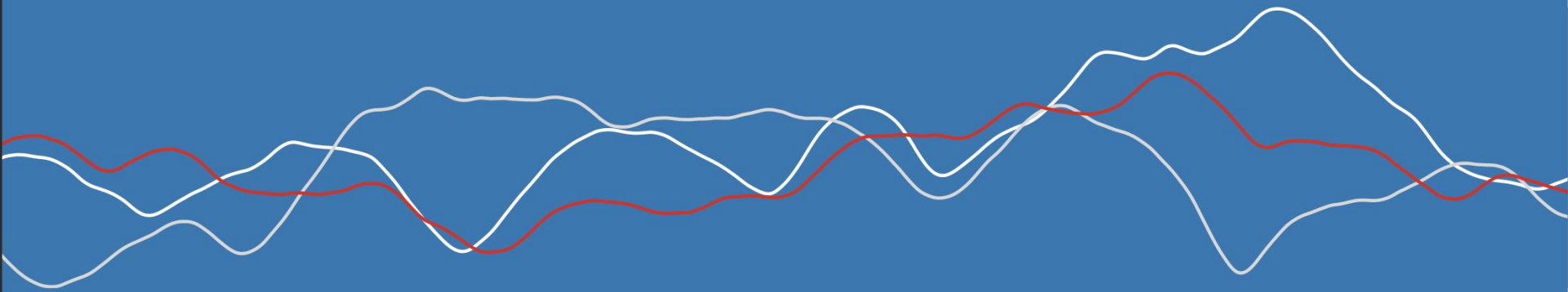
# Bayesian optimization

Roman Garnett

PN Summer School, Tübingen, March 2023



# Motivation

# Motivation

Numerous design problems spanning the sciences, engineering, ML, and beyond suffer from shared prototypical challenges:

- an *enormous* design space that is
- *inherently slow* to explore.

We are left “searching for needles in a haystack.”

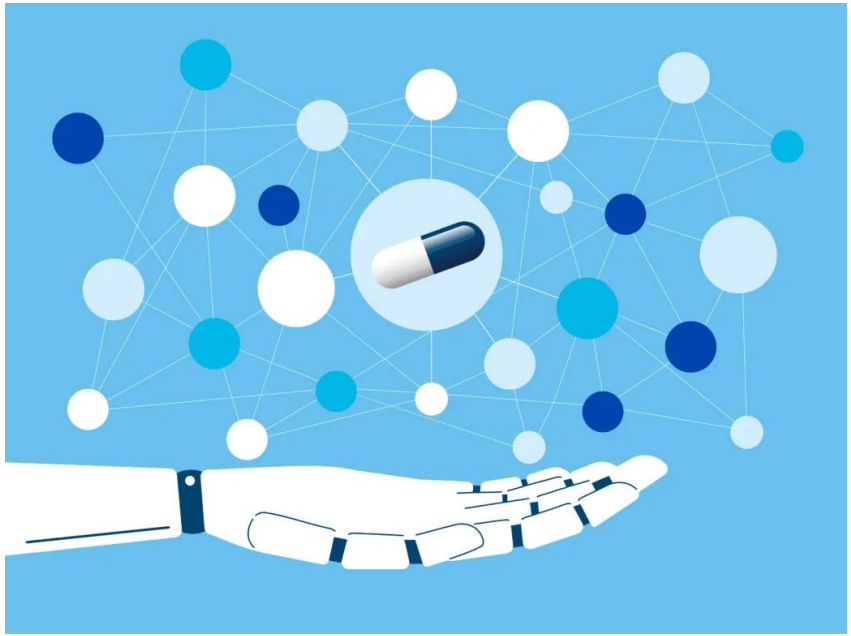
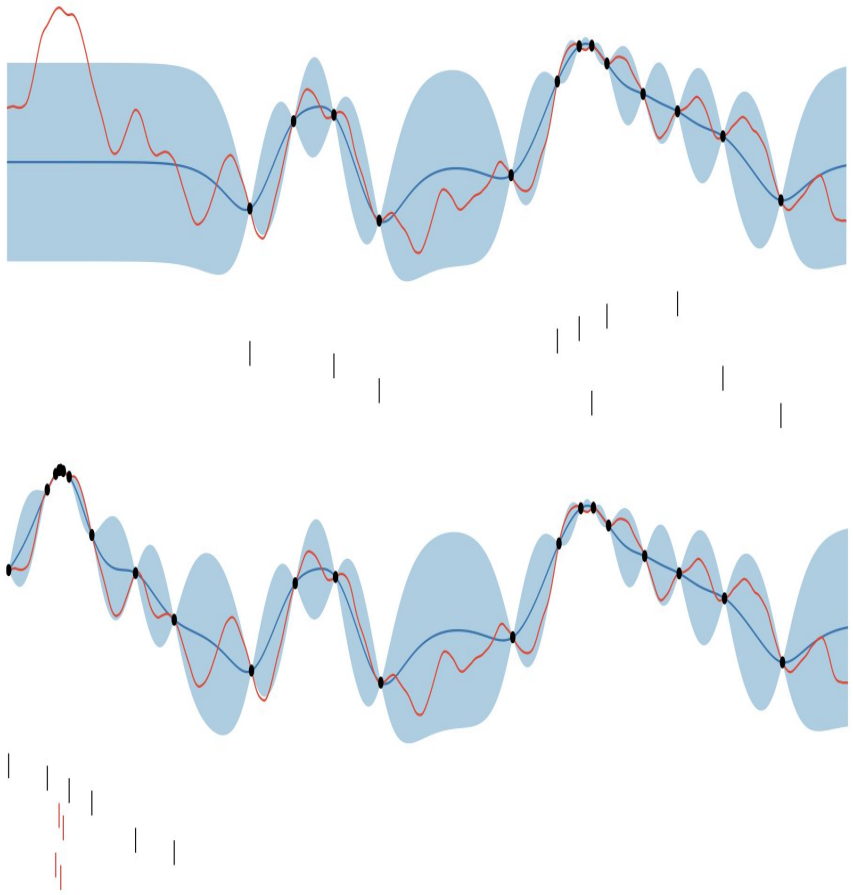


image: shutterstock

Experimental design got its start in *crop breeding*, where you may only get to run one experiment a year!

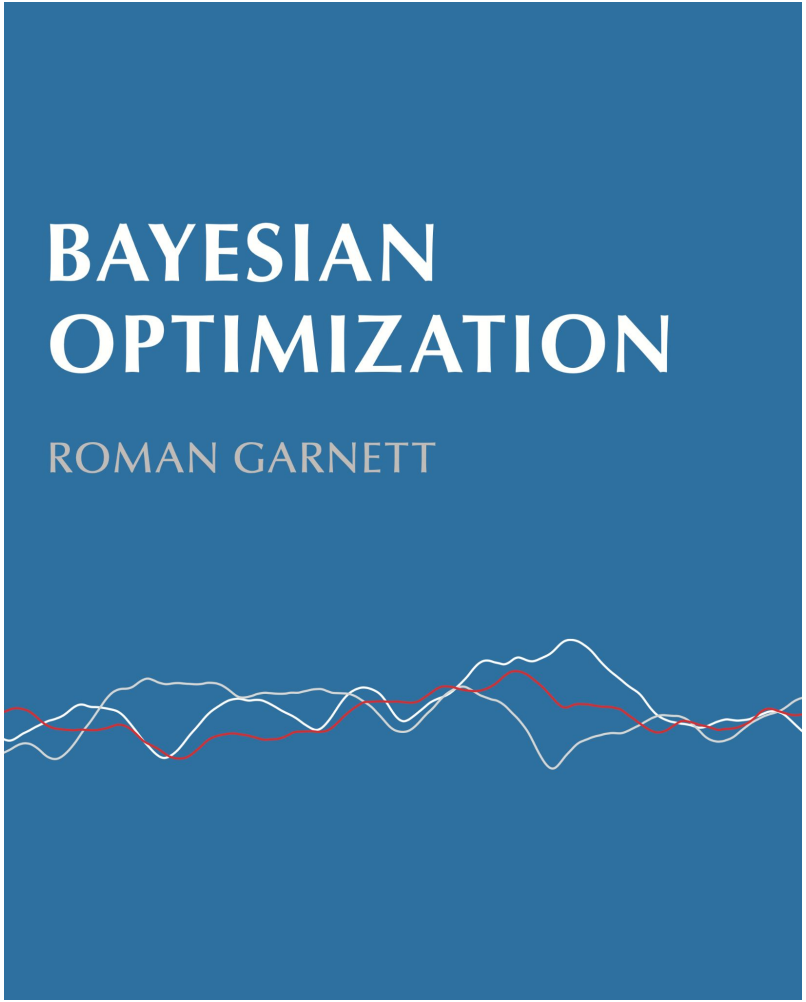
image: Jean Wimmerlin/Unsplash

The Bayesian approach to experimental design has found a niche in accelerating discovery and design in such situations.



Bayesian optimization is one popular realization of this paradigm that has made inroads across the sciences, engineering, and beyond.

bayesoptbook.com



# Experiments and goals

# Experimental design

Central to any experimental design problem is a mapping from some design space to properties of interest:

design space → properties of interest

**We can control this...** **...and measure this.**

# Experimental design

Central to any experimental design problem is a mapping from some design space to properties of interest:

hyperparameters → validation error

# Experimental design

Central to any experimental design problem is a mapping from some design space to properties of interest:

composition
processing parameters → material properties
etc. (optical, thermal, electronic, mechanical, etc.)

# Experimental goals

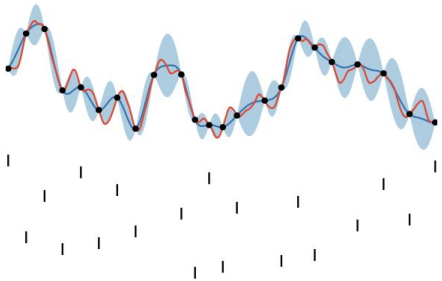
In order to rationally design experiments, we need to have a *goal* in mind. Some typical high-level goals:

- *learning* this mapping, e.g. to make confident predictions
- *optimizing* properties of interest (*design*)
- *finding* novel configurations with favorable properties of interest (*discovery)*

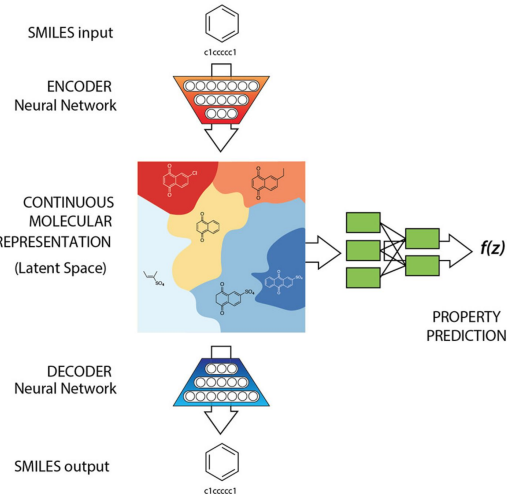
(Sometimes we might wish to accomplish a combination of the above!)

*Bayesian* experimental design can address all of these goals and more:

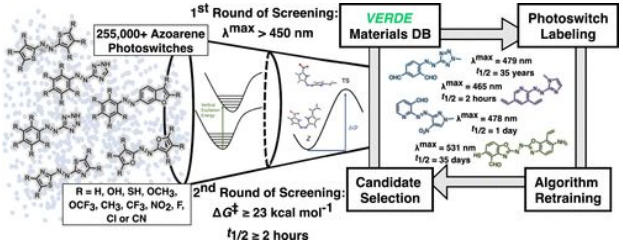
pure learning



optimization/design




discovery




**active learning*** **Bayesian optimization** **active search**

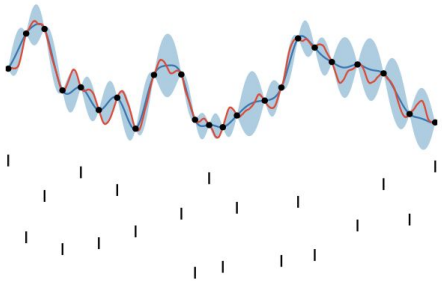
*this phrase is a bit overloaded and is sometimes used to refer to any realization of Bayesian experimental design

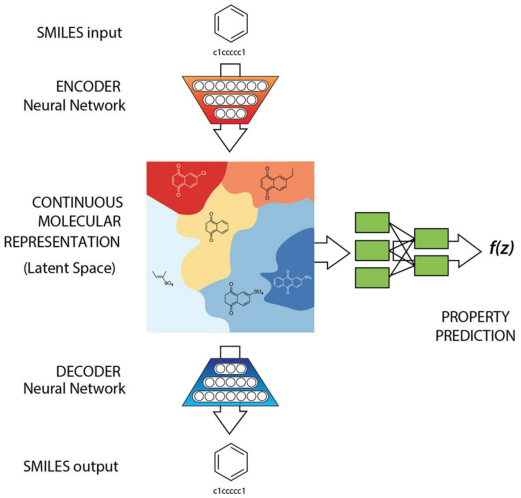
*Bayesian* experimental design can address all of these goals and more:

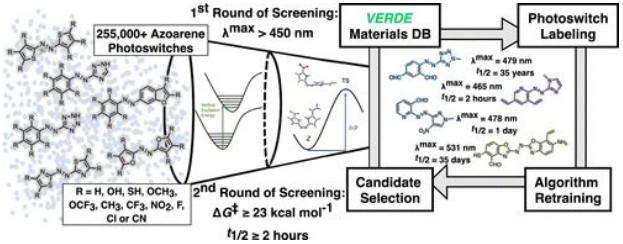
pure learning

optimization/design

discovery






**active learning**

**Bayesian optimization**

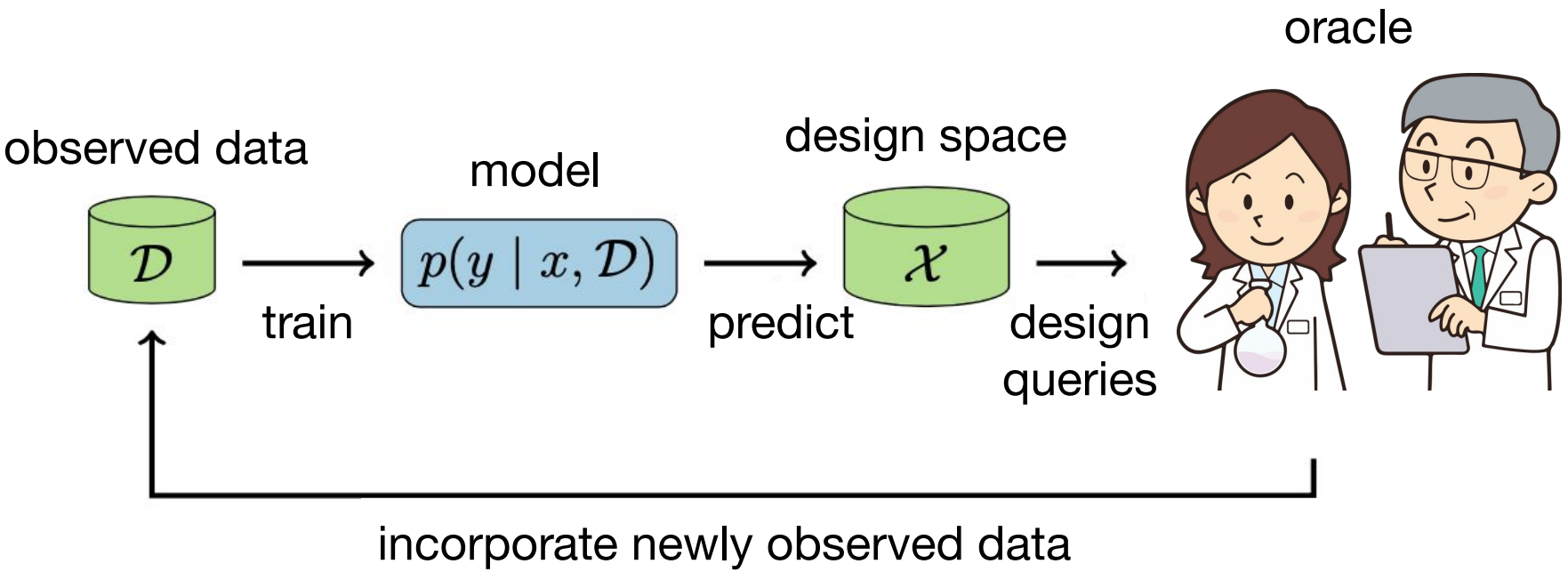
**active search**

images: Gómez-Bombarelli, et al. (2018), Mukadum, et al. (2021)

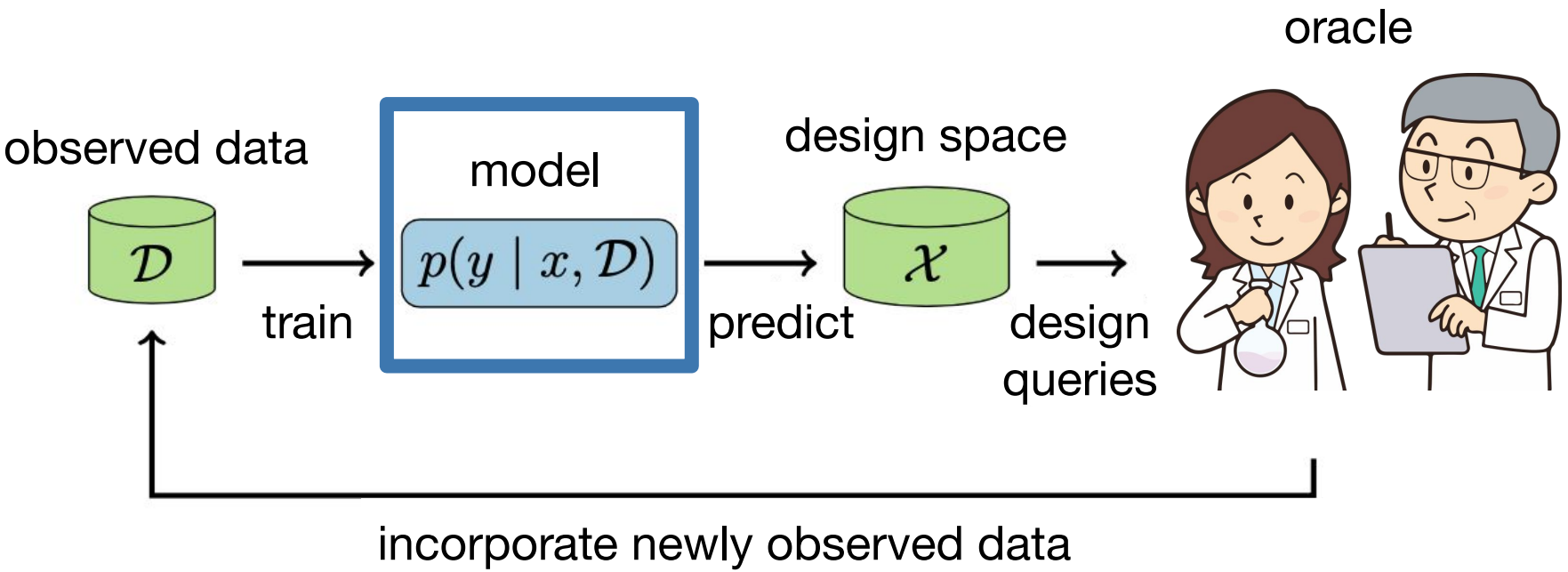
# Main ideas

[BayesOpt Book Chapter 1]

# Bayesian experimental design

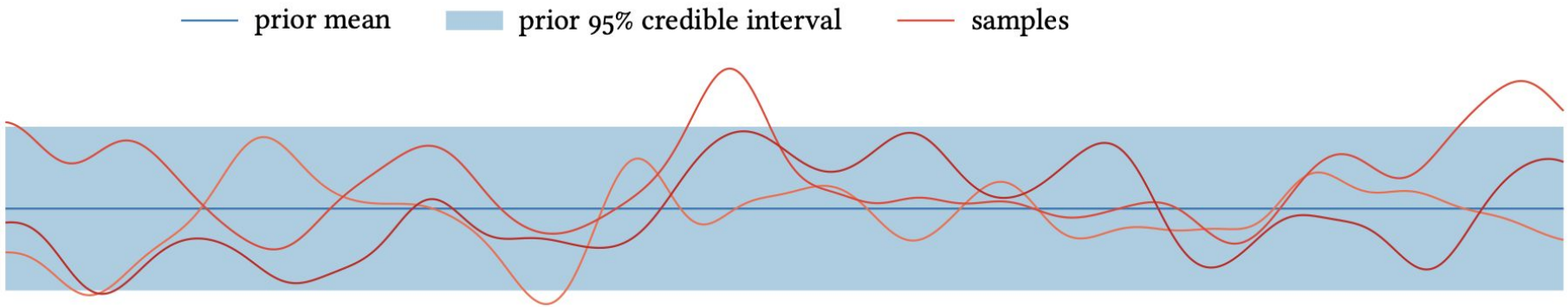


# Bayesian experimental design



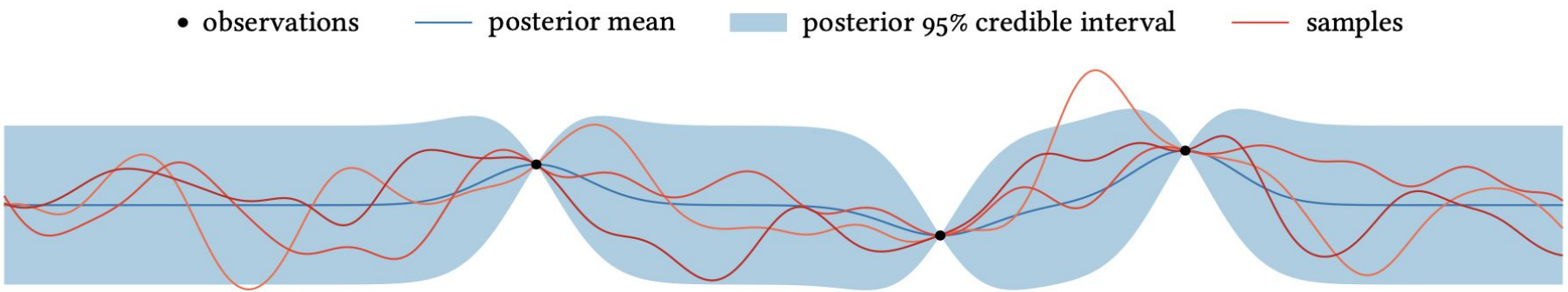
# Modeling experimental outcomes

Experimental design is driven by a model of the properties of interest; this can take many forms but ultimately yields *predictions* with *quantified uncertainty.* We begin with a prior...

