loop closes — whether uncertainty can still be **reduced** by looking.

One model, two signs



A model of the same fifteen points whose mean extrapolates optimistically — like Session 1's surrogate — but whose σ grows honestly away from the data. Optimism walks to the far edge, where the band is widest; pessimism with κ about 2 stays at the data.

Where σ comes from for a neural surrogate — Deep Ensembles

Lakshminarayanan, Pritzel & Blundell · NeurIPS 2017

A GP gives σ in closed form. A neural network does not. The simplest widely used answer: train several networks and read their disagreement.

$$\mu(x) = \frac{1}{M} \sum_{m=1}^M \hat{f}_m(x), \quad \sigma_{\text{ens}}^2(x) = \frac{1}{M} \sum_{m=1}^M (\hat{f}_m(x) - \mu(x))^2$$

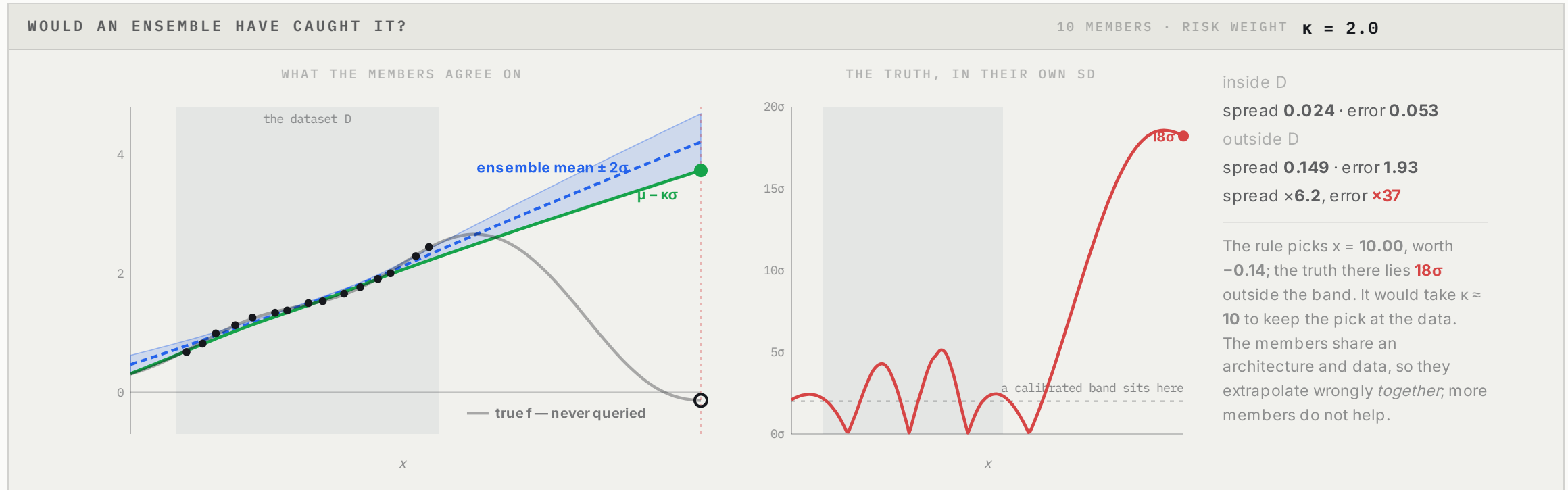
THE BET

Members trained from different random starts (or resampled data) agree where the data pins the function down, and **disagree** where it does not.

THE OFFLINE RULE

Plug the ensemble's μ and σ into $\mu - \kappa\sigma$. If the bet holds, the rule avoids the designs the members argue about.

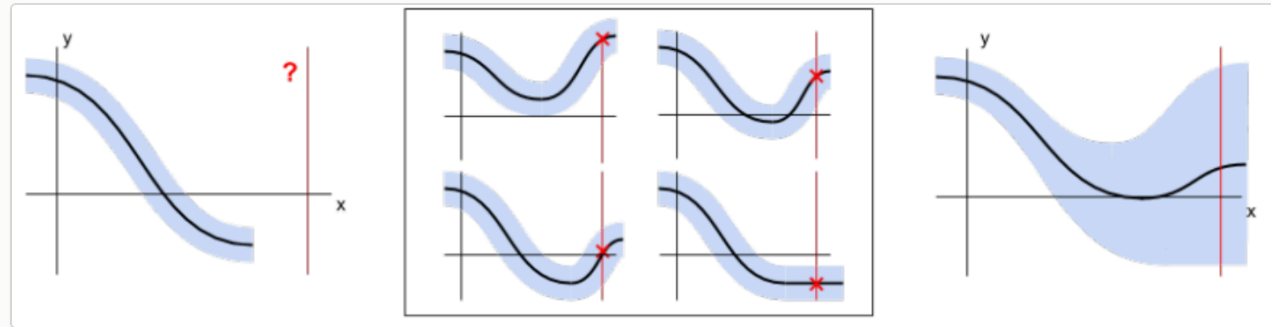
In our toy, the alarm is too quiet



Ten members, each fitted to a resample of the same fifteen points. Their spread does widen off the data — but at the far edge it is only 0.24 while their mean is wrong by 4.4. The rule $\mu - \kappa\sigma$ still walks to $x = 10$; it would take $\kappa \approx 10$ to stop it. The members share an architecture and data, so they **extrapolate wrongly together**.