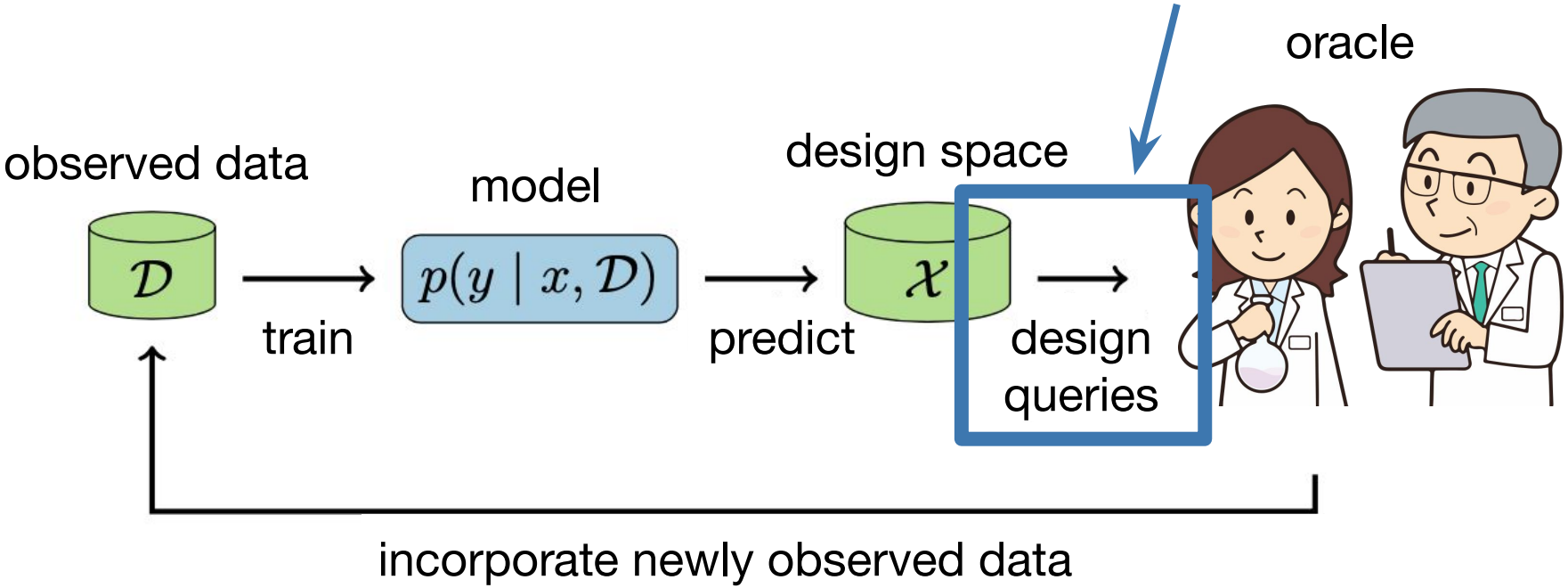


# Updating the model

...and we *condition* the model on data as we acquire it, forming a posterior belief encapsulating our current understanding of the properties of interest:

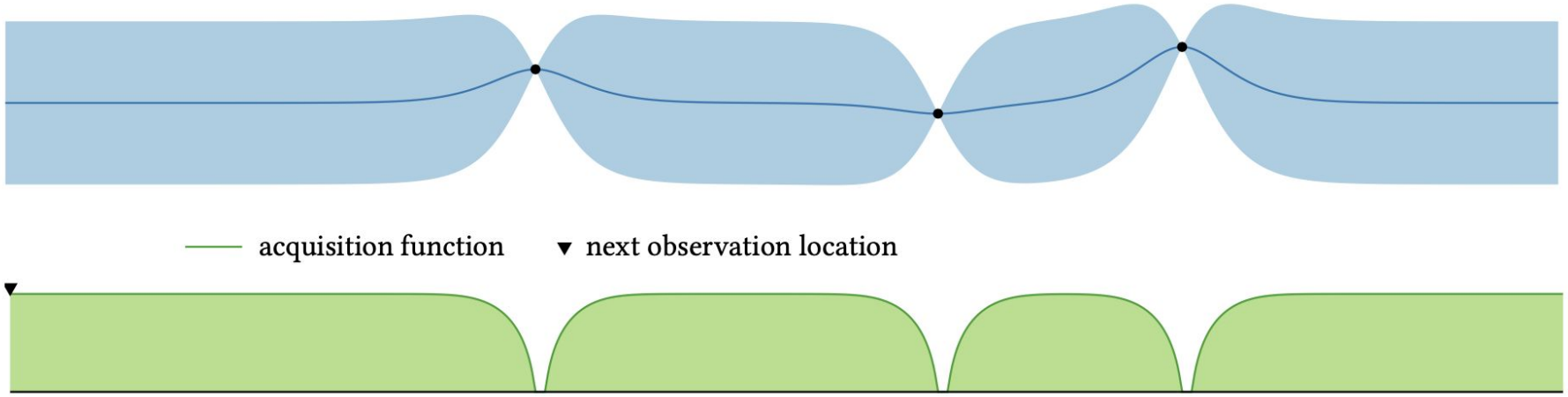


# Bayesian experimental design



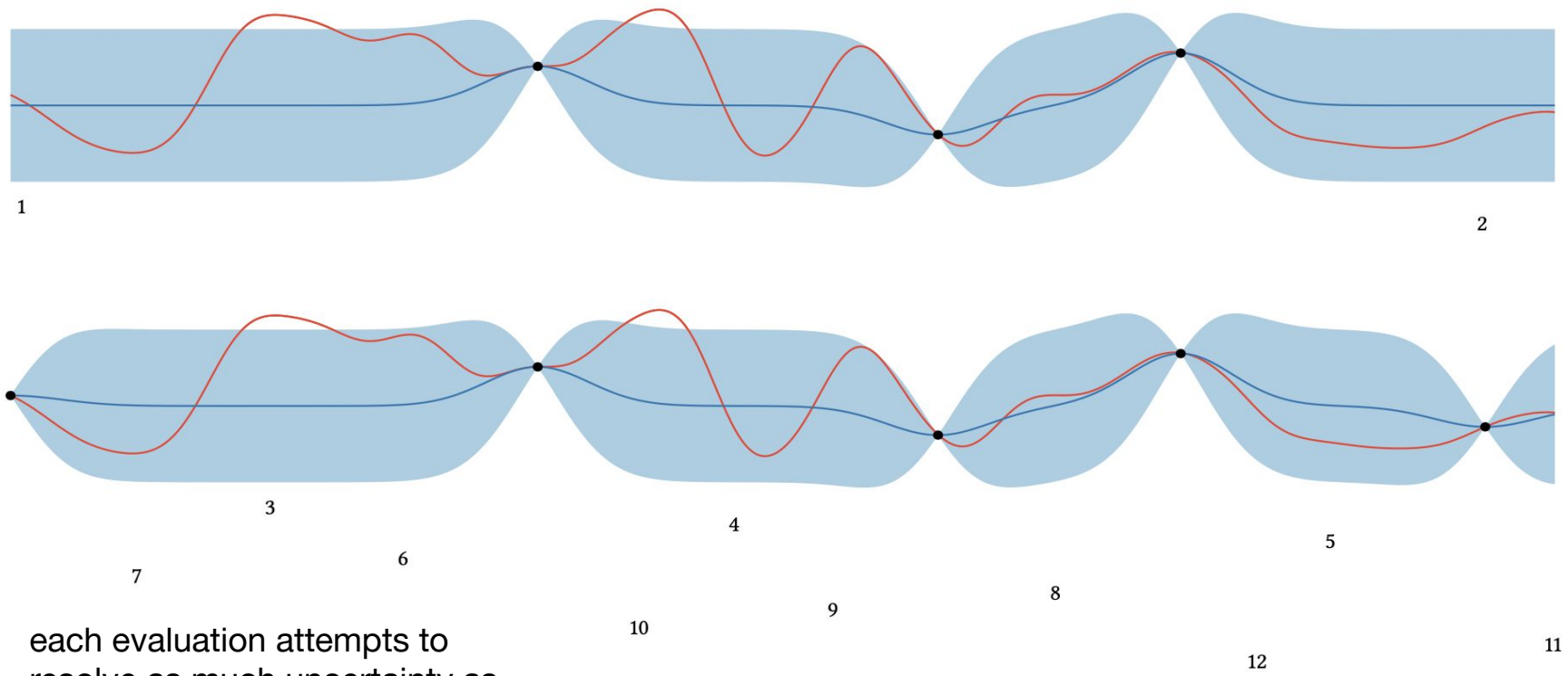

A procedure mapping our belief to experiment(s) is called a *policy.*

Bayesian experimental design operates by deriving and optimizing a score function (or *acquisition function*) from the model that somehow quantifies how useful a given measurement would be for our goals.



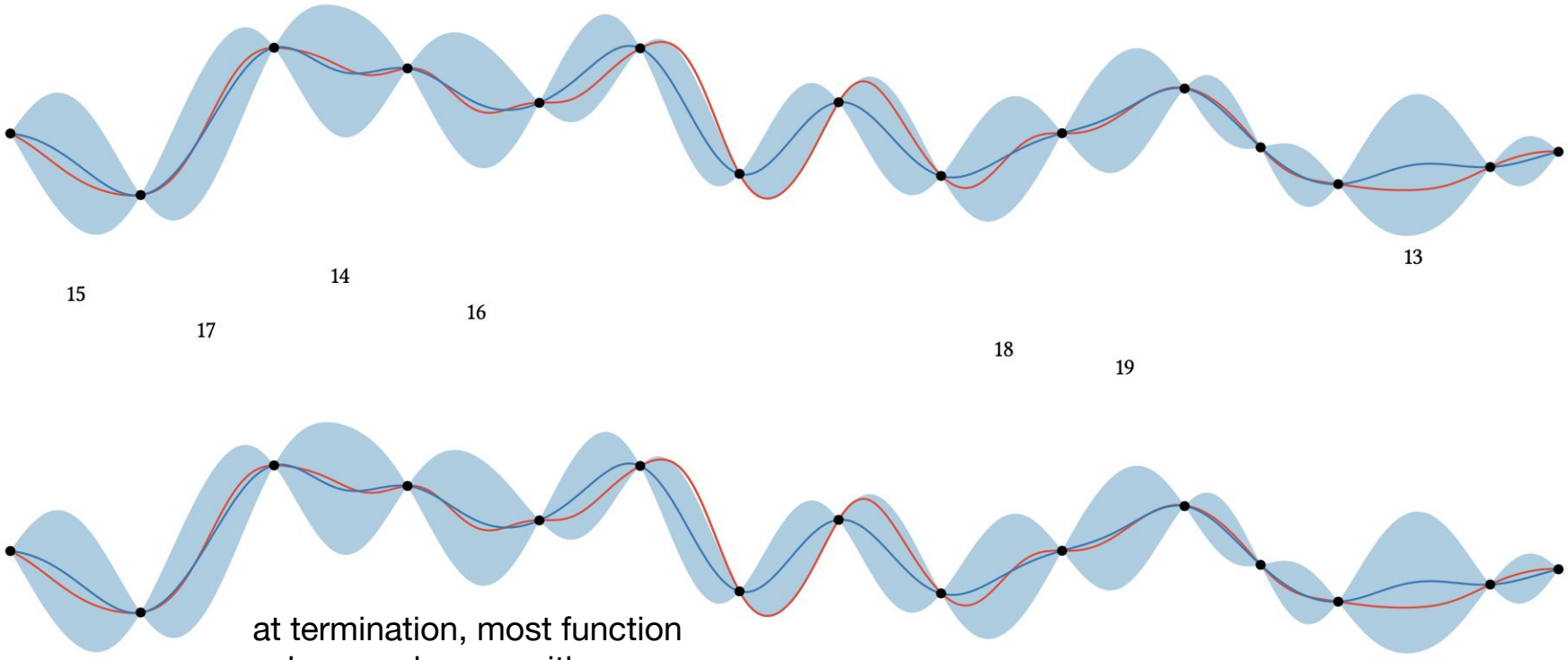

Optimizing the acquisition function tells us where to measure next.

# Demo: Pure learning




each evaluation attempts to resolve as much uncertainty as possible

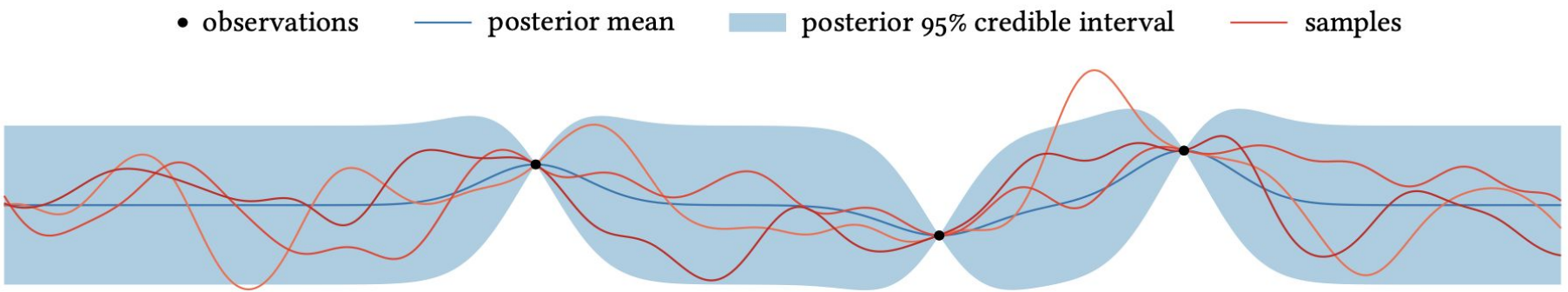
# Demo: Pure learning




at termination, most function values are known with confidence!

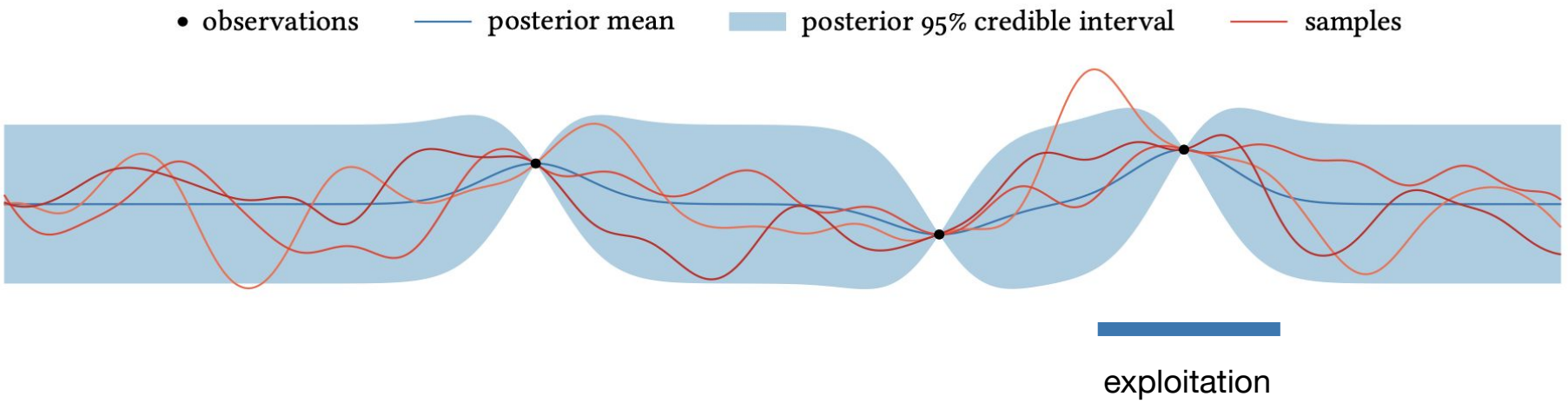
# Exploration vs. Exploitation

In the case of optimization, a key idea in policy design is the *exploitation–exploration* tradeoff: do I evaluate somewhere I believe will yield good properties? Or do I go somewhere unknown?



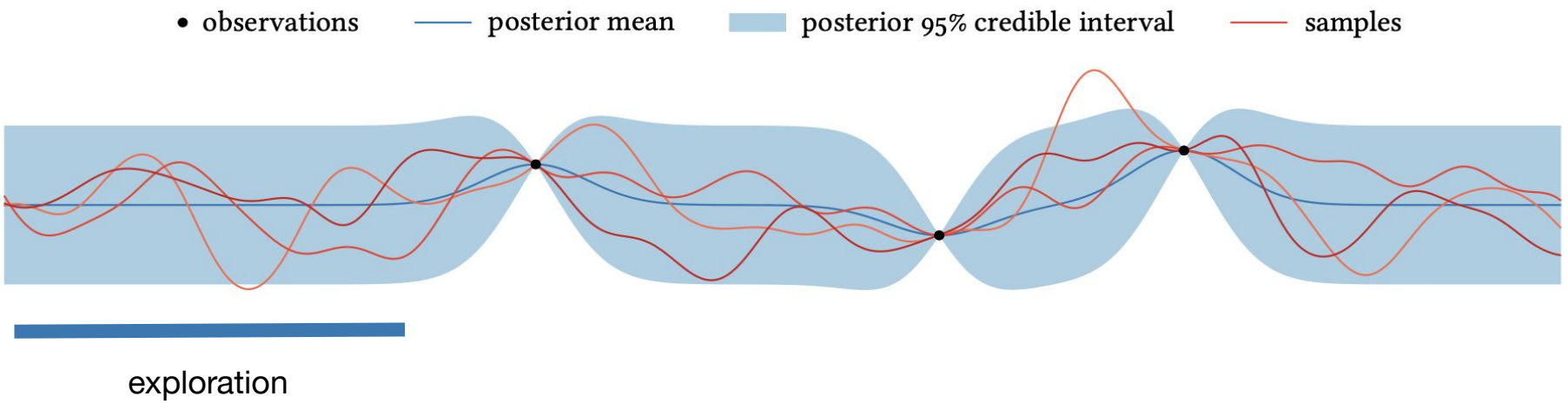
# Exploration vs. Exploitation

Exploitation can lead to *immediate improvement...*

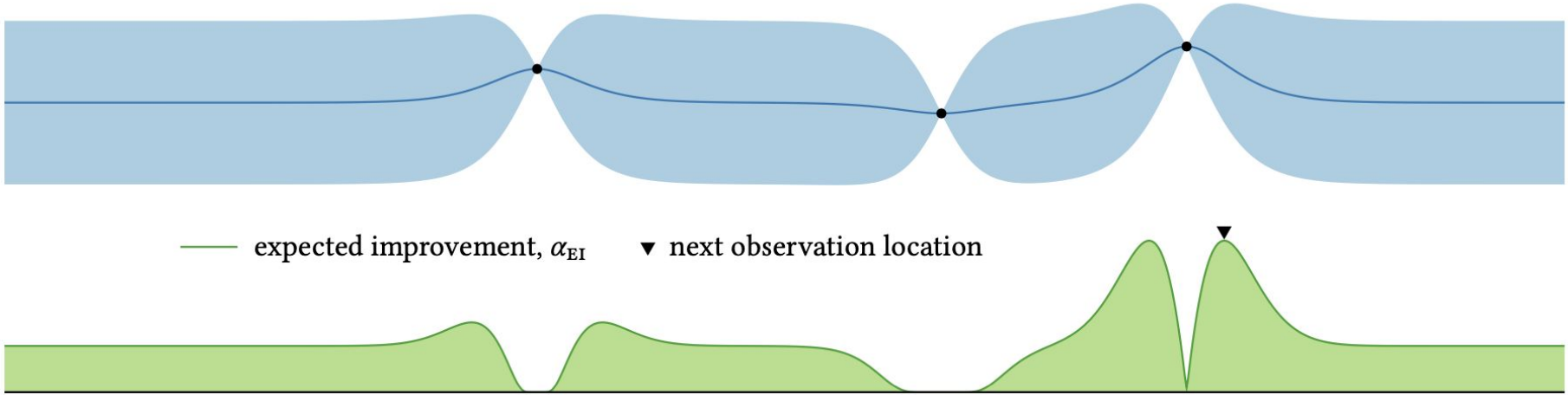


# Exploration vs. Exploitation

...whereas exploration can enable *future improvements.*

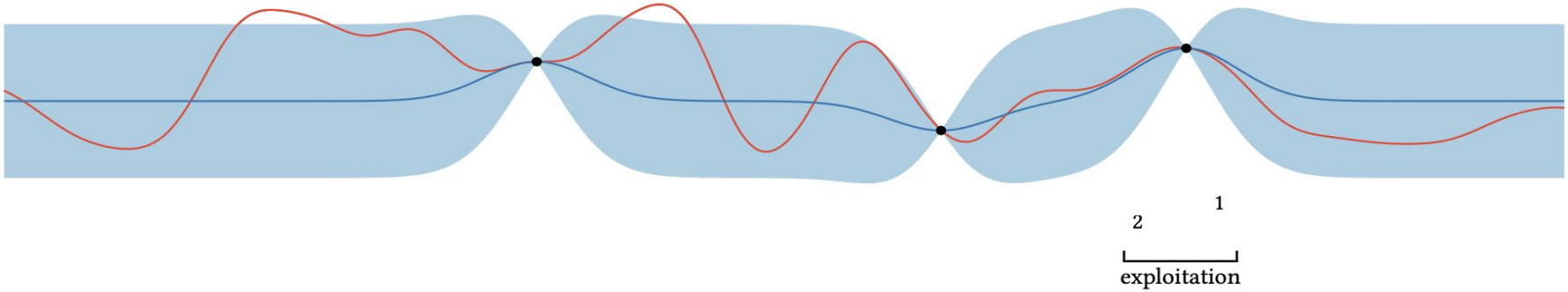


For effective optimization, we must construct acquisition functions that very carefully trade off exploration vs. exploitation.