NEMO — uncertainty as how easily the model could be talked into any answer

Fu & Levine · ICLR 2021



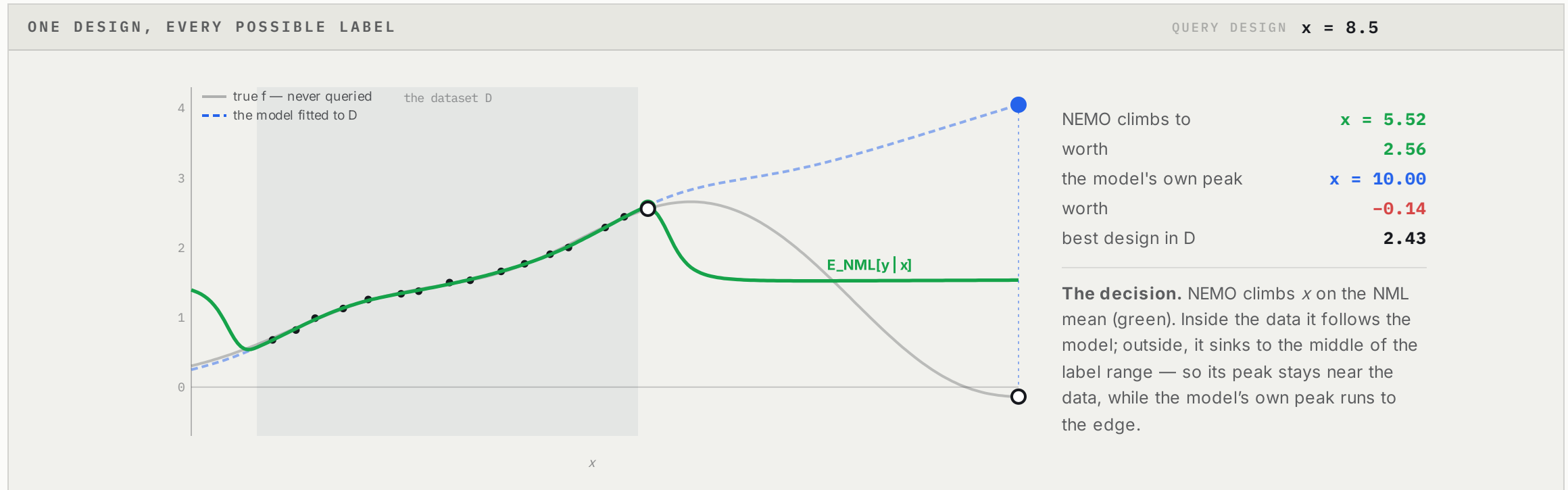
Left: we want $p(y | x)$ at a design far from the data. **Middle:** pretend each possible label y was observed there, and refit. **Right:** normalise across all those refits. (Fu & Levine, 2021, Fig. 1)

pretend this label was measured here and refit — how likely is it now?

$$p_{\text{NML}}(y | x) = \frac{\overbrace{p(y | x; \hat{\theta}_{\mathcal{D} \cup (x,y)})}}{\underbrace{\int p(y' | x; \hat{\theta}_{\mathcal{D} \cup (x,y')}) dy'}_{\text{the same score for every label — so they sum to one}}}$$

How easily could one extra point drag the model to any answer? Far from the data, easily: every label passes its own test, the scores come out equal, and the distribution is flat — *I don't know*. Inside the data, the neighbours hold the model: only labels near the trend pass, and it is narrow.

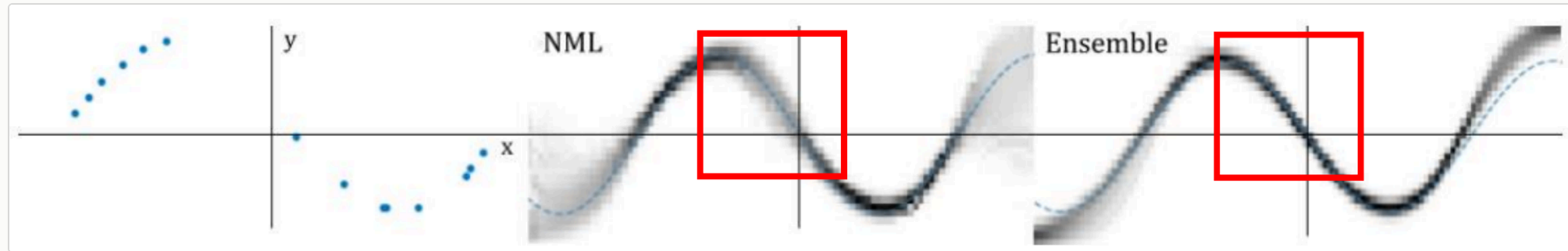
The NML formula, one label at a time



The same fifteen points. Press → four times: pretend and refit, score each label (the numerator), normalise (the denominator), decide. Drag the design x into the data and back out at any step.