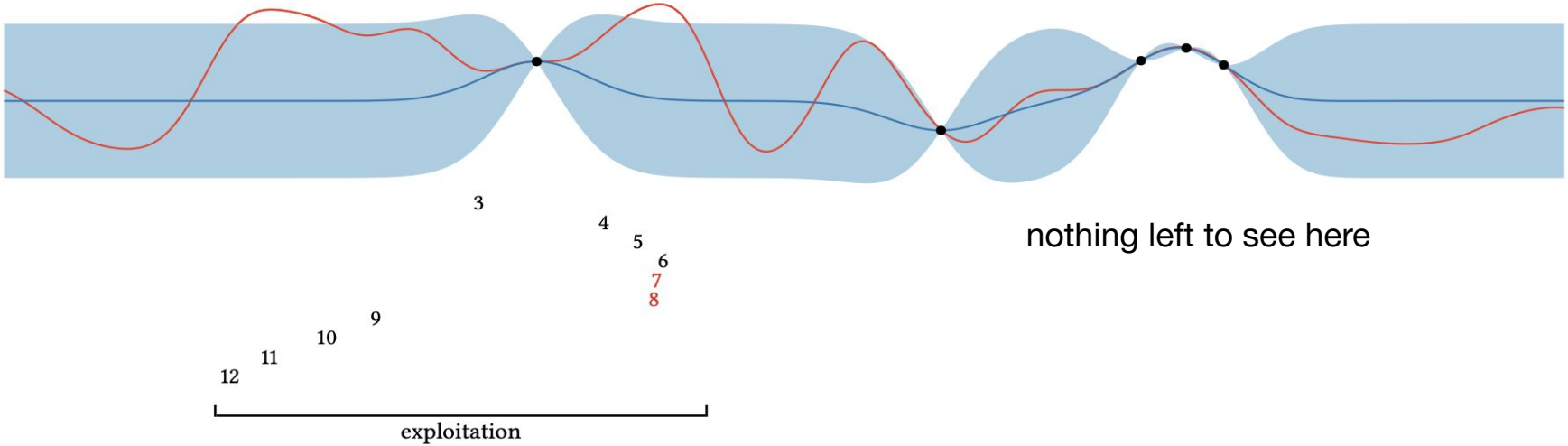


# Demo: Optimization

most promising region to start

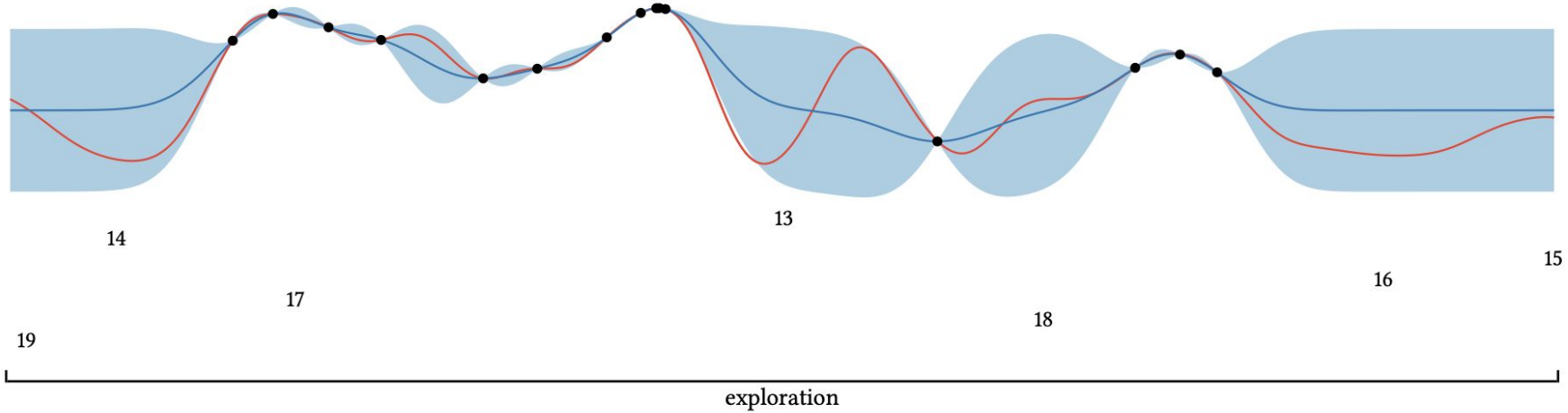


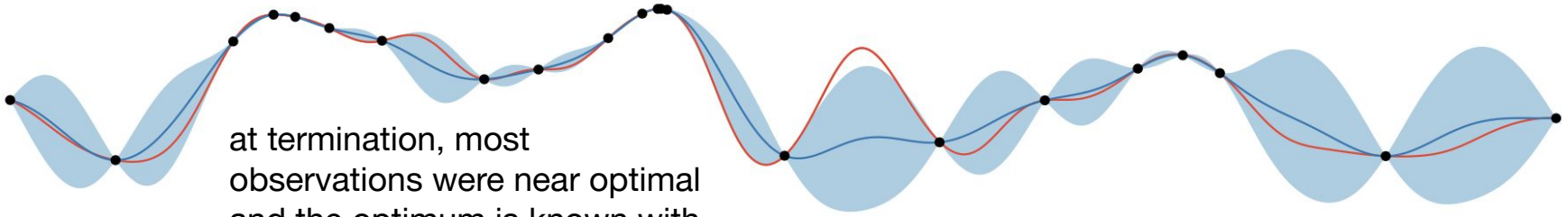

next most promising region




nothing left to see here

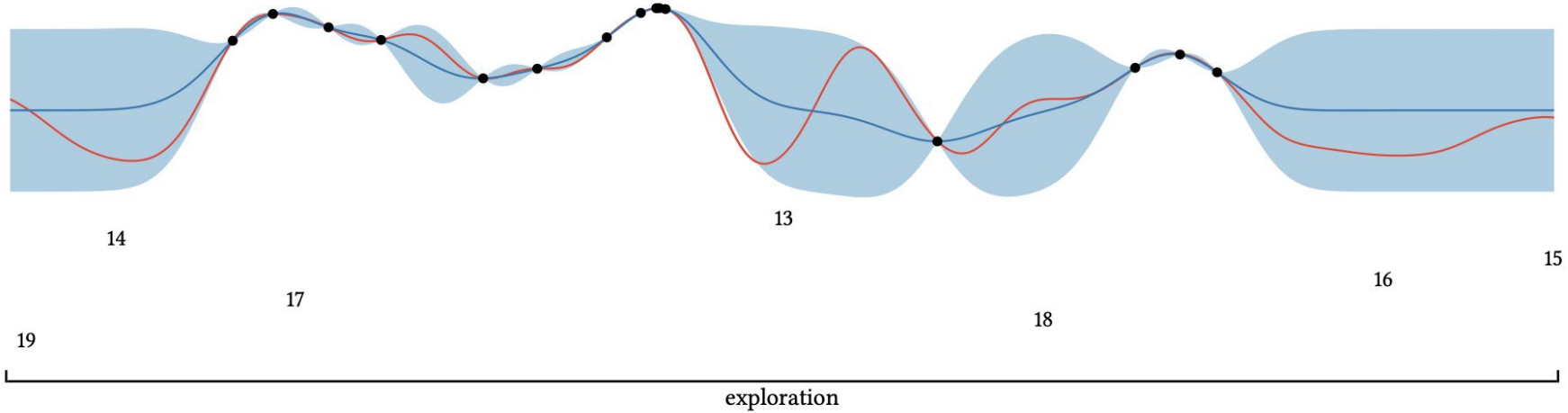
# Demo: Optimization

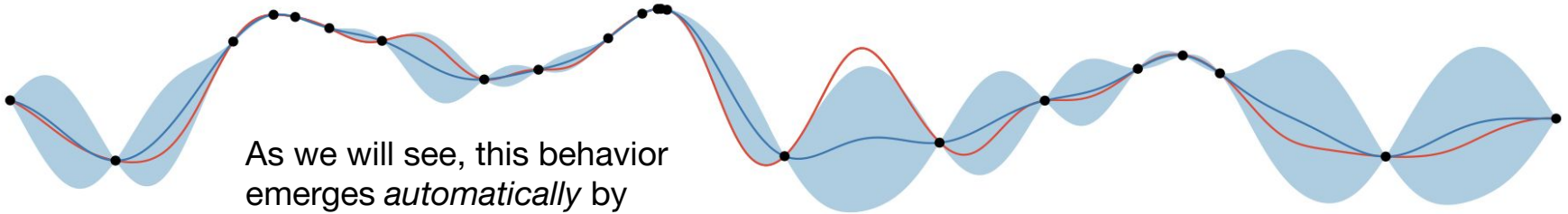


at termination, most observations were near optimal and the optimum is known with confidence!

# Demo: Optimization



As we will see, this behavior emerges *automatically* by "turning a Bayesian crank."

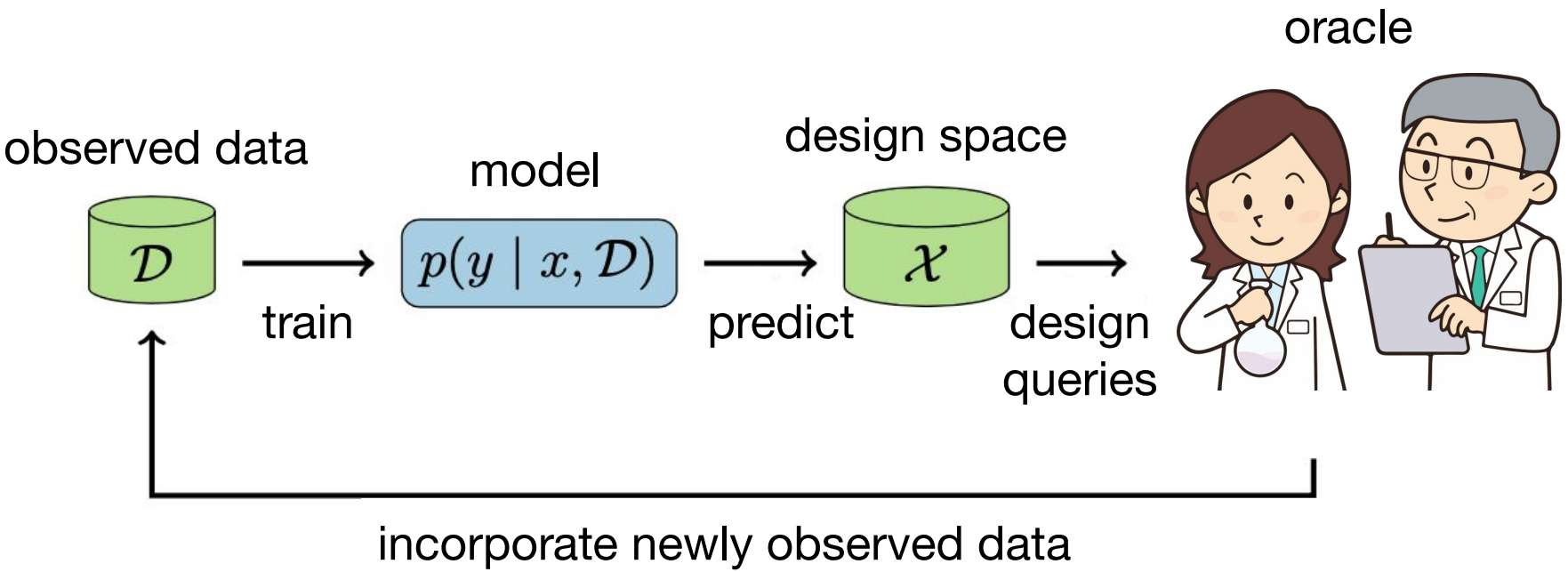
# Caveats

There is some effort required in building, validating, and maintaining a model of the properties of interest, and further effort in computing the policy. However, the return for this investment is *unparalleled sample efficiency*: easily 20-100+ times more efficient than, e.g., random search.

| Rank | Team | Score | Median | RS Iters. | RS Efficiency |
|---|---|---|---|---|---|
| 1 | Huawei Noah's Ark Lab | 93.519 | 99.166 | 15,512 | 121.188 |
| 2 | NVIDIA RAPIDS.AI | 92.928 | 98.616 | 12,089 | 94.445 |
| * | AutoML.org | 92.551 | 98.693 | 10,353 | 80.883 |
| 3 | JetBrains Research | 92.509 | 99.131 | 10,179 | 79.523 |

results from NeurIPS 2020 black-box optimization competition

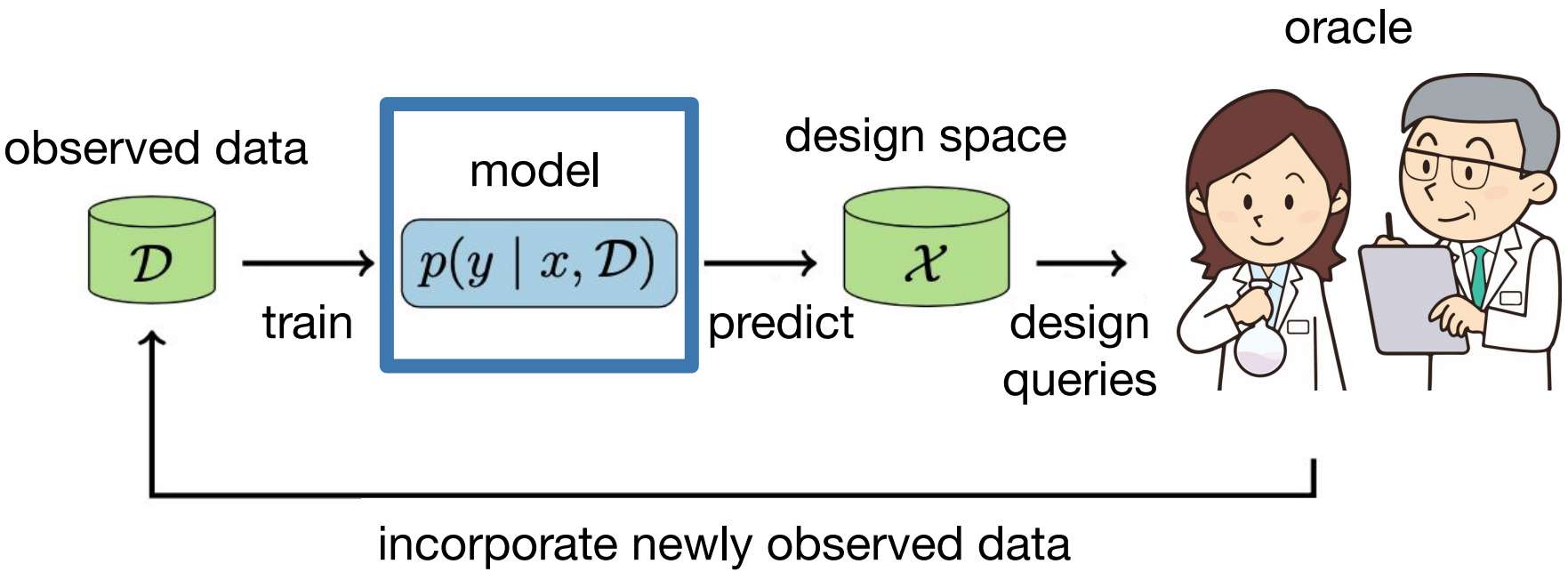
# Bayesian experimental design



# Modeling and inference

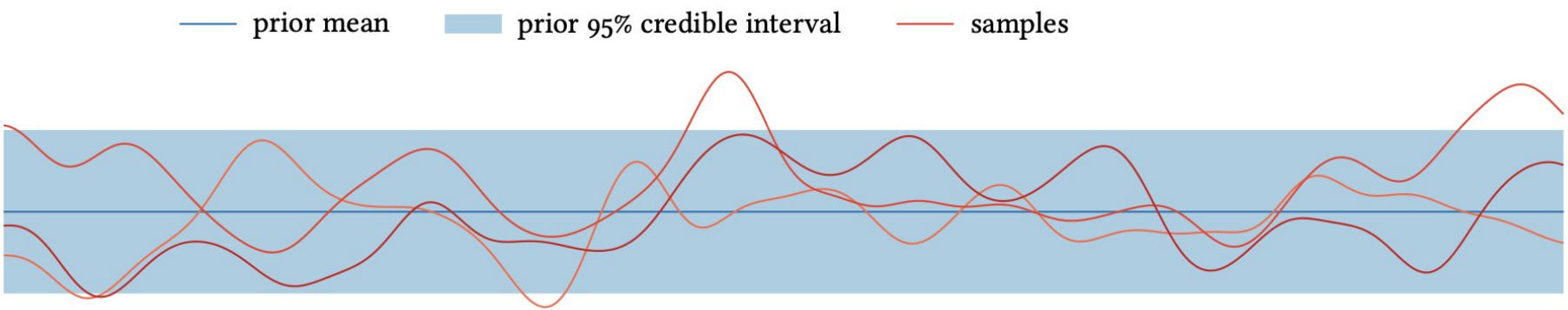
[BayesOpt Book Chapters 2-4]

# Bayesian experimental design



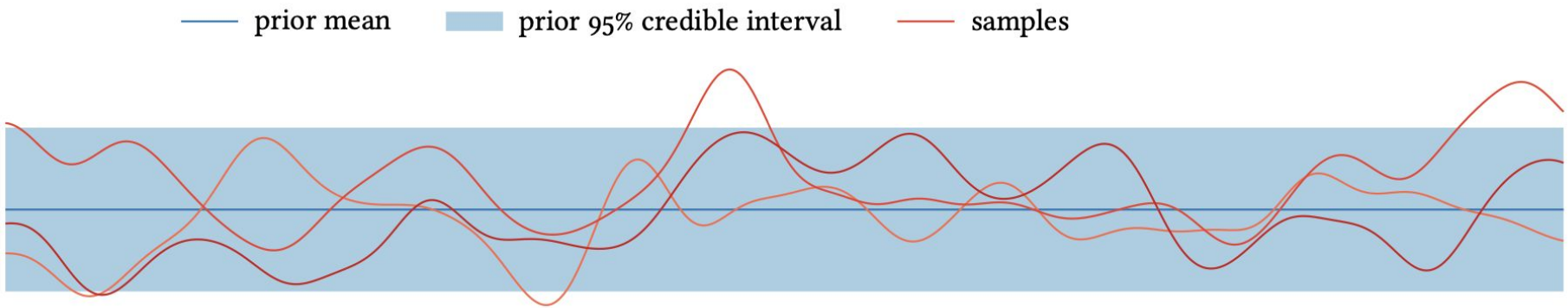
# Modeling properties and observations

Building a model of properties of interest and our observations of them is an essential step of the Bayesian approach to experimental design.



Gaussian processes are especially convenient, both for building expressive models and enabling straightforward inference. They are by *far* the most common choice for Bayesian optimization.

In particular, they are a solid choice when faced with “small data.”



Of course, using a Gaussian process is not mandatory if another model would be more suitable:

- random forests,
- (Bayesian) neural networks,
- etc.

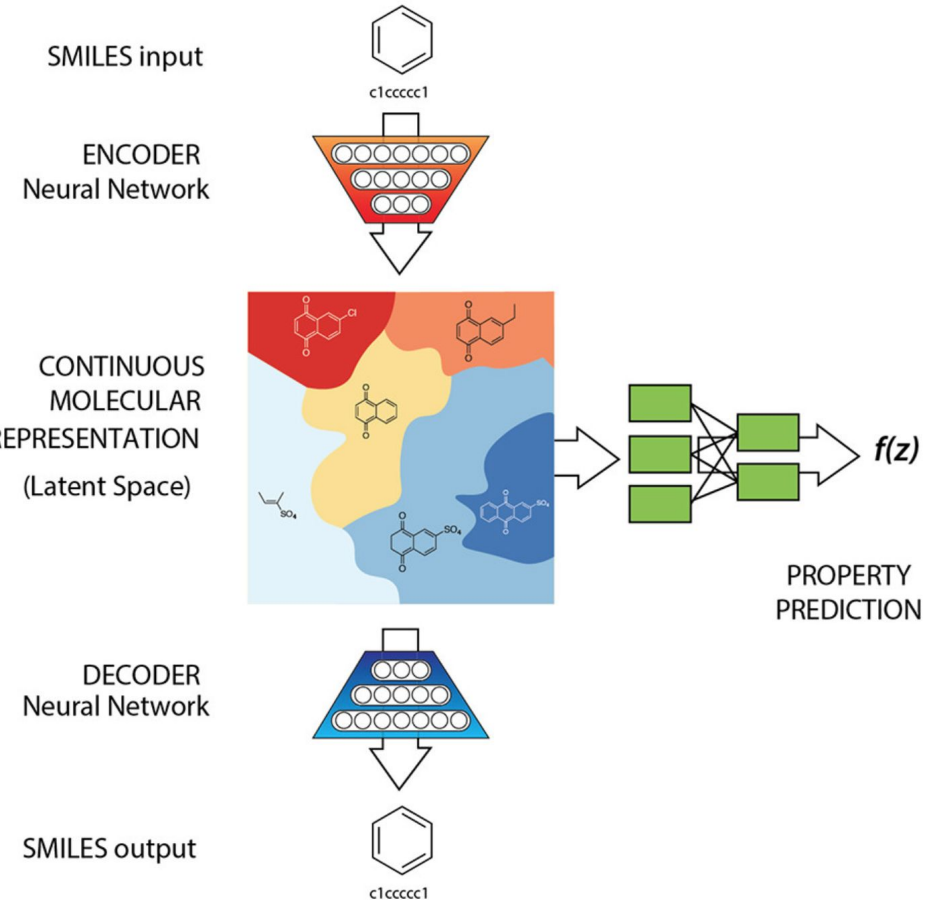
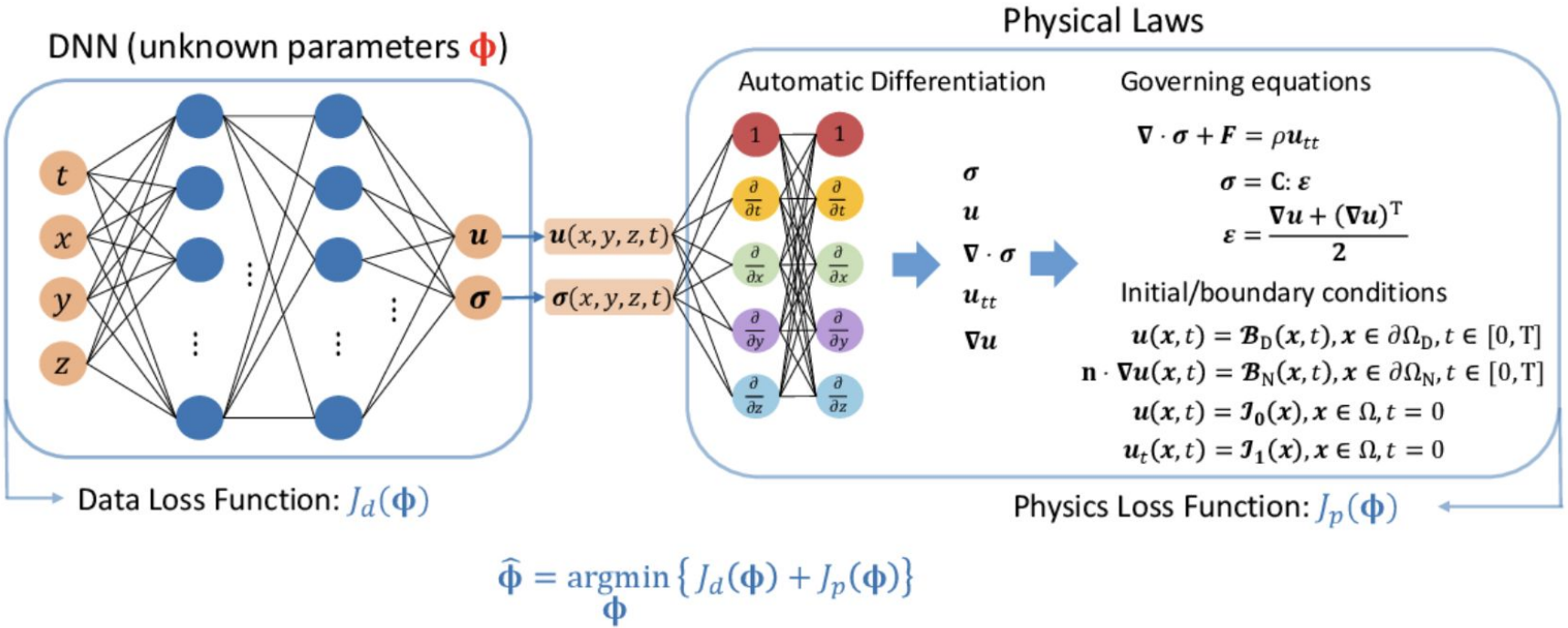



image: Gómez-Bombarelli, et al. (2018)

In some cases we might be able to build a sophisticated model, e.g. a physics-guided neural network.




$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}} \{ J_d(\phi) + J_p(\phi) \}$$

image: George Nentidis

A recent innovation is to learn continuous representations of molecules/proteins/etc. using deep learning and then build models in the discovered latent space. We may even continue to tweak as we learn.

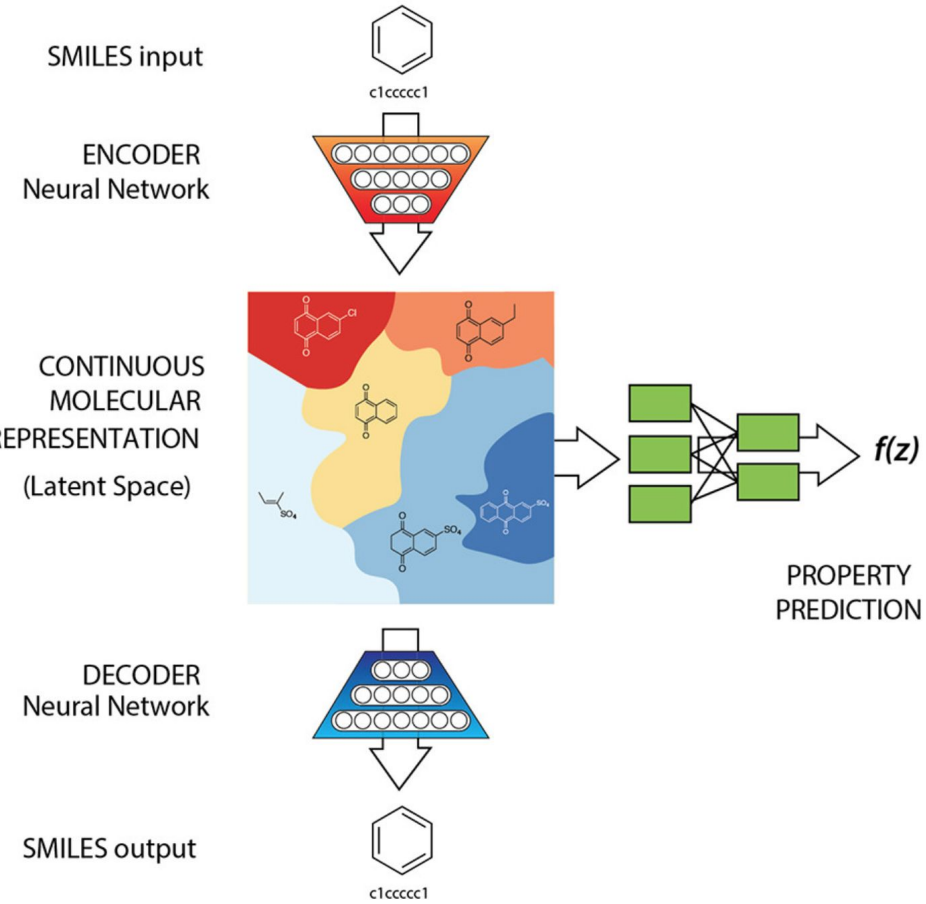
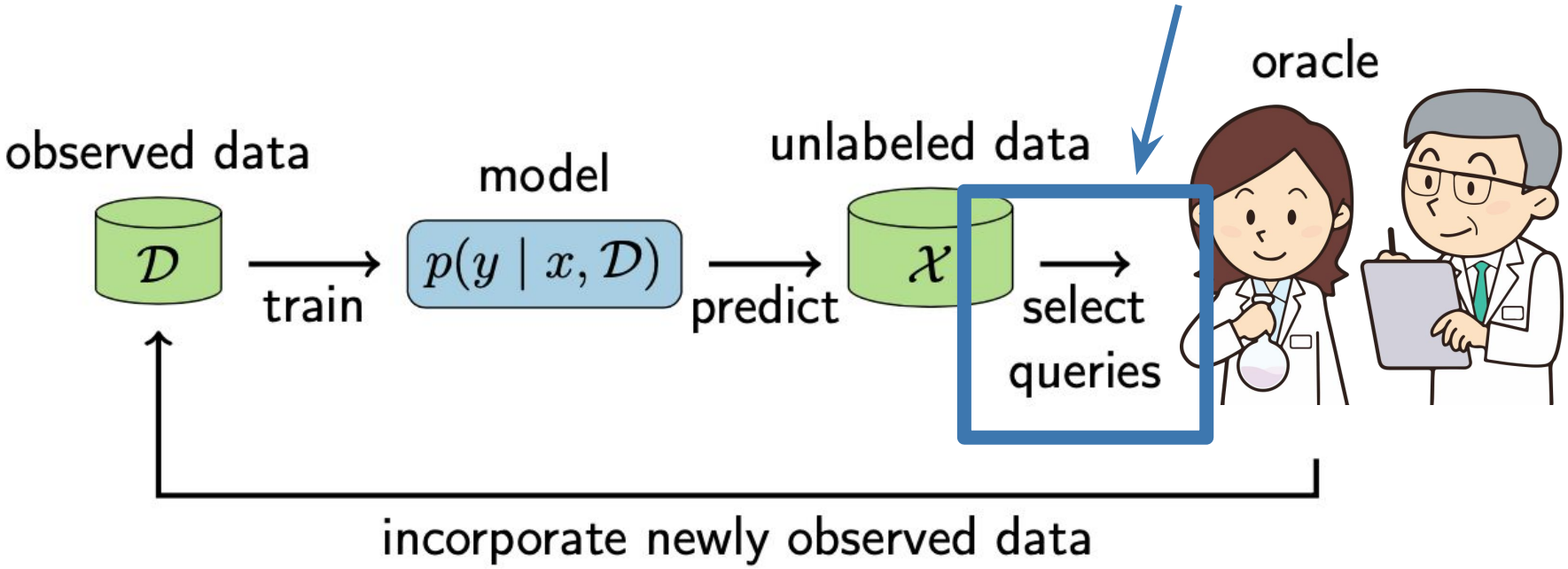



image: Gómez-Bombarelli, et al. (2018)

# Building a policy

[BayesOpt Book Chapters 5-8]

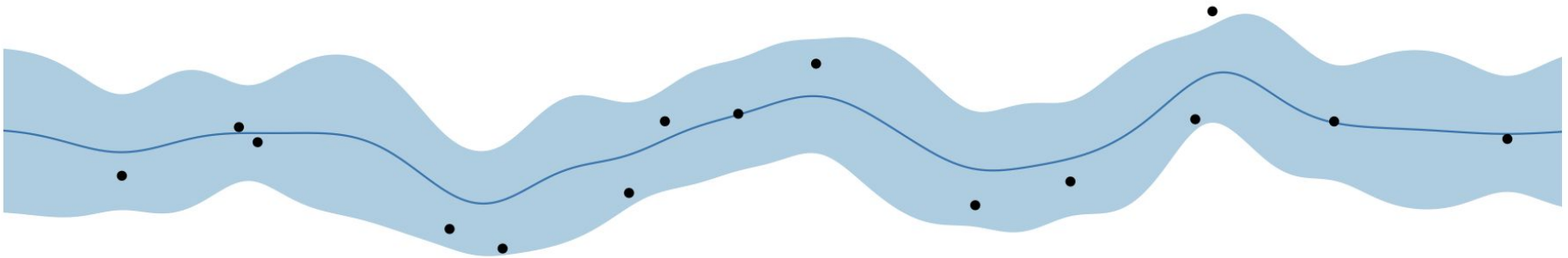
# Bayesian experimental design




Recall a *policy* transforms our beliefs into insightful experiments.

# Bayesian decision theory

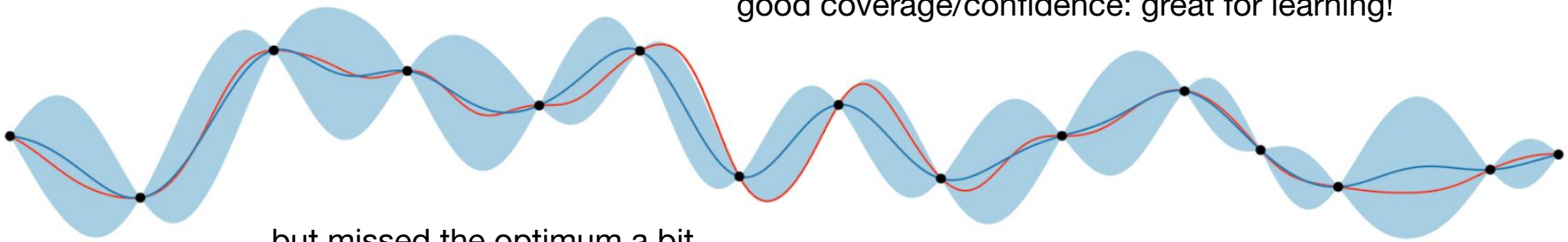
One common approach is policy design is to appeal to *Bayesian decision theory*. The key idea is to define a *utility function* quantifying how useful a given set of experimental results are *in retrospect.*



Whereas the model summarizes our *beliefs*, the utility function summarizes our *preferences.*

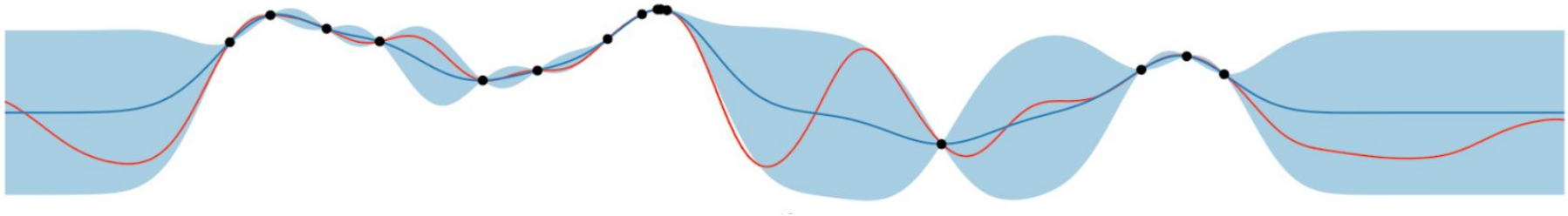
# Example

good coverage/confidence: great for learning!



...but missed the optimum a bit

found the optimum with confidence...



...but has some regions with high uncertainty

# Bayesian decision theory

At its heart, Bayesian decision theory is quite simple. Once we have selected a utility function for our outcomes, we simply choose the action maximizing the *expected* utility resulting from that action:

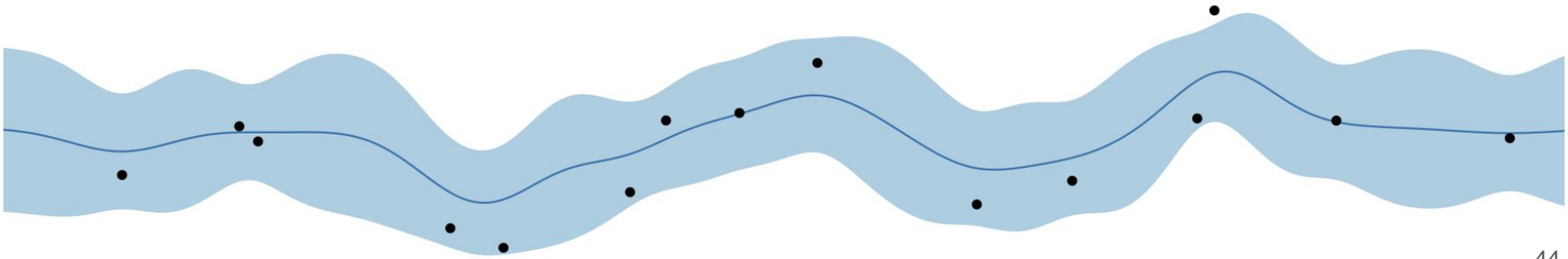
$$\text{optimal decision} = \operatorname{argmax} \mathrm{E}[\text{utility after action}]$$

Here the expectation is taken with respect to our belief over anything unknown.

# Bayesian decision theory

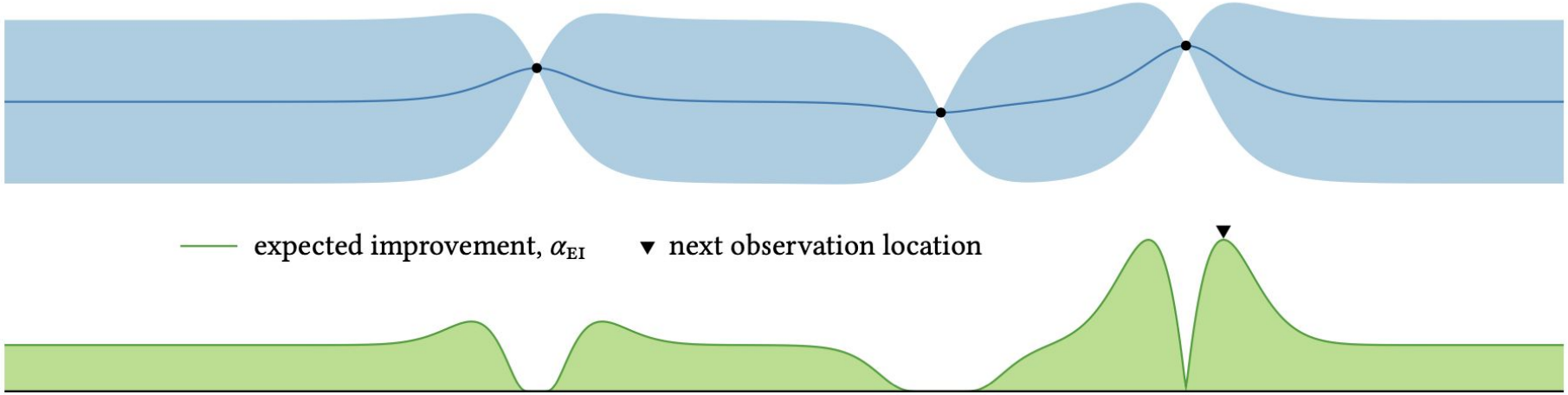
In the context of experimental design, the unknown information in the expectation critically includes the outcome of a proposed experiment! This was the whole point of modeling!

optimal decision = argmax E[utility after experiment]



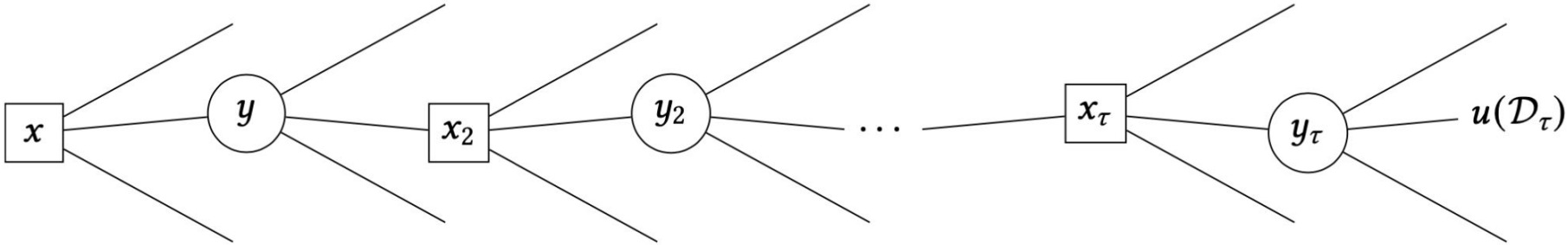
# Bayesian decision theory

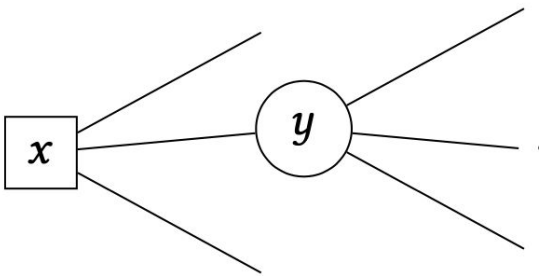
We can now create a policy. We create an acquisition function that is simply the expected utility from adding a proposed experiment to our data, then run the experiment with the highest expected utility!



# (Non)myopia in planning experiments

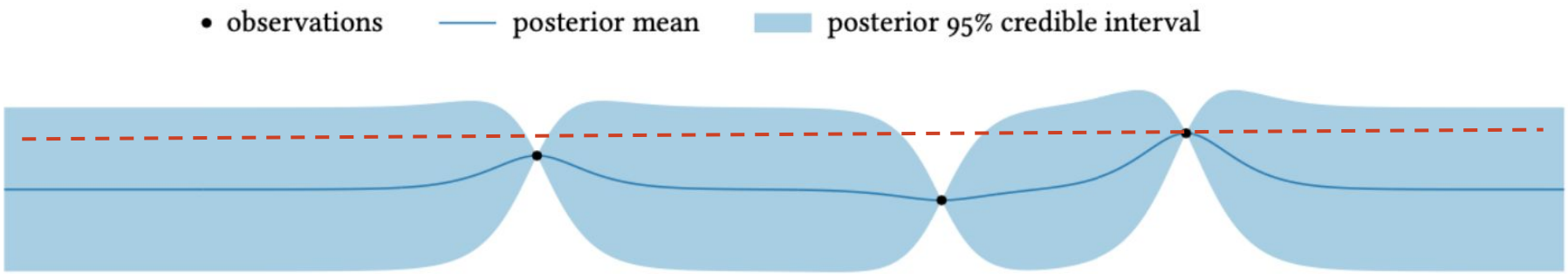
Theoretically we would like to plan *ahead* in experimental design, considering how every sequential decision will affect our final data.

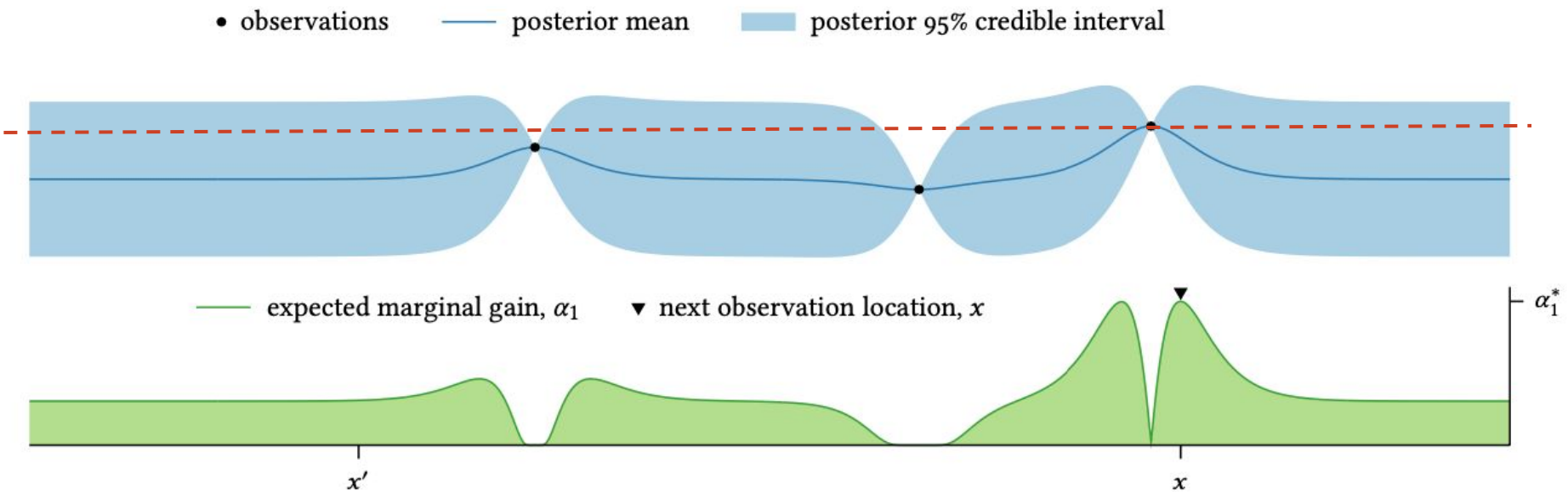


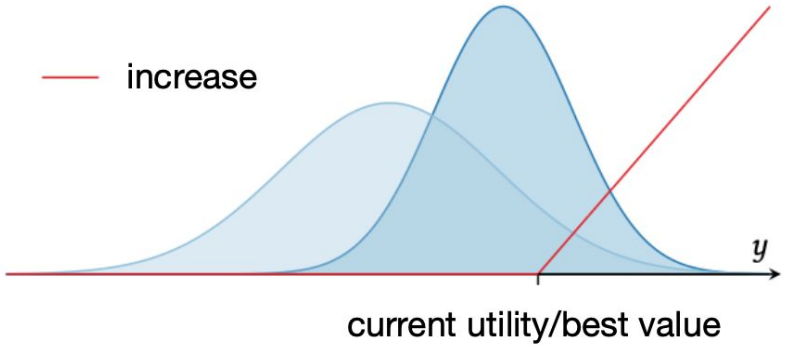




Unfortunately this is extremely expensive, so we usually only plan a few (often just one) step ahead. This is known as a *myopic approximation* or (*one-step*) *lookahead*.