The decision. NEMO climbs x on $\mathbb{E}_{p_{\text{NML}}}[y | x]$. Where the distribution is flat, that mean sits mid-range, so **optimistic extrapolation stops paying** and the climb stays near the data.

NEMO's own test — against the ensemble



The same data (left), modelled by NML (middle) and by a bootstrapped ensemble (right). In the boxed region, outside the data, NML is smeared; the ensemble is still a thin, confident line. (Fu & Levine, 2021)

The point NEMO makes is the one our toy just made: an uncertainty estimate is only useful offline if it **widens where the data runs out**.

How far can an uncertainty estimate be trusted?

Ovadia et al., "Can you trust your model's uncertainty?" · NeurIPS 2019

WHAT THEY FOUND

Across many methods, calibration — whether a 90 % band holds the truth 90 % of the time — gets worse as the test data drift away from the training data. Ensembles held up best, but they also degraded.

WHAT IT MEANS HERE

An optimiser does not drift gently away from the data: it **searches for the most extreme design it can find**. Offline MBO puts the uncertainty estimate exactly where such studies find it least reliable.

What uncertainty buys, and what it costs

ASSUMPTION ADDED

σ tracks the error: wide where the model is wrong, narrow where it is right.

WHAT IT SOLVES

The decision can see the difference between *I have checked this* and *I have not* — and avoid the second.

WHAT IT LEAVES

σ is itself a model output. Off the data it can be as wrong as μ , and in the same direction.

Check — reading an ensemble's agreement

At a design far from the data, all ten members of an ensemble agree to within 0.05. What can you conclude?

- ☐ A The prediction there is reliable
- ☐ B The design must lie inside the data distribution
- ☒ C Very little: members that share an architecture and data can extrapolate wrongly together
- ☐ D The true value is within 0.1 of the mean

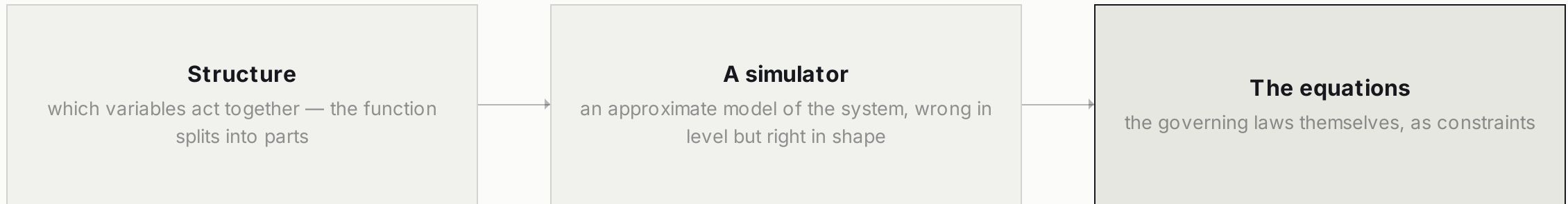
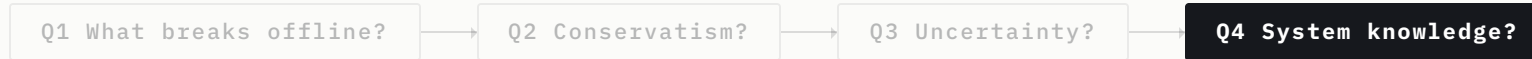
Agreement is evidence only where the members could have disagreed. Off the data, members built the same way from the same points tend to extrapolate the same way — our toy's ensemble agreed to 0.24 where it was wrong by 4.4. Uncertainty estimates also degrade under shift for every method studied (Ovadia et al., 2019), and an optimiser seeks out exactly the most shifted designs.

— SESSION 2 • ACT 4

System knowledge: give extrapolation grounds

Q4. The data cannot say what happens where there is no data. What we know about the system sometimes can.