Let’s consider an example from optimization. The *simple reward* utility is effectively “the highest property value I have seen.”

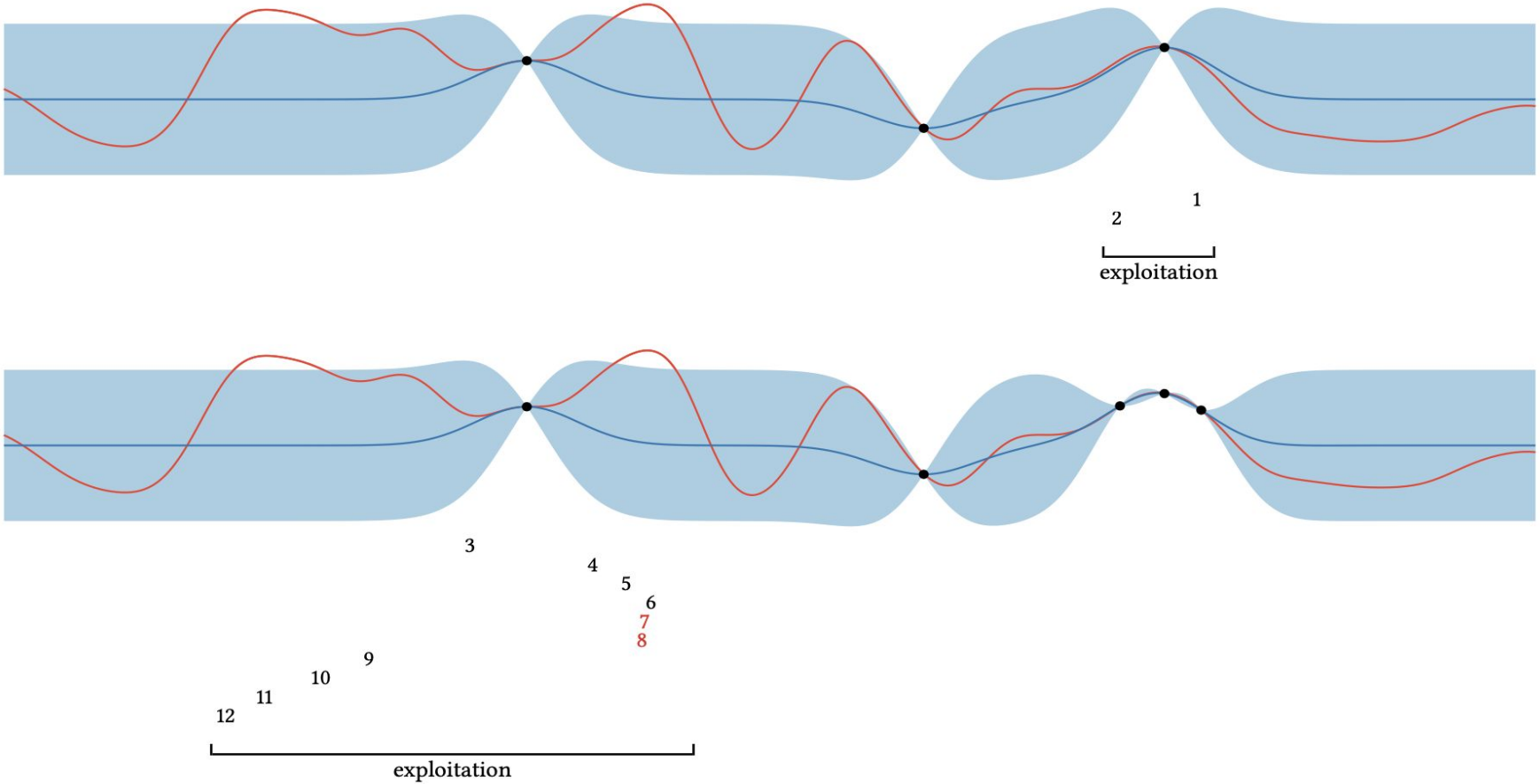




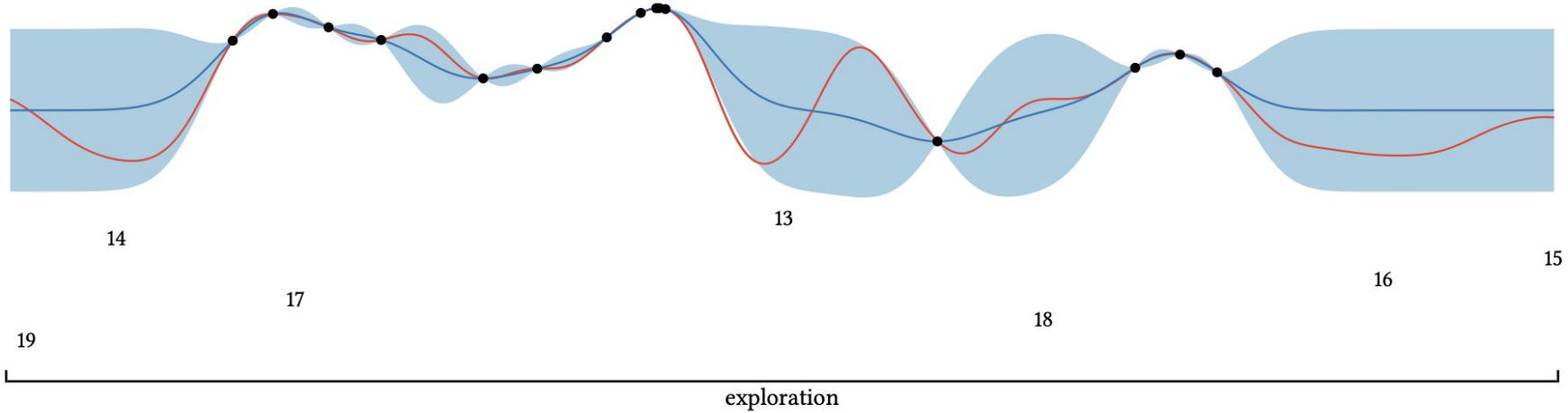



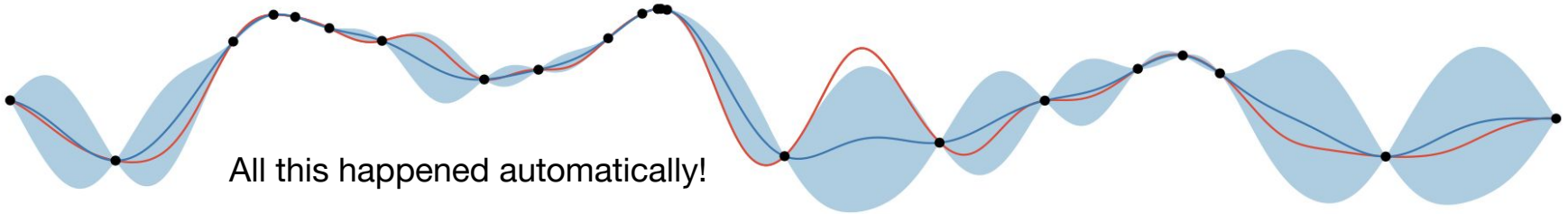

Computing this expectation at every point in the design space reveals the green acquisition function.

# Emergent behavior



# Emergent behavior



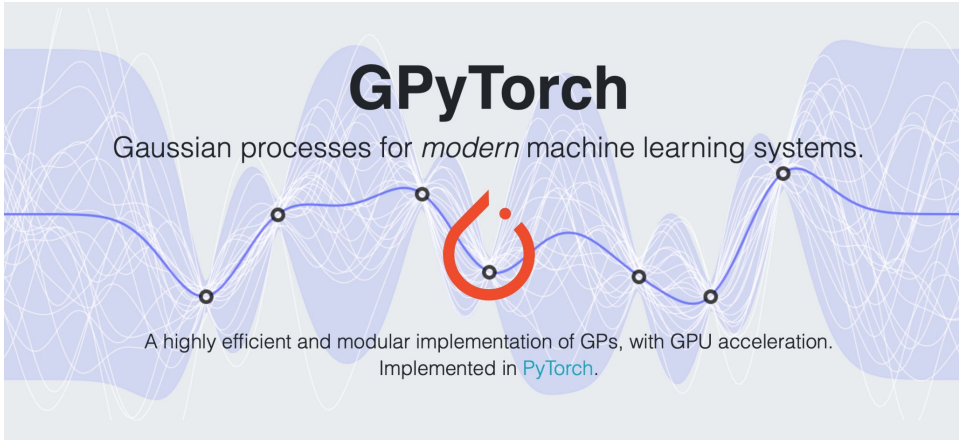



All this happened automatically!

# A complete recipe

1. choose a utility function encapsulating your experimental goals
2. identify a design space and model the properties of interest

3. compute the expected utility of every possible experiment in light of current beliefs
4. run the experiment maximizing the expected utility
5. add the outcome to your data
6. update your model
7. go to step 3 and repeat as desired



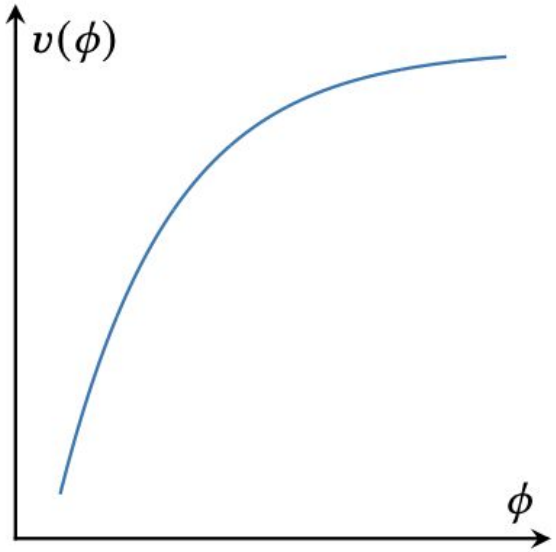

Fortunately there is excellent *off-the-shelf software* available to assist in building such a system.

# Utility Functions and Common Policies

[BayesOpt Book Chapter 6–7]

# Off-the-shelf utilities

Designing a utility function consistent with your preferences can be challenging. Fortunately, there are well-behaved utility functions available *off-the-shelf* for common tasks, and performing *one-step-lookahead* with them yields common policies!

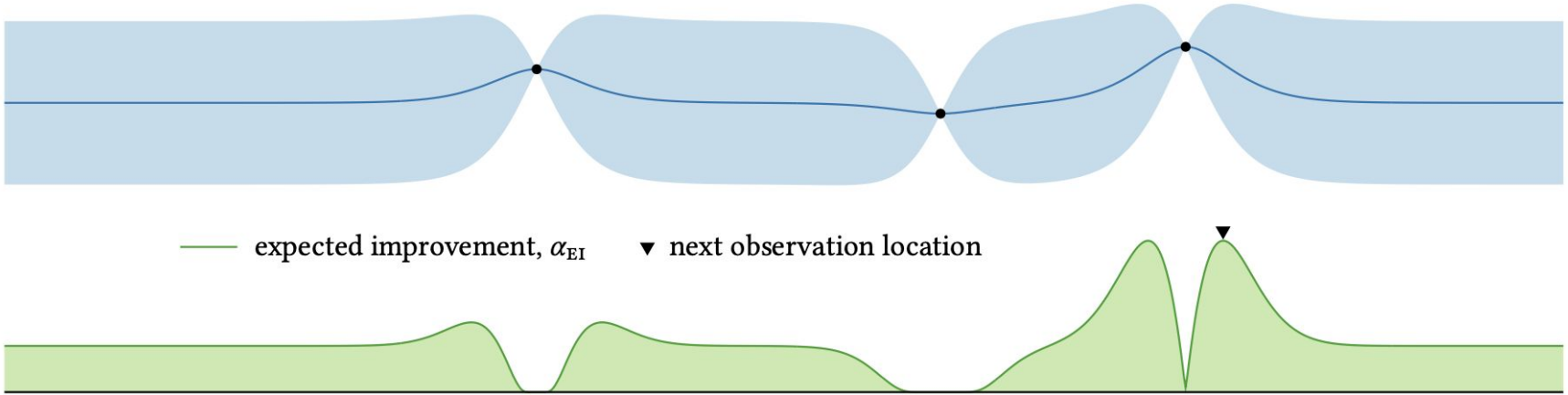


# Simple reward

Utility: highest function value evaluated during optimization

(measured by the posterior mean)

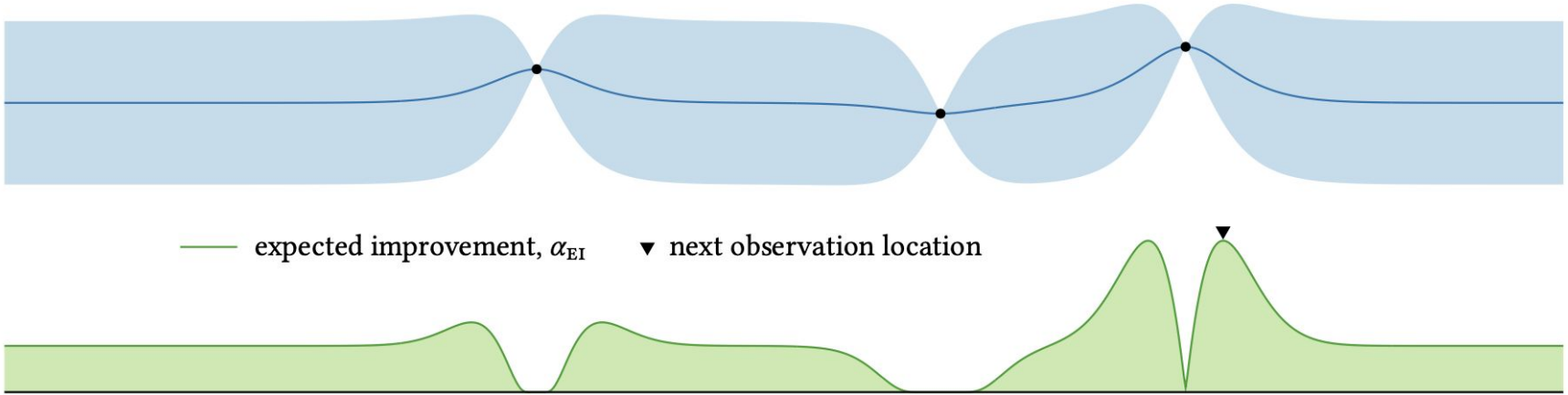
One-step lookahead yields **expected improvement**



# Simple reward

Utility: expected utility of optimal risk-neutral recommendation after termination

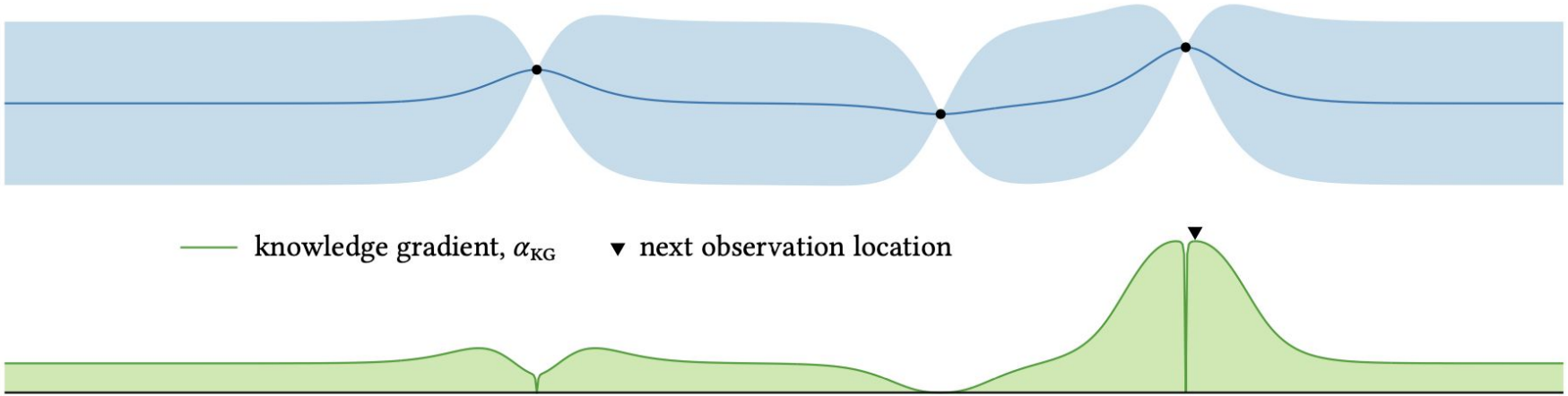
One-step lookahead yields **expected improvement**

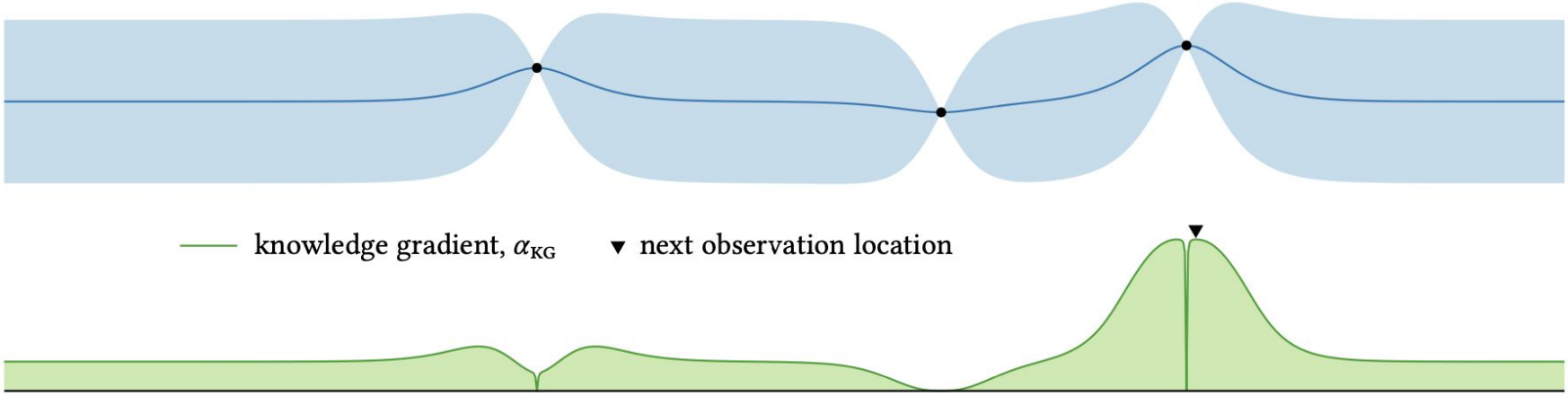


# Global reward

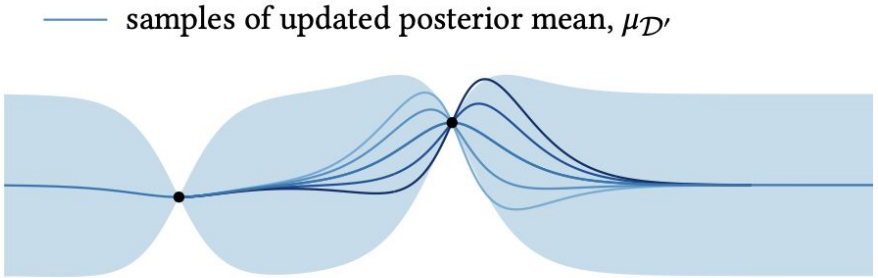
Utility: highest function value ~~evaluated during optimization~~ *anywhere*

(measured by the posterior mean)





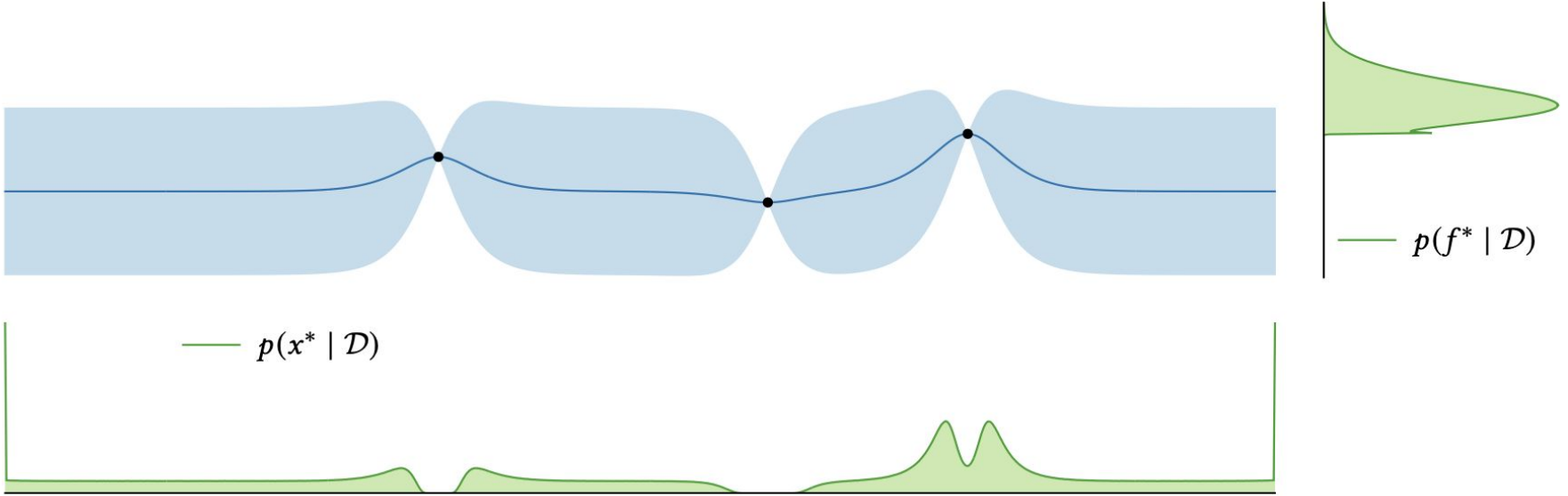

because the maximum is allowed to occur anywhere, the policy here automatically learned to use a finite difference approximation!