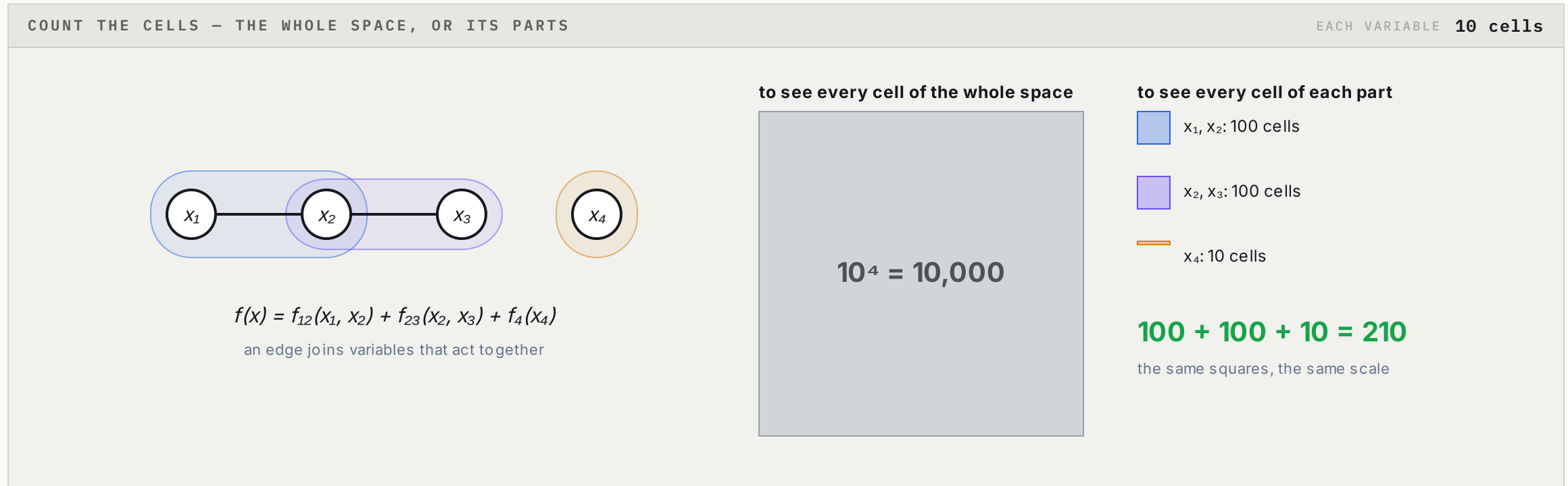
Knowledge, ordered by how much of it you have



Each step down adds information the dataset does not contain — and each is an assumption. Right knowledge makes extrapolation trustworthy; **wrong knowledge biases every prediction.**

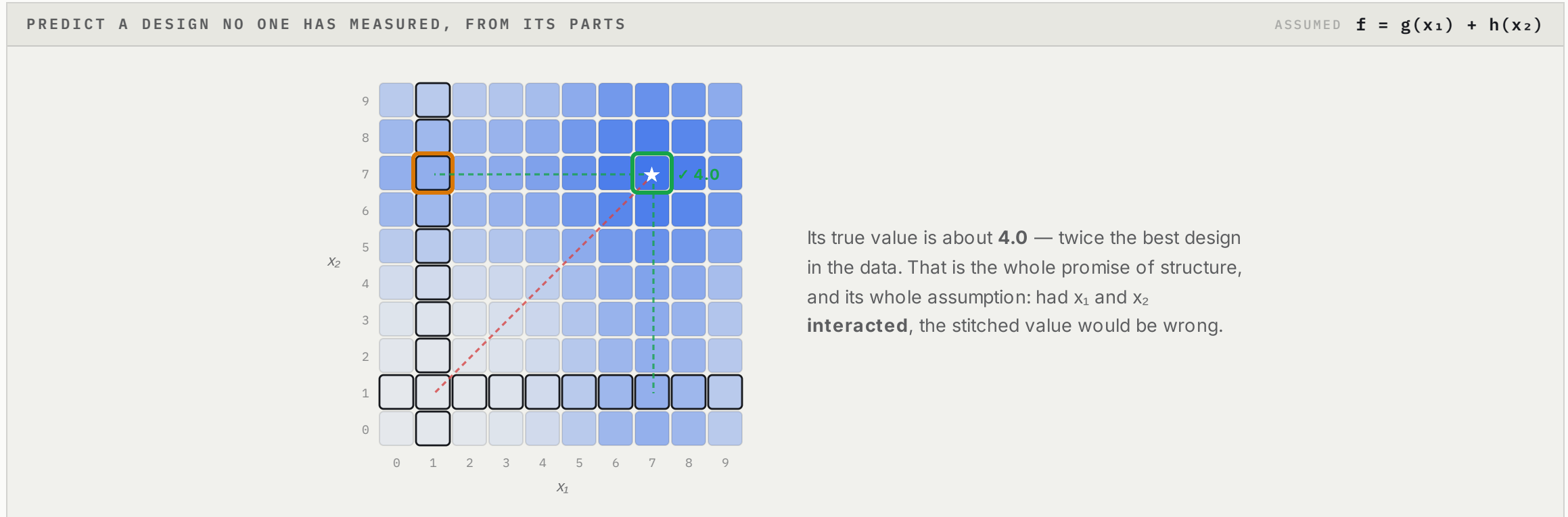
Structure — count what it takes to see the space

functional graphical models · Grudzien Kuba, Uehara, Abbeel & Levine, AISTATS 2024



Cut each variable into ten cells. To see every cell of the whole space takes 10^4 designs; if the function splits into parts, it is enough to see every cell of each part — press →. A count of coverage, not the paper's theorem.