# Information gain

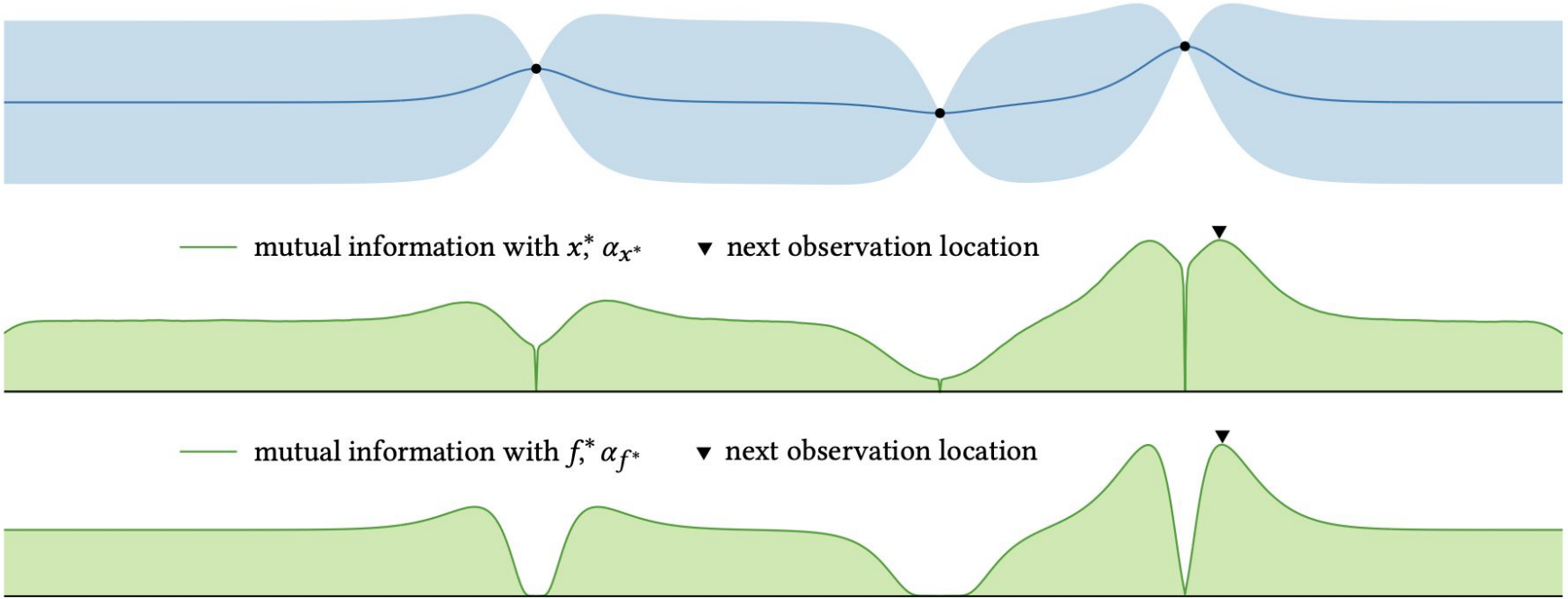
Utility: reduction in entropy about location or value of global optimum

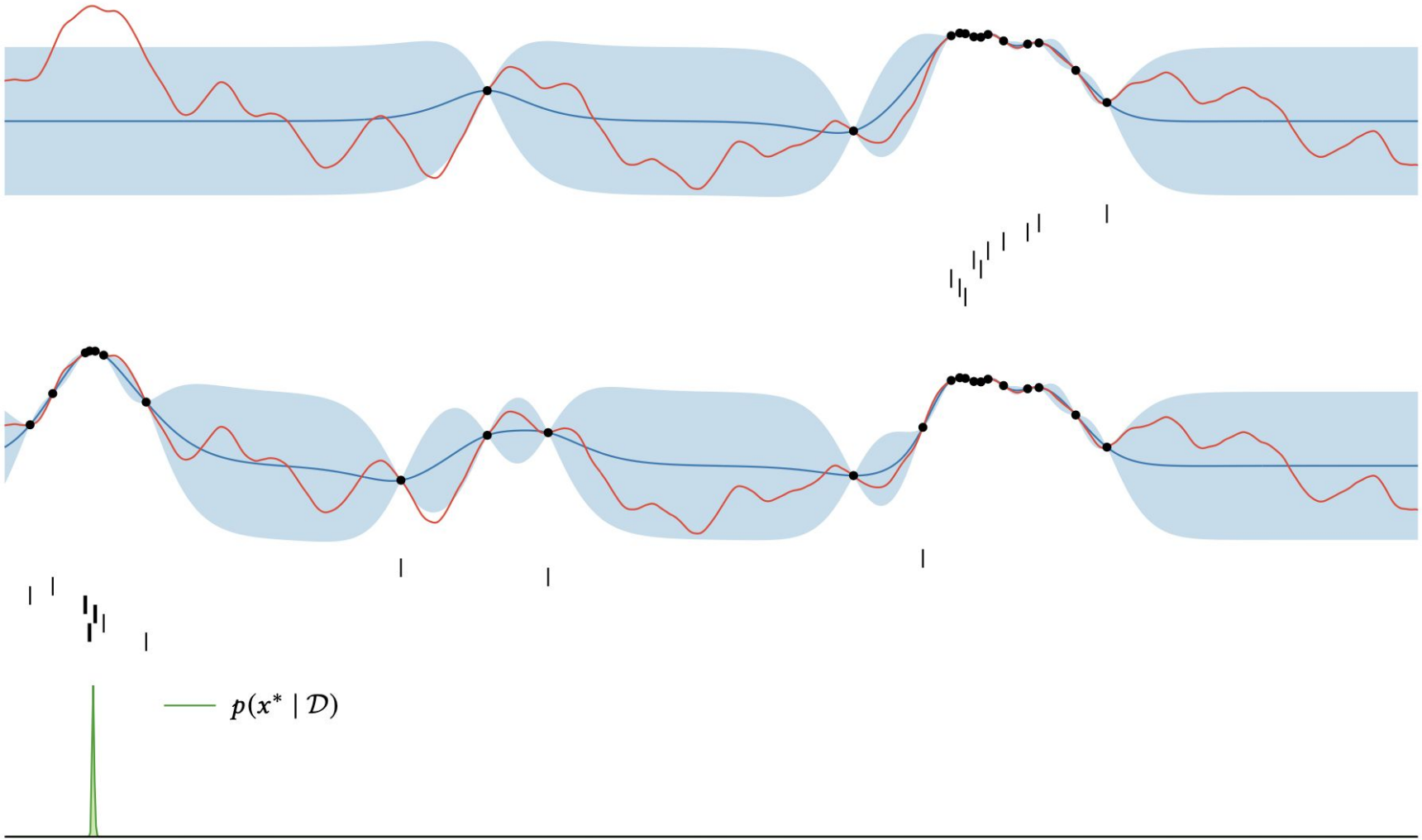
(or, if you prefer, KL divergence moving from prior to posterior)

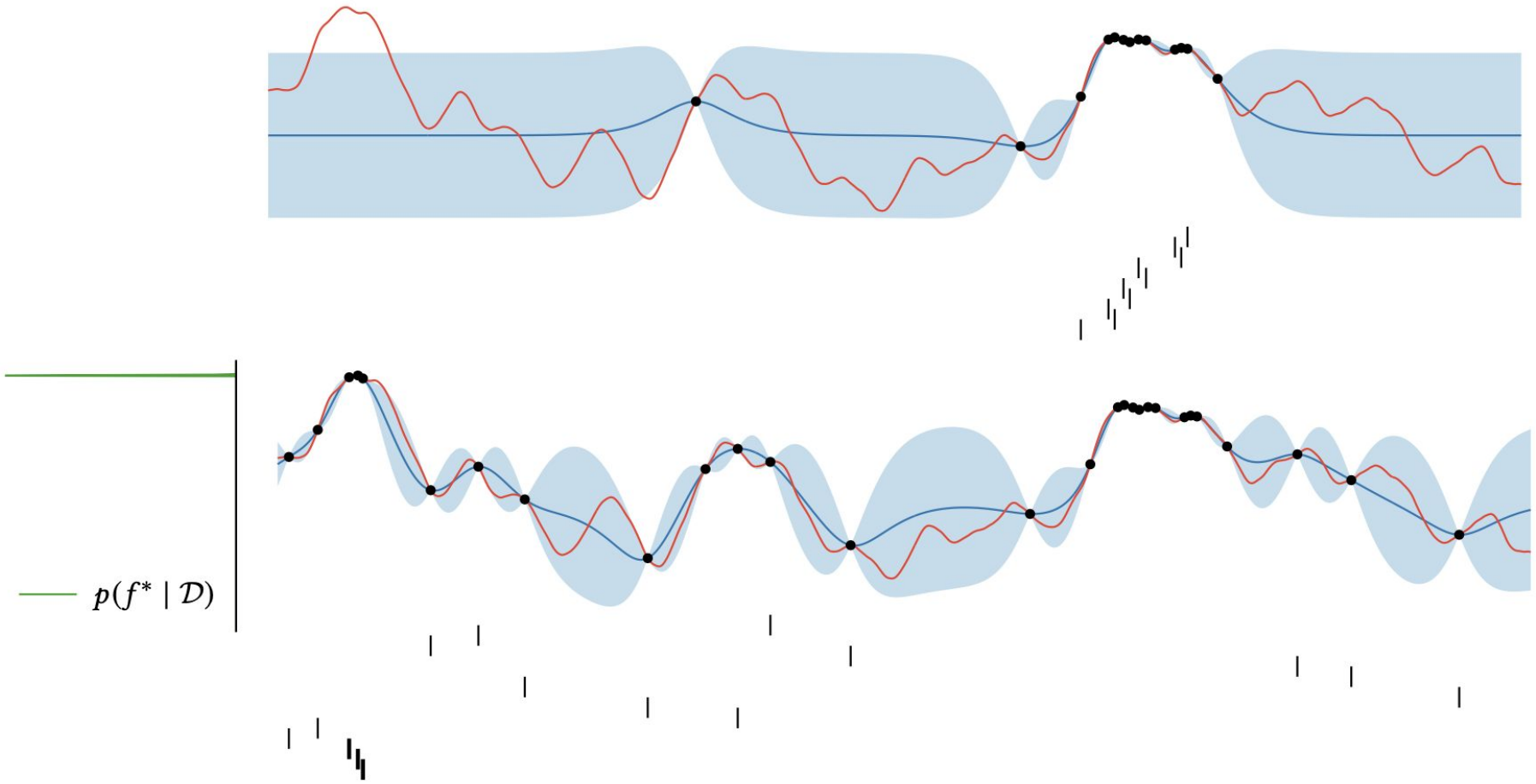


# Mutual Information / Entropy Search

One-step lookahead yields **mutual information** (*entropy search*)

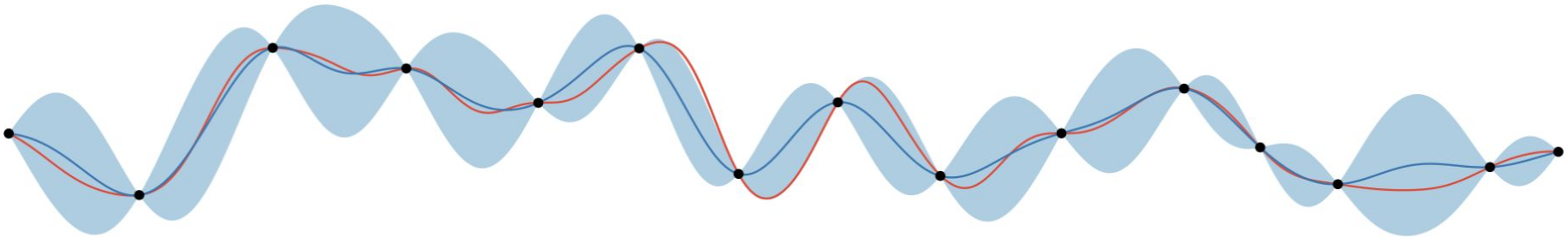



$p(x^* \mid \mathcal{D})$


$p(f^* \mid \mathcal{D})$

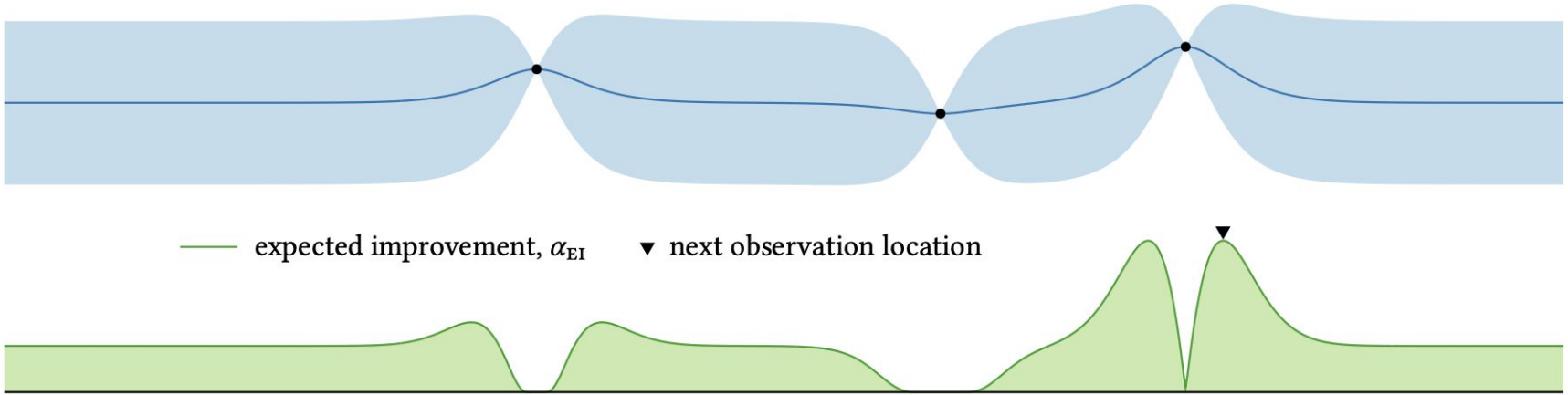
# Mutual information

Useful far beyond optimization! Can be regarded as a mathematical realization of the *scientific method* (Lindley 1956).



# Computation

All of these policies can be efficiently computed (or at least approximated) to yield relatively inexpensive and differentiable acquisition functions.



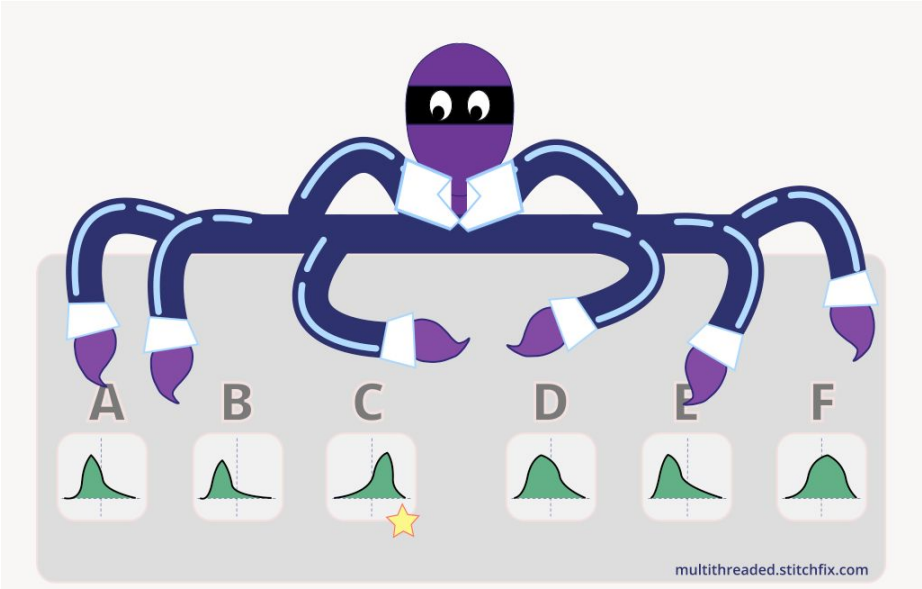
# Bandit Policies and Theory

[BayesOpt Book Chapters 7, 10]

# Multi-armed bandits

Another pathway toward optimization policies is to adapt policies from classical *multi-armed bandits* to this setting.

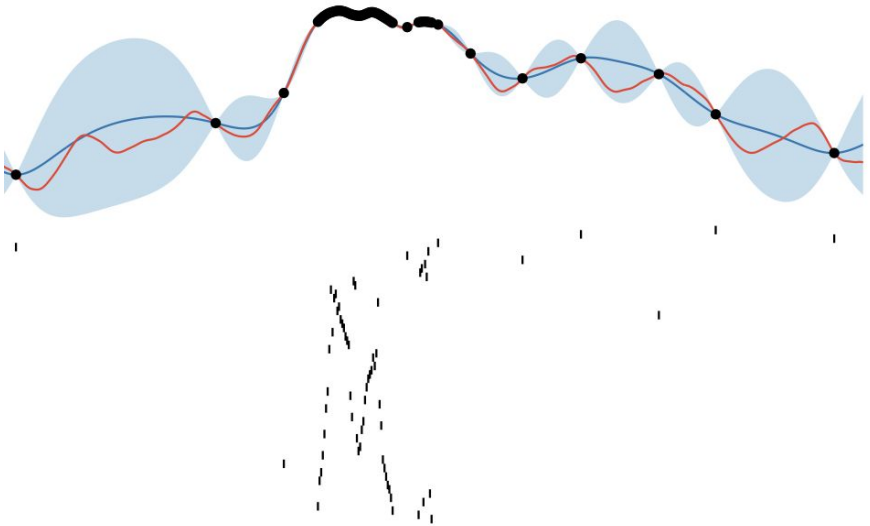
- discrete set of “arms”
- each with reward distribution (typically independent)
- seek adaptive policy to maximize the *cumulative reward*, $\Sigma_i y_i$
- **strong theoretical results**



# Infinite-armed bandits

Can model optimization as an “infinite-armed” bandit and adapt policies for MABs, hoping to inherit their theoretical guarantees.

- **infinite** set of “arms”
- each with reward distribution (**correlated!**)
- seek adaptive policy to maximize the *cumulative reward*, $\mathbf{\Sigma}_i\, y_i$
- **strong theoretical results**

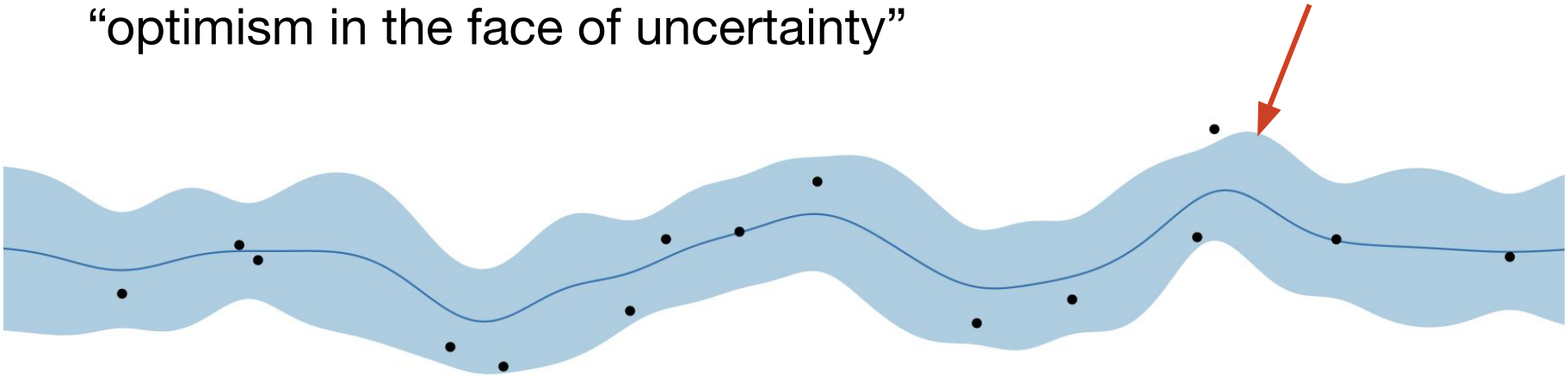


# Bandit-inspired policies

- *Upper confidence bound* (UCB): maximize statistical upper bound of objective, e.g.