Structure — a design better than any in the data

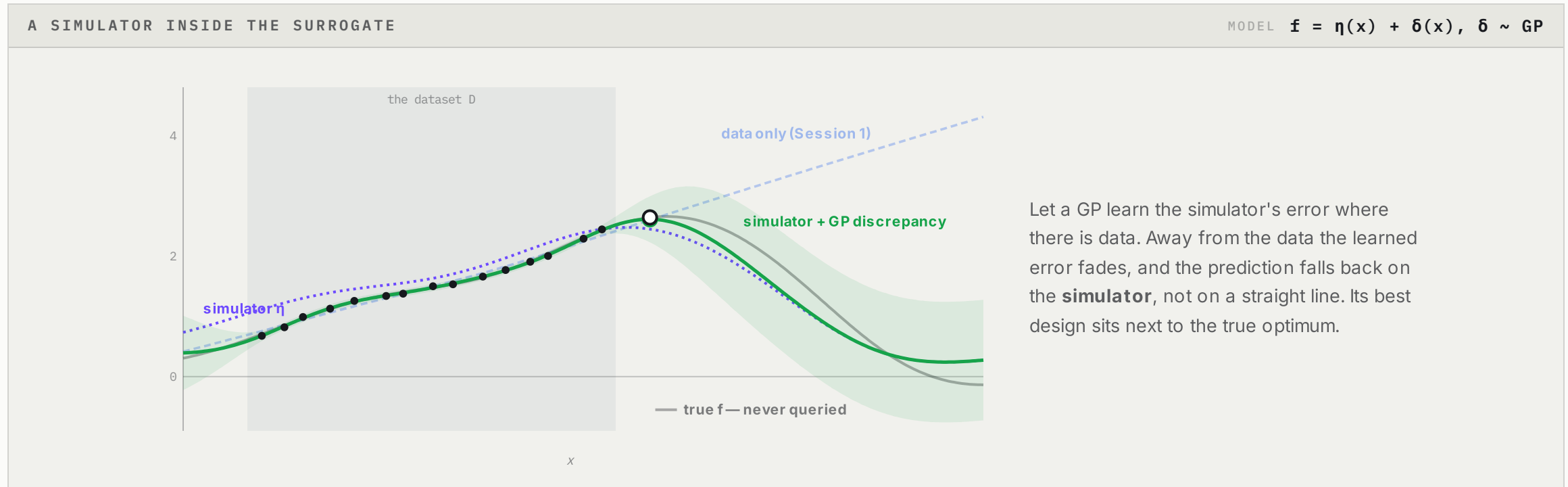


Press → to stitch, and again to check.

FGM's question is exactly slide 7's: can offline optimisation beat the best design in the data? With the right structure, yes — by combining parts that were each seen, **in a combination that never was**.

A simulator — Kennedy & O'Hagan

Bayesian calibration of computer models · JRSS-B 2001

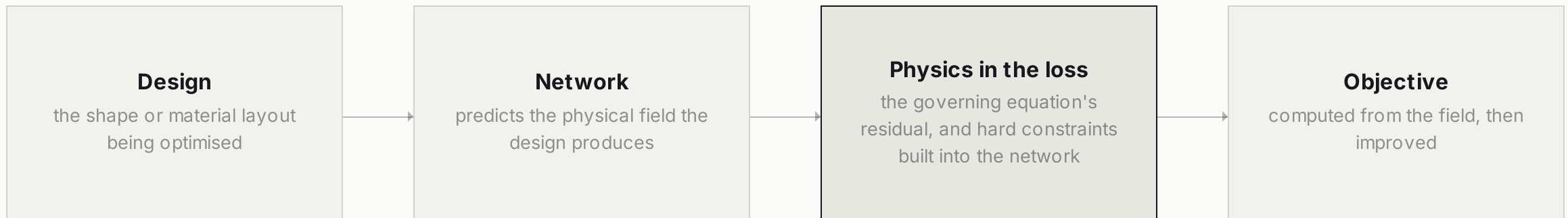


The same fifteen points and the same hidden function as Session 1. Press → twice.

The prediction relaxes to the simulator, not to a straight line, because the GP's error term returns to its zero prior mean away from the data. That is an assumption, not a guarantee: **where the simulator is wrong and there is no data, so is the model.**

The equations — physics-informed surrogates

hPINN · Lu, Pestourie, Yao, Wang, Verdugo & Johnson · SIAM J. Sci. Comput. 2021



When the governing equation is known, extrapolation is no longer left to the data: the network is trained to satisfy the equation at points spread over the domain, with the design's constraints imposed explicitly. The cost is the strongest assumption of the three — **a known equation** — and the physics holds only as well as it is enforced.

What knowledge buys, and what it costs

ASSUMPTION ADDED

The structure, the simulator or the equations are **right** — beyond the data, where they cannot be checked.

WHAT IT SOLVES

Extrapolation gets grounds other than the data: parts combine, a simulator supplies the shape, physics rules out the impossible.

WHAT IT LEAVES

Knowledge is often partial or missing, and wrong knowledge does not announce itself — it is trusted most exactly where the data cannot correct it.

Check — when to trust a stitched design

An additive model $f(x_1, x_2) = g(x_1) + h(x_2)$ predicts that a combination never measured beats every design in the data. When should you trust it?

- ☐ A Always — the model fits the data closely
- ☒ B When x_1 and x_2 really do not interact, and each part was observed across the range used
- ☐ C Never — the design is out of distribution
- ☐ D When the dataset is large enough

The prediction rests on the structure, not on the fit: if x_1 and x_2 interact, a small training error can coexist with a badly wrong stitched value. With the right structure and coverage of each part, combining parts that were each seen is exactly how offline optimisation can beat the best design in the data.