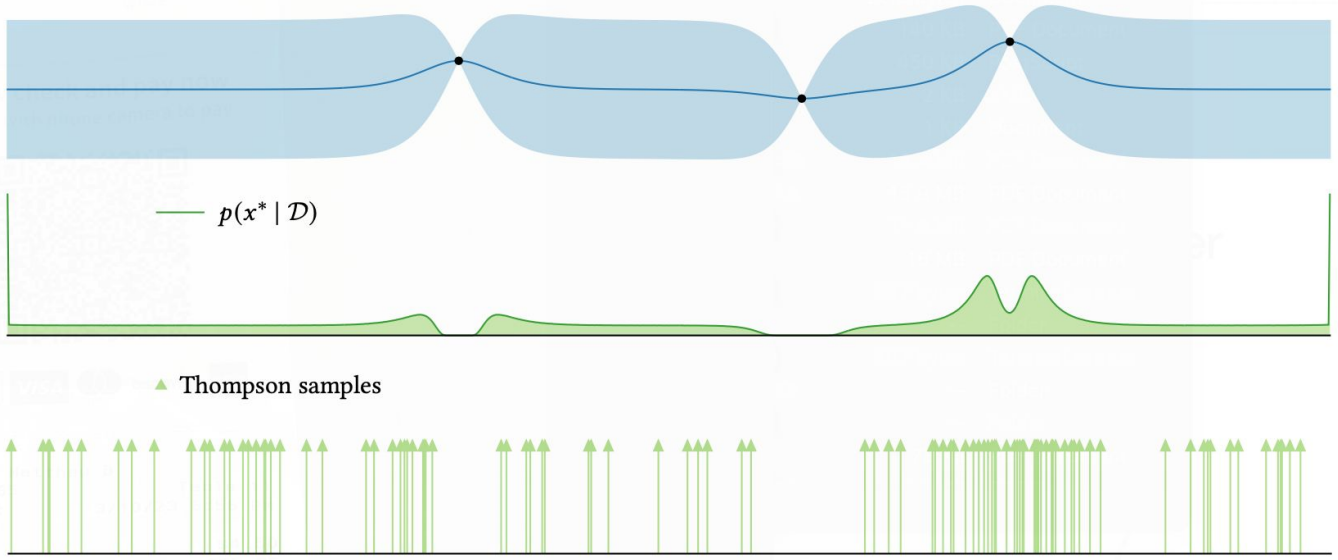
$$\mu + \beta\sigma$$

"optimism in the face of uncertainty"



# Bandit-inspired policies

- *Upper confidence bound* (UCB)
- *Thompson sampling*: sample optimum $x^*$ from posterior




both policies enjoy strong theoretical guarantees!

# Theory sketch: Goal

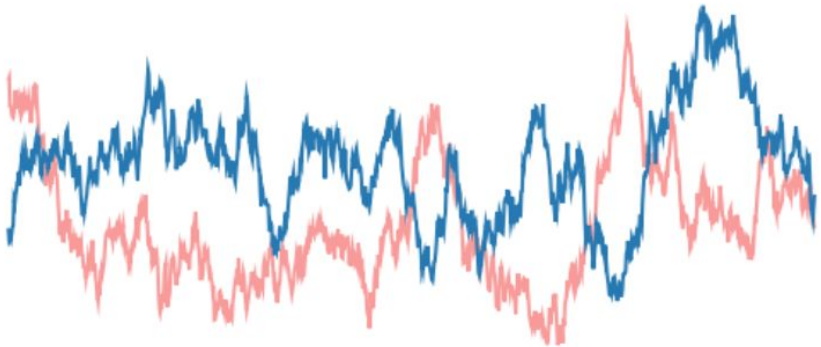
Typically seek to bound the asymptotic growth of the *cumulative regret*

$$\Sigma_i f^* - y_i$$

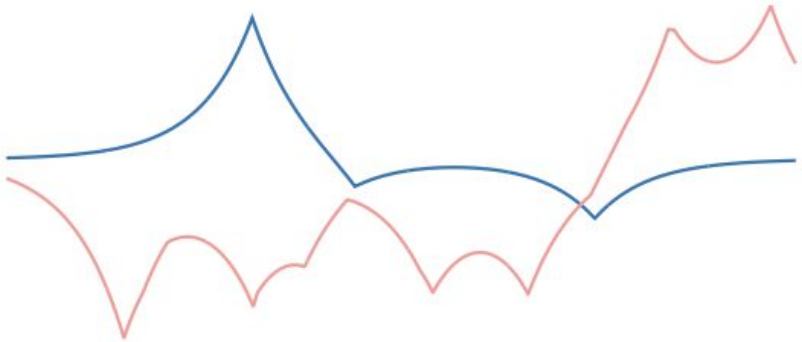
If this grows sublinearly, all observations are eventually effectively near optimal, and thus we converge to $f^*$ at some rate.

# Theory sketch: Regularity

In order to proceed, we first to establish some regularity conditions on the objective to ensure feasibility.



GP sample path

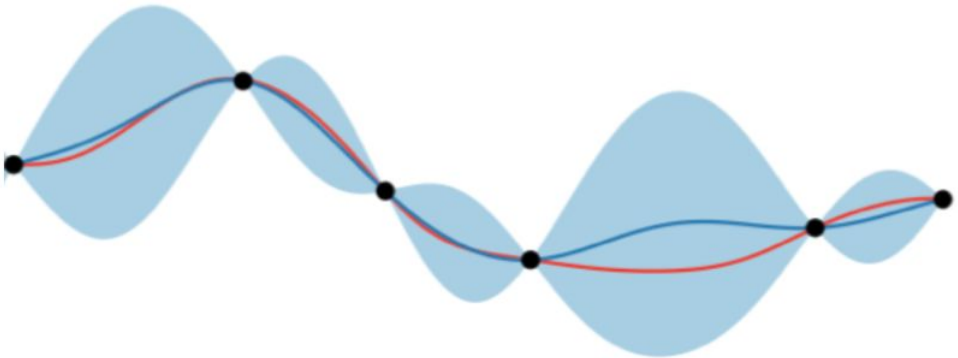


element of RKHS

# Theory sketch: Strategy

A typical strategy is to establish some *concentration* property for the posterior. For example, that we can find a sequence $\{\beta_t\}$ such that the objective function is (whp) always within the "confidence region"

$$\mu_t \pm \beta_t \sigma_t.$$

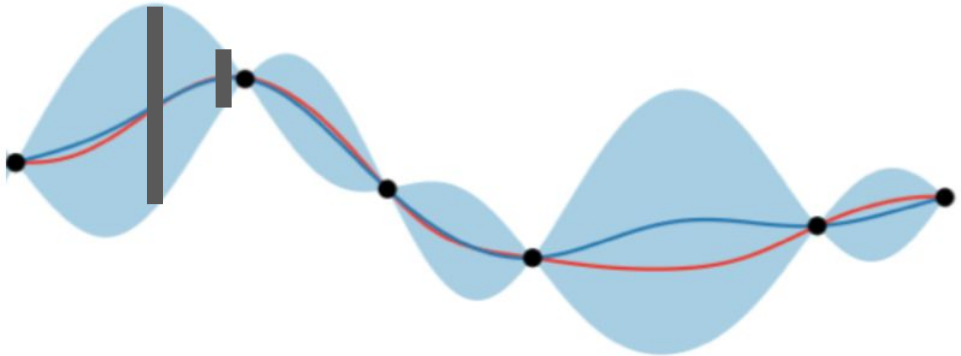


With such a guarantee, analyzing UCB is easy!

# Theory sketch

A typical strategy is to establish some *concentration* property for the posterior. For example, that we can find a sequence $\{\beta_t\}$ such that the objective function is (whp) always within the "confidence region"

$$\mu_t + \beta_t \sigma_t.$$



Optimum always lies in confidence region for selected point! (Why?)

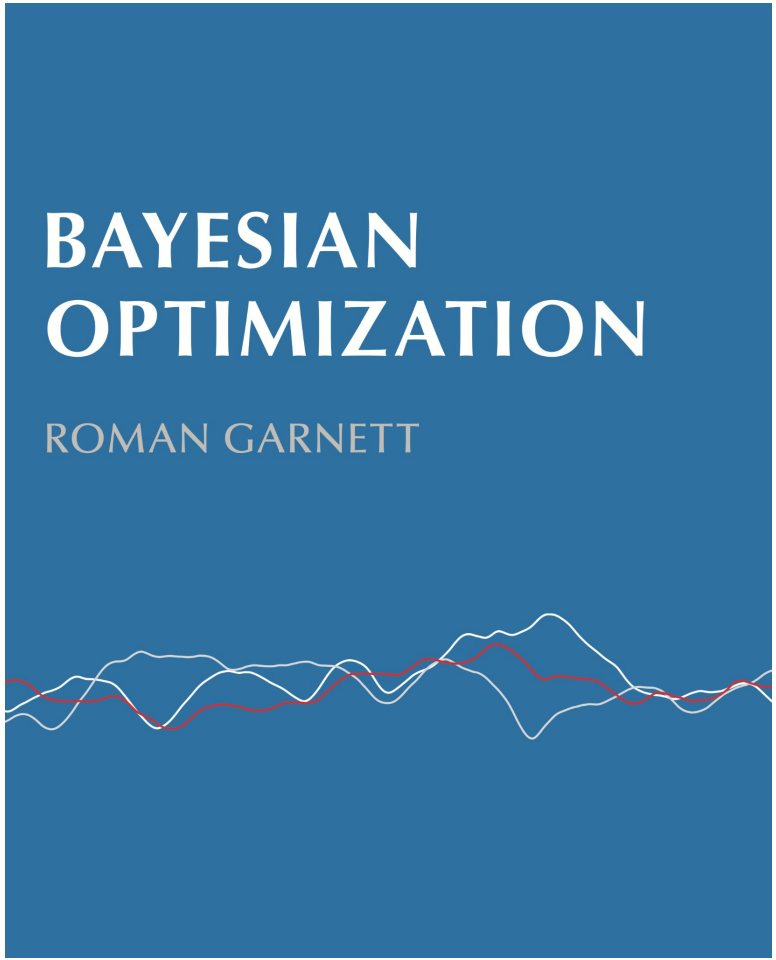
Thus regret is bounded by sum of widths.

# Information capacity

- for GPs (with Gaussian observation noise), we can proceed by bounding the so-called *information capacity* of the process


- useful bounds are known for a rich family of kernels (Matérn family/squared exponential)

- can use these to derive strong asymptotic bounds for UCB and Thompson sampling, in both the expected and worst-case




$\{\beta_t\}$

gory details in chapter 10...

# Active search

[BayesOpt Book Chapter 11]

# Active search

If the design space is an ocean punctuated by “islands” with desirable properties, then:

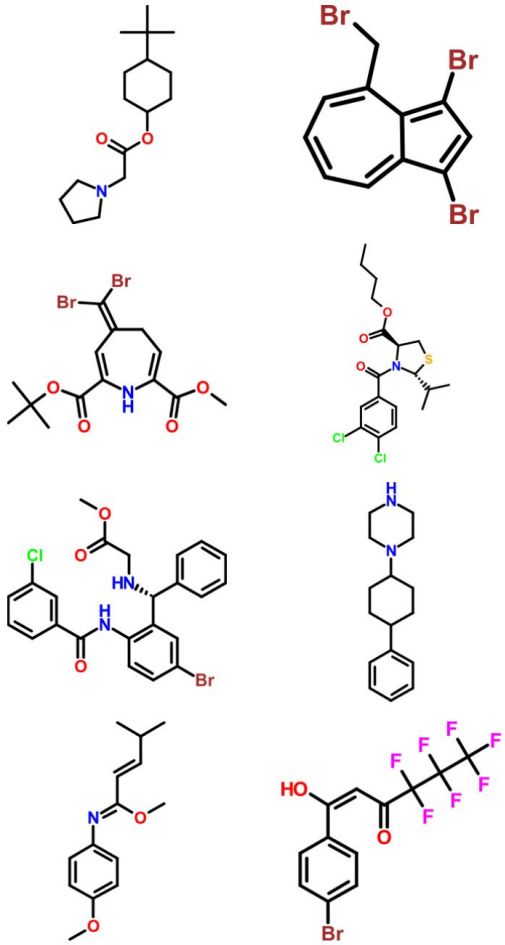
- Bayesian optimization seeks the *highest peak,* whereas
- active search seeks to *map out all the islands.*

Useful for automating *discovery*.

# Active search

Imagine that we have a binary condition indicating “success” of an experiment. For example, in drug discovery, we might say “exhibits sufficient binding activity to warrant future investigation.”

The goal of active search is to find as many “hits” as possible in a given budget.



# Active search

Specifically, imagine that we have a binary condition indicating “success” of an experiment. For example, in drug discovery, we might say “exhibits sufficient binding activity to warrant future investigation.”

The goal of active search is to find as many “hits” as possible in a given budget. Real example: Battleship.

# Obvious policy

There’s a very intuitive, perhaps even obvious policy for this problem:

- train a classifier predicting hits
- evaluate the point with the highest probability of being a hit.

# Active search utility

For active search we can define a simple utility function that simply counts the number of hits:

utility = number of discoveries in data

# One-step lookahead

For this utility, one-step lookahead becomes easy:

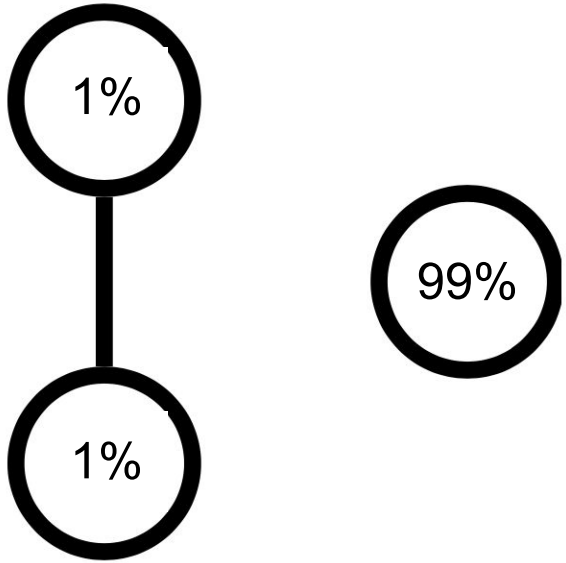
optimal decision = most-likely hit

Recovering and reinterpreting the obvious policy!

# The power of nonmyopia

However, for active search, it turns out that looking farther ahead can help a great deal.

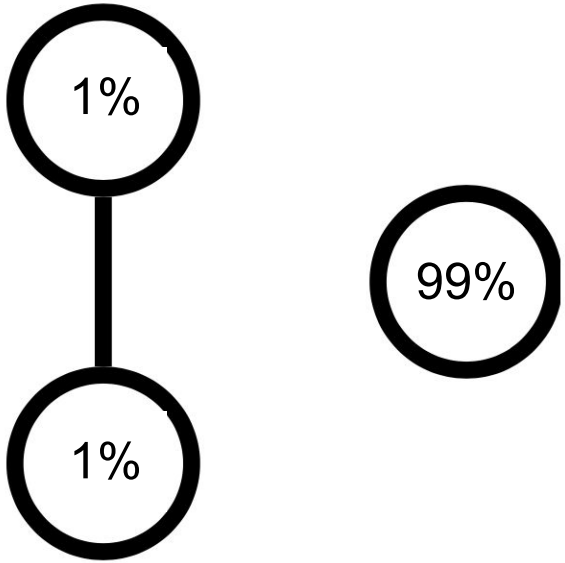
For this example, one-step lookahead would choose the point on the right...



# The power of nonmyopia

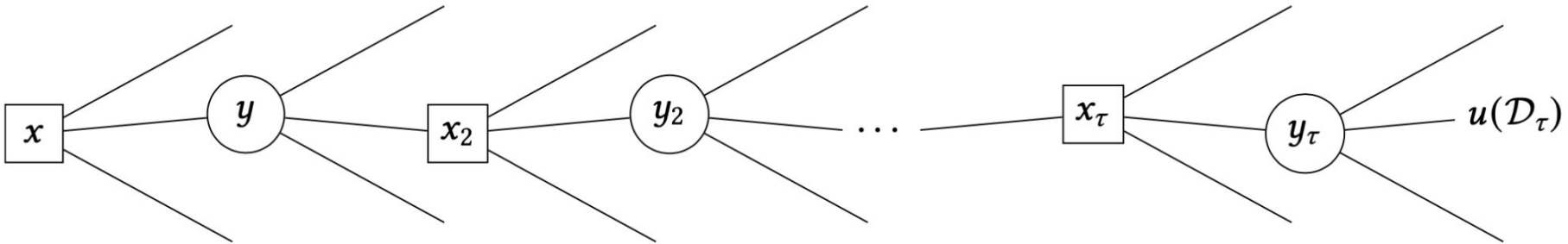
...but if we have two experiments remaining the optimal choice is a point on the left! So the *least* likely hit can be optimal, even in seemingly simple situations!

It turns out that further lookahead can help by any arbitrary amount.



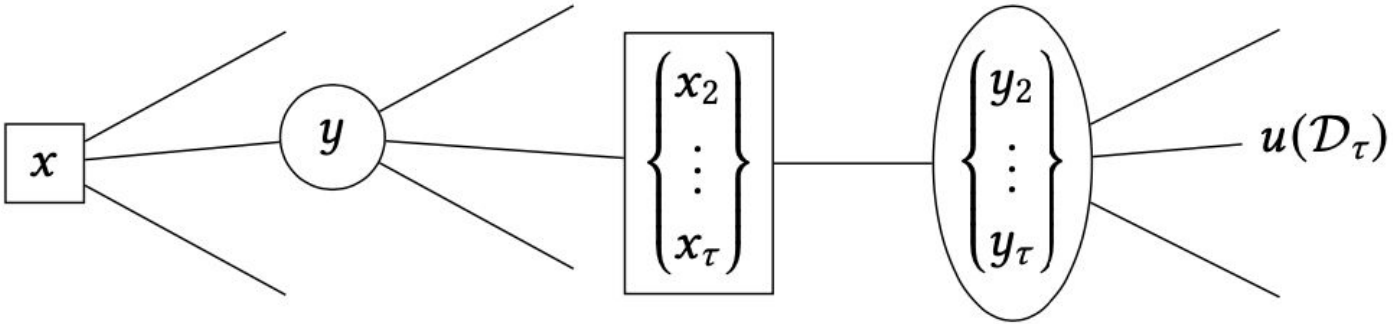
# Nonmyopic active search

We can gain huge boosts by seeking nonmyopic (but efficient) approximations to the optimal policy! Some helpful structure here due to binary observations and linear reward function.



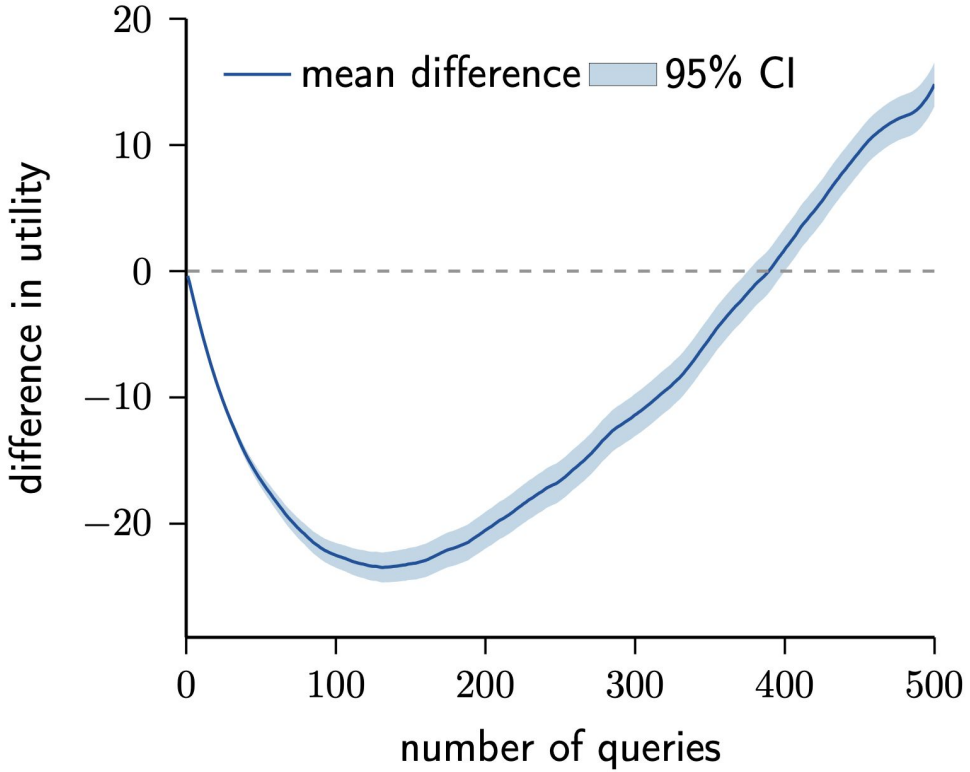
# Nonmyopic active search

One promising idea is to simulate all remaining experiments as if they were all designed *simultaneously* in one big batch. This can be done efficiently while introducing awareness of the remaining budget into the policy. A similar idea has been used in Bayesian optimization as well (GLASSES).



Nonmyopic policy shows very interesting behavior from *budget consciousness!*

Automatic shift from exploration to exploitation!



# Theoretical guarantees

**Bayesian optimization**

**Active search**

Very strong guarantees:
under mild assumptions,
efficient policies are close to
*optimal*, asymptotically as
#experiments $\longrightarrow \infty$.

# Theoretical guarantees

**Bayesian optimization**

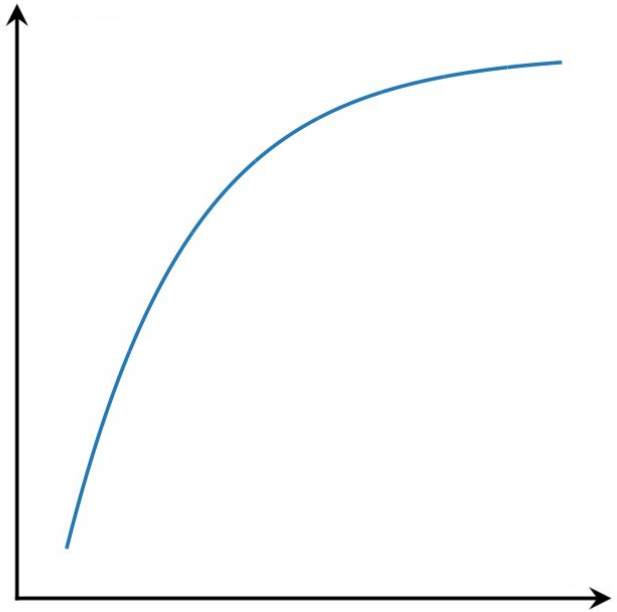
Very strong guarantees: under mild assumptions, efficient policies are close to *optimal*, asymptotically as #experiments $\longrightarrow \infty$.