— CLOSING

Closing

Three answers to one failure — and the problem they leave for Lecture 6.

Three answers, side by side

	CONSERVATISM	UNCERTAINTY	SYSTEM KNOWLEDGE
in depth	COMs	Deep Ensembles, NEMO	FGM
mechanism	lower the scores the optimiser would reach	estimate what the model does not know; decide with $\mu - \kappa\sigma$	constrain how the model may extrapolate
assumption	the attacker finds the tempting designs	σ tracks the error	the structure, simulator or physics is right
strength	curbs optimism where the search goes	separates <i>unchecked</i> from <i>checked</i>	can beat the best design in the data
weakness	too much inflates A ; tied to its search	σ fails where it matters most	wrong knowledge biases everything

Not rivals: an uncertainty penalty is itself conservative, and a structured model can be trained conservatively. Each controls B under a different assumption.

What all three leave — and what Lecture 6 takes up

PERFORMANCE

Is the design good? — *the objective*

RELIABILITY

Can we trust the predicted score? — [this lecture](#)

VALIDITY

Is the design possible at all — a molecule, a foldable protein? — [Lecture 6](#)

OOD and invalid are different problems: a design outside the data can be valid and excellent, and one next to the data can break a constraint. This lecture gave predictions grounds for trust, each under a stated assumption; it did not make the search space valid or give a batch of different good candidates. Lecture 6 learns [the distribution of plausible designs](#) and lets a performance model in at four places — and every tool of this lecture can still be used there.

Where we are — a forward model, then a search

THE MODEL		THE DECISION
Surrogate (<i>Lec 5</i> ✓)	forward: $f_{\theta}(x) \approx f(x)$	search f_{θ} — conservatively, with uncertainty, or with knowledge
Generative (<i>Lec 6</i>)	a distribution of plausible designs, plus a performance model	search its coordinates, refit it, condition it, or steer its sampler

Hold the shape of this lecture: a forward model searched carefully. Lecture 6 keeps a performance model in the loop and adds a generator that decides **where to look** — and this lecture's failure reappears there wherever the performance model steers the generator. Lecture 12 meets it again, with a policy exploiting a Q -function.

Offline, every number off the data is a guess —
decide accordingly.

Lower the guess where the search goes, measure how unsure it is, or give it grounds from what you know
about the system.

Questions?

Two things to carry out of here. **The optimiser is an adversary of its own model** — it will find the errors.

And **trust needs grounds** — a penalty, an honest uncertainty, or knowledge of the system.

— APPENDIX

Backup slides

Complete statements, kept out of the narrative.

Reading guide — Session 1 papers: the problem and conservatism

PAPER	WHY WE READ IT	HOW IT CONNECTS
Adversarial examples · Goodfellow et al., ICLR 2015	the mechanism: gradient search finds where a model is wrong	COMs generates this attack during training and trains against it
Design-Bench · Trabucco et al., ICML 2022	defines offline evaluation; every method here is scored on it	we use its 2021 task versions
COMs · Trabucco et al., ICML 2021 — <i>in depth</i>	the clearest conservative surrogate	one-step twin of CQL; PRIME and LCOMs build on it
PRIME · Kumar et al., ICLR 2022	conservatism in real chip design, from logged simulations	builds on COMs
RoMA · Yu et al., NeurIPS 2021	another lever: shape, not level	compared with COMs
CQL · Kumar et al., NeurIPS 2020	the same disease, with a policy as the optimiser	Lecture 12
LCOMs · Qi et al., ICLR 2023 workshop	a conservative surrogate searched in a learned space	Lecture 6

Reading guide, continued — Session 2 papers: uncertainty and system knowledge

PAPER	WHY WE READ IT	HOW IT CONNECTS
Deep Ensembles · Lakshminarayanan et al., NeurIPS 2017 — <i>in depth</i>	the standard σ behind $\mu - \kappa\sigma$	Lecture 4's UCB, sign turned
NEMO · Fu & Levine, ICLR 2021	uncertainty built for offline MBO	same group and year as COMs
Uncertainty under shift · Ovadia et al., NeurIPS 2019	calibration degrades with shift, even for ensembles	stress-tests the uncertainty stream
Functional graphical models · Grudzien Kuba et al., AISTATS 2024 — <i>in depth</i>	when structure lets us beat the best design in the data	Lecture 3's graphs
Kennedy & O'Hagan · JRSS-B 2001	a simulator inside a surrogate	Lecture 4's GP as the error term
Hard-constraint PINN · Lu et al., SIAM J. Sci. Comput. 2021	the governing equations themselves	needs a known PDE

Backup 1— offline MBO against Bayesian optimisation

The same goal, $\arg \max_x f(x)$, under different access — and the difference dictates the method.

	BAYESIAN OPTIMISATION (LEC 4)	OFFLINE MBO (LEC 5)
data	<i>active</i> — query f each round	<i>fixed</i> — a static dataset D
a mistake is	corrected by the next query	uncorrectable — it is returned as the answer
uncertainty drives	<i>where to sample</i>	<i>where not to trust</i>
the core risk is	slow convergence	overestimation → an invalid design
the design principle	exploration	conservatism

Both build a surrogate; both maximise something over it. What differs is whether the loop closes. When it does, optimism is self-correcting and therefore cheap — an over-rated point gets queried, found wanting, and the posterior repairs itself. When it does not, optimism is a one-way door.

Backup 2 — NEMO, made tractable

The estimator. p_{NML} is the minimax solution of $\arg \min_h \max_{y'} (\log p(y' | x; \hat{\theta}_{D \cup (x, y')}) - \log h(y' | x))$, where $\hat{\theta}_{D \cup (x, y)} = \arg \max_{\theta} \frac{1}{N+1} \sum_{(x, y) \in D \cup (x, y)} \log p(y | x, \theta)$.

Problem. The denominator requires training an MLE *for every possible* y and then integrating over them — impossible twice over for a deep network.

Fix 1 — quantise. Floor each y into one of K bins, so the integral becomes a sum: $\int_y p(y | x, \hat{\theta}_{D \cup (x, y)}) dy \approx B \sum_{k=1}^K p(\lfloor y_k \rfloor | x, \hat{\theta}_{D \cup (x, \lfloor y_k \rfloor)})$.

Fix 2 — amortise. Keep K models and update them incrementally *while* optimising x , rather than retraining from scratch at each iterate:

```
for t = 1 ... T:
  for k = 1 ... K:  D' ← D ∪ (x_t, ⌊y_k⌋);  θ^k ← θ^k + α_θ ∇ LogLik(θ^k, D')
  p̂_NML(y | x_t) ∝ p(y | x_t, θ^y) / Σ_k p(⌊y_k⌋ | x_t, θ^k)
  x_{t+1} ← x_t + α_x ∇_x E_{y ~ p̂_NML(y|x)}[ g(y) ]
```

Quantisation flattens the landscape and kills the gradient, so NEMO's head outputs one minus the CDF of a *logistic* distribution sampled at intervals of $1/K$ and takes the mean; gradients then flow through the logistic mean $\mu(x)$, and Proposition 4.1 guarantees $\langle \nabla_x \mu(x), \nabla_x y_{\text{mean}}(x) \rangle \geq 0$ — the smooth surrogate gradient never points against the one we want.

Backup — one COMs loss calculation

freeze the sampled candidates while inspecting the update

One measured design has target $y = 4$ and prediction $u = 3.5$. One adversarial design has prediction $v = 8$. Use $\alpha = 0.2$.

$$L = \frac{1}{2}(u - 4)^2 + 0.2(v - u) = 0.125 + 0.9 = 1.025.$$

AT THE MEASURED DESIGN

$$\frac{\partial L}{\partial u} = (u - 4) - 0.2 = -0.7.$$

Gradient descent increases u , improving the fit and the relative data score.

AT THE ADVERSARIAL DESIGN

$$\frac{\partial L}{\partial v} = 0.2.$$

Gradient descent decreases v , reducing the unsupported high prediction.

Fit measured targets and shrink the score gap to tempting candidates. In a network the two updates share parameters, so each moves the other's prediction too.

Backup — the tabular COMs solution

one free value per design, for intuition

Take a finite set of designs, one free value $\hat{f}(x)$ for each, data frequencies $d(x)$ and attacker frequencies $\mu(x)$. The loss is a sum over designs, so each $\hat{f}(x)$ is solved on its own:

$$L = \frac{1}{2} \sum_x d(x) (\hat{f}(x) - y(x))^2 + \alpha \left(\sum_x \mu(x) \hat{f}(x) - \sum_x d(x) \hat{f}(x) \right)$$

Setting $\partial L / \partial \hat{f}(x) = d(x)(\hat{f}(x) - y(x)) + \alpha(\mu(x) - d(x)) = 0$:

$$\hat{f}(x) = y(x) - \alpha \left(\frac{\mu(x)}{d(x)} - 1 \right)$$

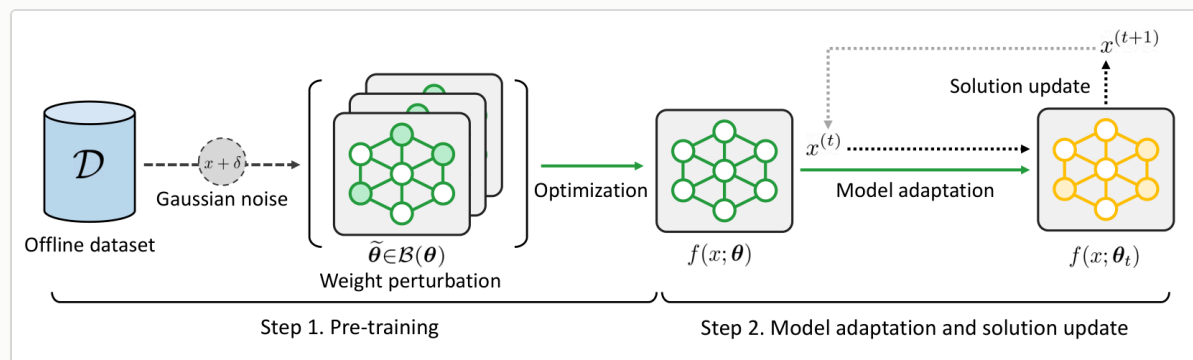
Average the error under each distribution, using $\sum_x \mu(x) = \sum_x d(x) = 1$:

$$\mathbb{E}_\mu[\hat{f} - y] = -\alpha \left(\sum_x \frac{\mu(x)^2}{d(x)} - 1 \right) = -\alpha \sum_x \frac{(\mu(x) - d(x))^2}{d(x)} \leq 0, \quad \mathbb{E}_d[\hat{f} - y] = 0.$$

A lower bound **on average under the attacker**, unbiased under the data — not design by design, and only inside the data, where $d(x) > 0$.

Backup — RoMA's two stages

Yu, Ahn, Song & Shin · NeurIPS 2021



Step 1 pre-trains under input noise and weight perturbation, but only smooths where the data is; so Step 2 re-adapts the model (yellow) at every candidate $x^{(t)}$ before the solution update. (Yu et al., 2021, Fig. 1)

STAGE 1 — TRAIN IT SMOOTH

$$L(\theta) = \max_{\tilde{\theta} \in B(\theta)} \mathbb{E}_{(x,y) \sim D, \delta \sim \mathcal{N}(0,\sigma)} \left[(f(x + \delta; \tilde{\theta}) - y)^2 \right]$$

Noise on the inputs, worst-case perturbations of the weights, $B(\theta) = \{\tilde{\theta} : \|\theta_l - \tilde{\theta}_l\|_F \leq \epsilon \|\theta_l\|_F\}$; the inner maximisation by projected gradient ascent.

STAGE 2 — RE-SMOOTH AT EACH STEP

$$\theta_t = \arg \min_{\tilde{\theta} \in B(\theta)} \left\{ \|\nabla_x f(x^{(t)}; \tilde{\theta})\|_2 + \alpha [f(x^{(t)}; \tilde{\theta}) - f(x^{(t)}; \theta_{t-1})]^2 \right\}$$

Flatten the model at the current candidate, without moving its prediction there far.

What the benchmark says

source-reported best-of-batch task scores; Avg uses the source's normalised aggregate

METHOD	GFP	MOLECULE	SUPERCOND.	HOPPER	ANT	DKITTY	AVG
<i>Dataset max</i>	3.152	6.558	73.90	1361.6	108.5	215.9	<i>1.000</i>
Gradient ascent	2.894	6.636	89.64	1050.8	399.9	390.7	1.237
MINs	3.315	6.508	80.23	746.1	388.5	352.9	1.304
CbAS	3.408	6.301	72.17	547.1	393.0	396.1	1.324
COMs	3.305	6.876	110.0	2395.7	378.8	341.4	1.589
NEMO	3.359	6.682	127.0	2130.1	393.7	431.6	1.687
RoMA	3.357	6.890	103.9	2466.5	468.5	384.3	1.705

Naive gradient ascent is not useless — it has the lowest aggregate score among the six methods shown, and on HopperController it returns less than the best trajectory already in the dataset. Every method that beats it does so by **adding a constraint on what the surrogate is allowed to believe**, not by searching harder.

LCOMs — a conservative surrogate searched in a learned space

Latent Conservative Objective Models · Qi et al., ICLR 2023 workshop on ML for materials

Chemical space is not a vector space, so a **CD-VAE encoder** $\phi(x, c)$ maps a crystal into one, and a conservative surrogate of the lattice energy is optimised **inside that latent space**, then decoded:

$$\theta^* = \arg \min_{\theta} \mathbb{E}_{(x, c) \sim \mathcal{D}} \left[(E_{\theta}(\phi(x, c), c) - E(x, c))^2 \right] - \alpha \mathbb{E}_{x^- \sim \text{Opt}(E_{\theta})} \left[E_{\theta}(\phi(x^-, c), c) \right]$$

Mean energy improvement 2.25 against supervised learning's 1.10. A generative model supplies the coordinates; a conservative surrogate does the searching — the bridge Lecture 6 crosses.

Five ways to optimise a black box, and what every one of them needs

what the source lecture spends fifteen slides establishing

METHOD	WHAT IT DOES	WHAT IT COSTS
Gradient ascent on a proxy	fit f_θ to the data so far, step $x_{t+1} = x_t + \eta \nabla_x f_\theta(x_t)$, evaluate	one query per step
Genetic algorithms	population, truncation selection of the top E , crossover, mutation	N queries per generation
CMA-ES	sample from an adaptive Gaussian $\mathcal{N}(\mathbf{m}_t, \sigma_t^2 C_t)$; update mean, covariance and step size	n queries per generation
Bayesian optimisation (Lec 4)	GP posterior, then $x_{t+1} = \arg \max_x A_t(x)$	one query per round
Policy gradient	learn $\pi_\theta(x)$ by $\nabla_\theta \mathbb{E}_{x \sim \pi_\theta} [f(x)] = \mathbb{E}[\nabla_\theta \log \pi_\theta(x) f(x)]$	n queries per round

Every one of the five is a loop, and every loop closes through [an evaluation of the real \$f\$](#) .