**Active search**

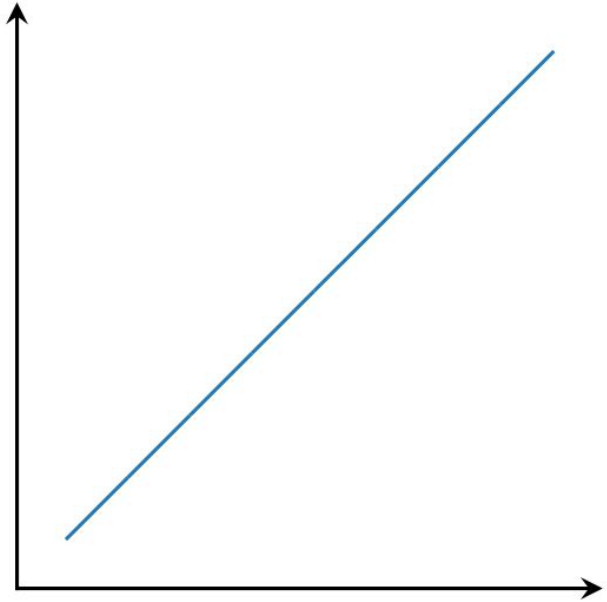
Very strong anti-guarantees: *no* efficient policy can be close to *optimal* in the worst case, no matter what crazy ideas you might want to try.

There is a natural property of *diminishing returns* in Bayesian optimization as you progress closer to the optimum. But the utility could scale *linearly* in active search for a long time.

**Bayesian optimization**



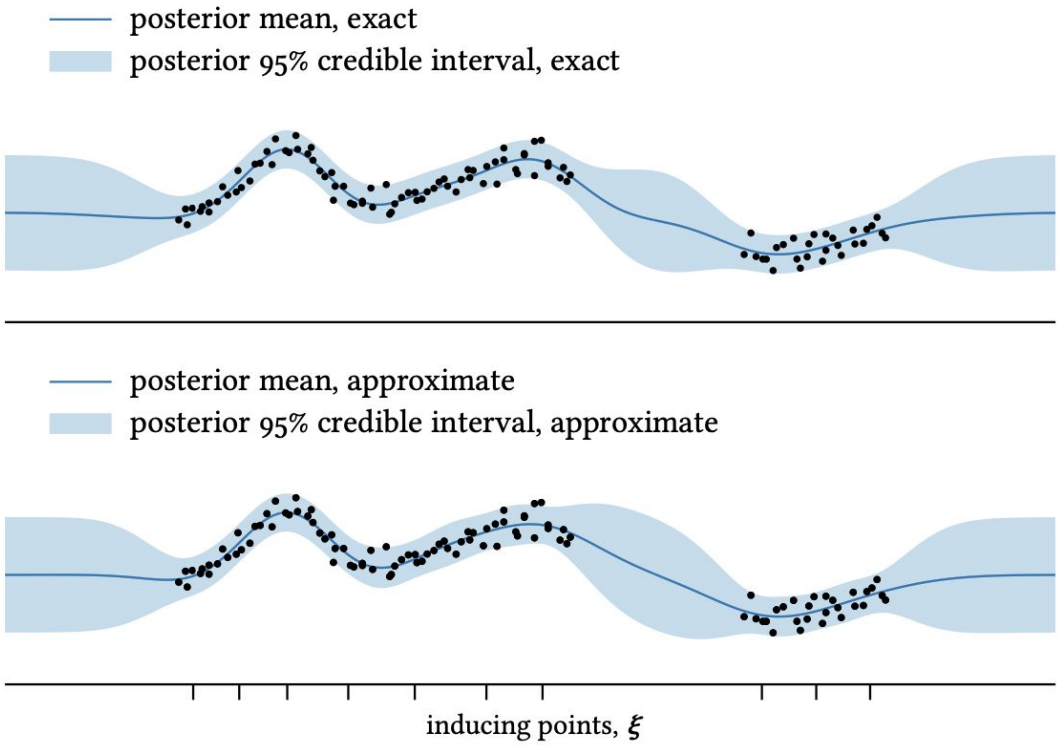
**Active search**



# BayesOpt and PN

Is this even PN?

A naïve implementation of GP inference scales cubically with the number of observations, but various schemes (sparse approximation, iterative numerical methods) can scale to millions of observations.



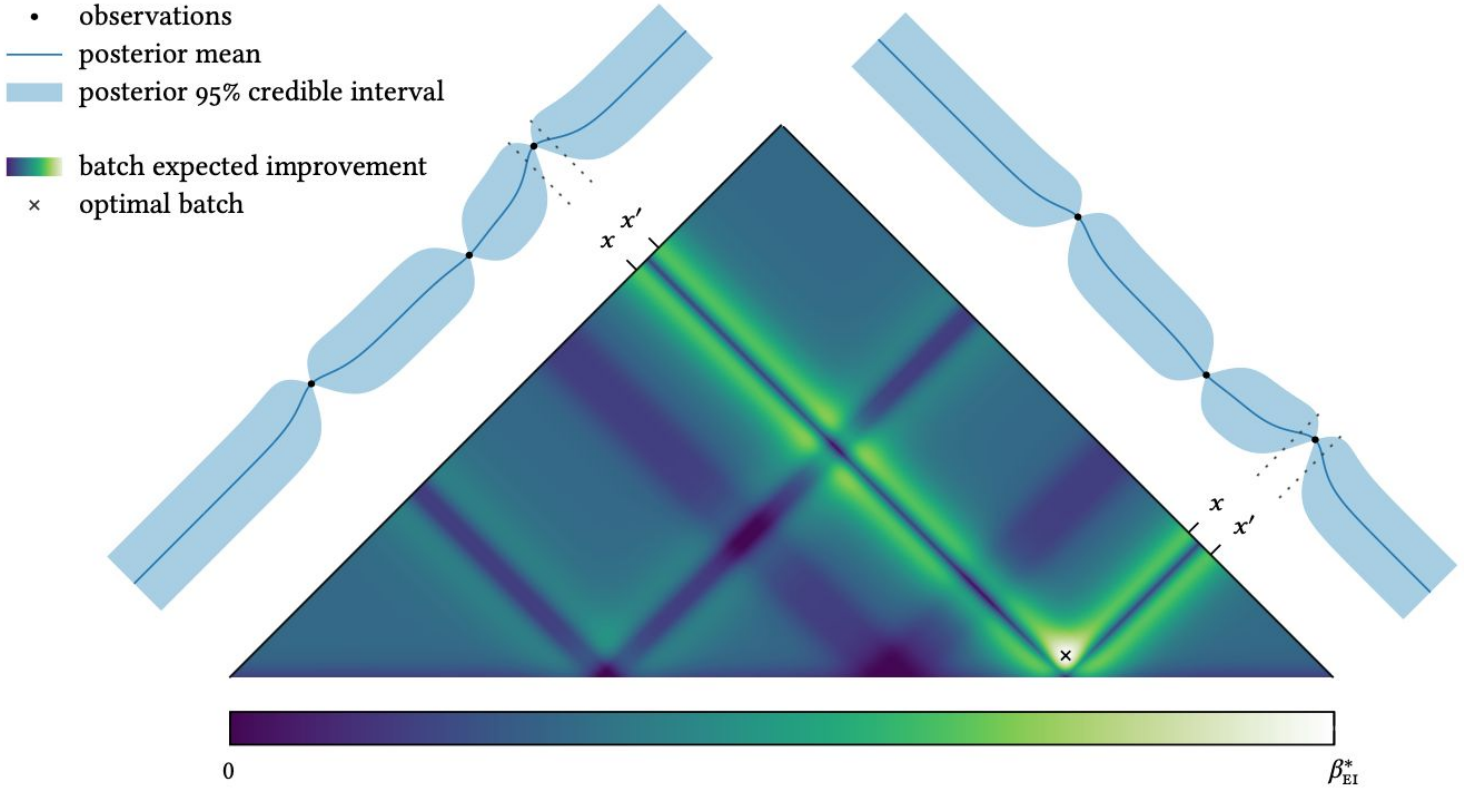
# Local optimization

Also closer to the spirit of “mainline PN,” there has been considerable effort in deriving Bayesian analogues of classical local optimization strategies such as line search and trust-region methods.

- For line searches, can use e.g. GP surrogates for determining optimal step sizes and/or search directions (enabling “zeroth-order” local optimization)
- For trust-treason methods, can replace e.g. quadratic surrogate with a GP (see TuRBO)

# Extensions

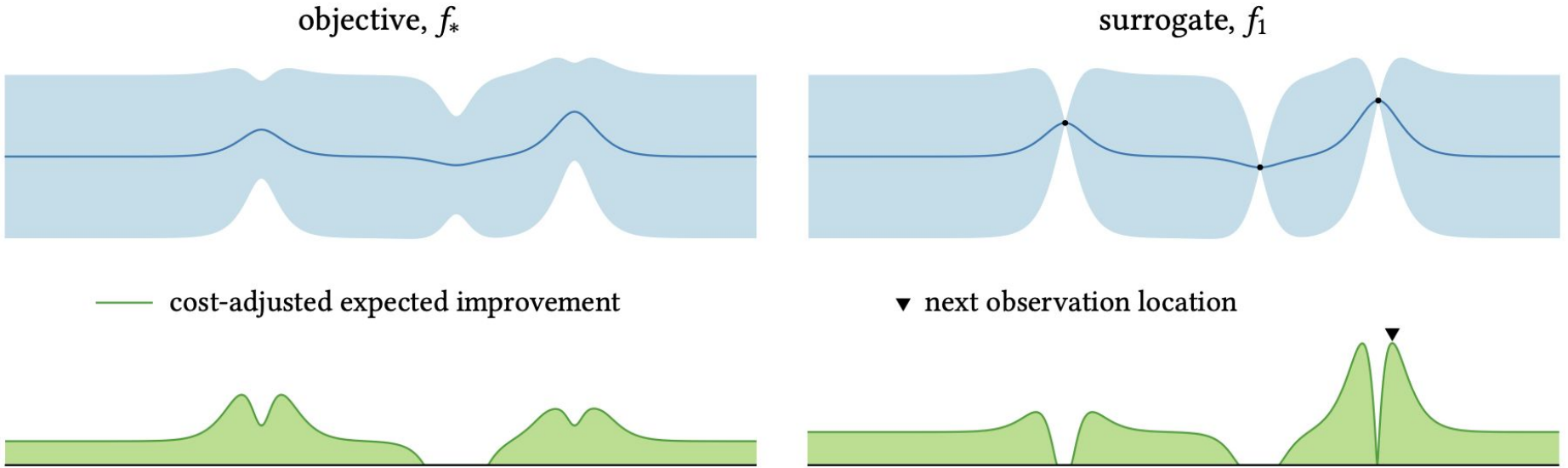
[BayesOpt Book Chapter 11]




Intelligent policies are available for ***batch observations*** (microwell plates, parallel simulation, etc.) trading off exploration, exploitation, and *diversity.*

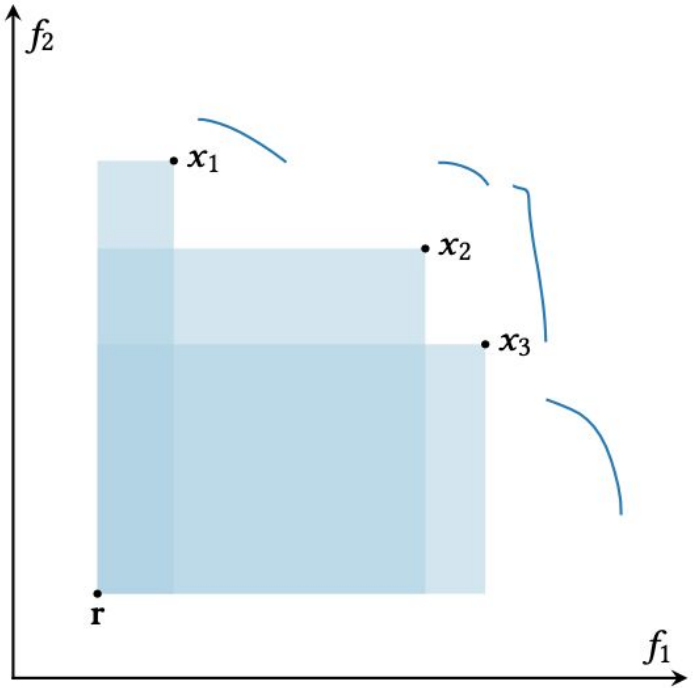
# Multifidelity optimization

In some cases, we may be able to observe inexpensive *surrogates* of the objective function – DFT, molecular dynamics, computational docking, etc. Policies are available to guide the use of surrogates.



# Multiobjective optimization

Intelligent policies are available to drive the simultaneous optimization of *multiple objectives* (or tradeoffs between them) by actively mapping out the Pareto frontier.



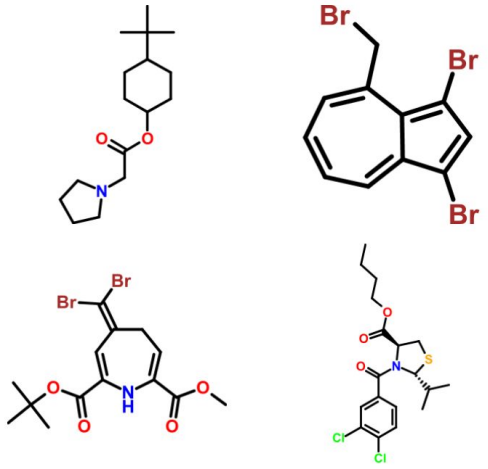
# Applications

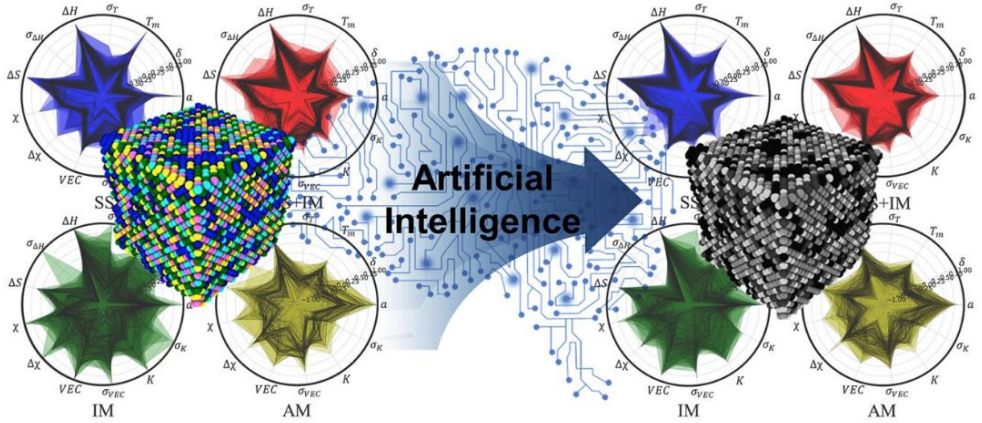
[BayesOpt Book Appendix D]

# Chemistry / MatSci

BayesOpt has seen widespread adoption especially in light of the rise of computational tools (DFT, molecular dynamics, etc.)

- molecular design
- optimization of material properties
- reaction optimization



# Applied chemistry

## Bayesian Optimization for a Better Dessert

**Greg Kochanski, Daniel Golovin, John Karro, Benjamin Solnik,**
**Subhodeep Moitra, and D. Sculley**
{gpk, dgg, karro, bsolnik, smoitra, dsculley}@google.com; Google Brain Team

**Abstract**

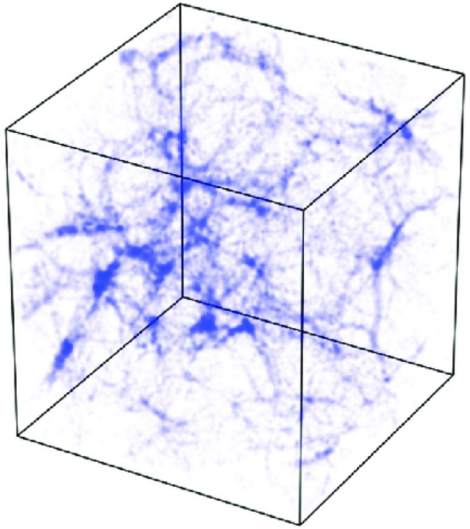
We present a case study on applying Bayesian Optimization to a complex real-world system; our challenge was to optimize chocolate chip cookies. The process was a mixed-initiative system where both human chefs, human raters, and a machine optimizer participated in 144 experiments. This process resulted in highly rated cookies that deviated from expectations in some surprising ways – much less sugar in California, and cayenne in Pittsburgh. Our experience highlights the importance of incorporating domain expertise and the value of transfer learning approaches.

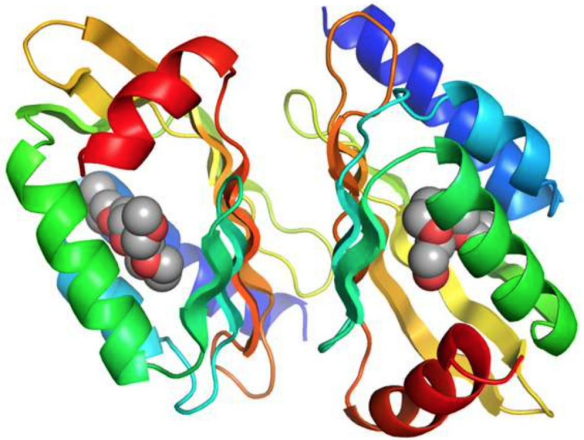
image: J Gibson

# Hoptimization?

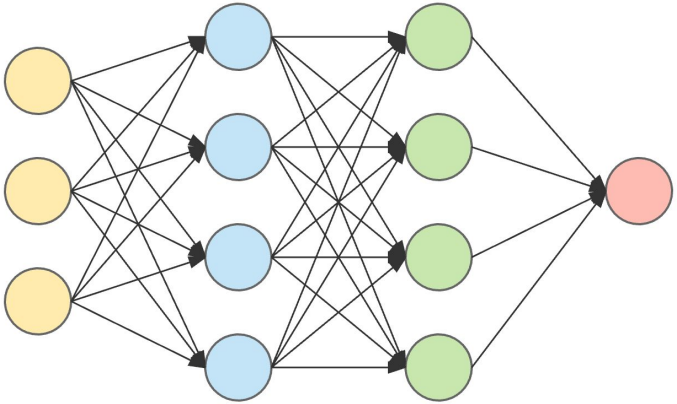
# Physics/Biology

- solving inverse problems (*n*-body simulations)
- controlling complex instruments
- gene and protein design
- adaptive experimentation
- plant breeding





# Computer science



Recent interest from the ML community has been driven by the difficulty of *hyperparameter tuning* in deep learning contexts.

Was used in AlphaGo!

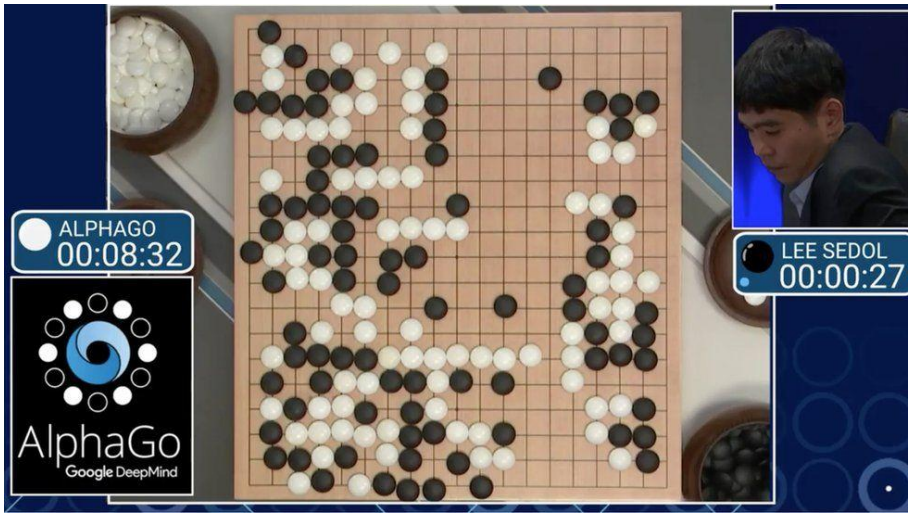
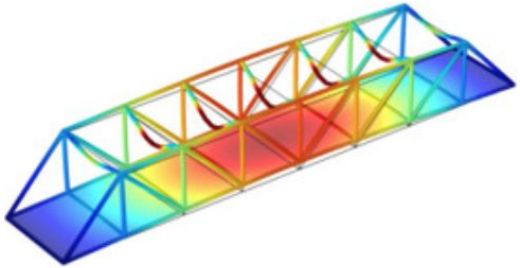


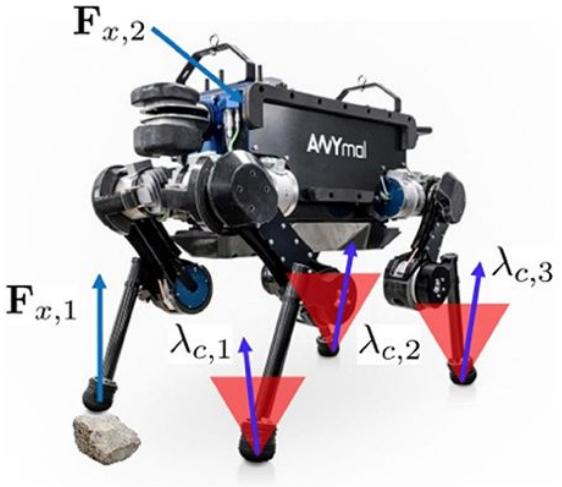

image: BBC

# Engineering



BayesOpt has proven useful in virtually every area of engineering, where large design spaces and the relatively high cost of prototyping can present challenges.

- robotics
- airfoil design (CFD)
- mechanical and structural engineering, etc.




images: Comsol, Xin, et al. (2020)

# Challenges for the future

# Collaboration!

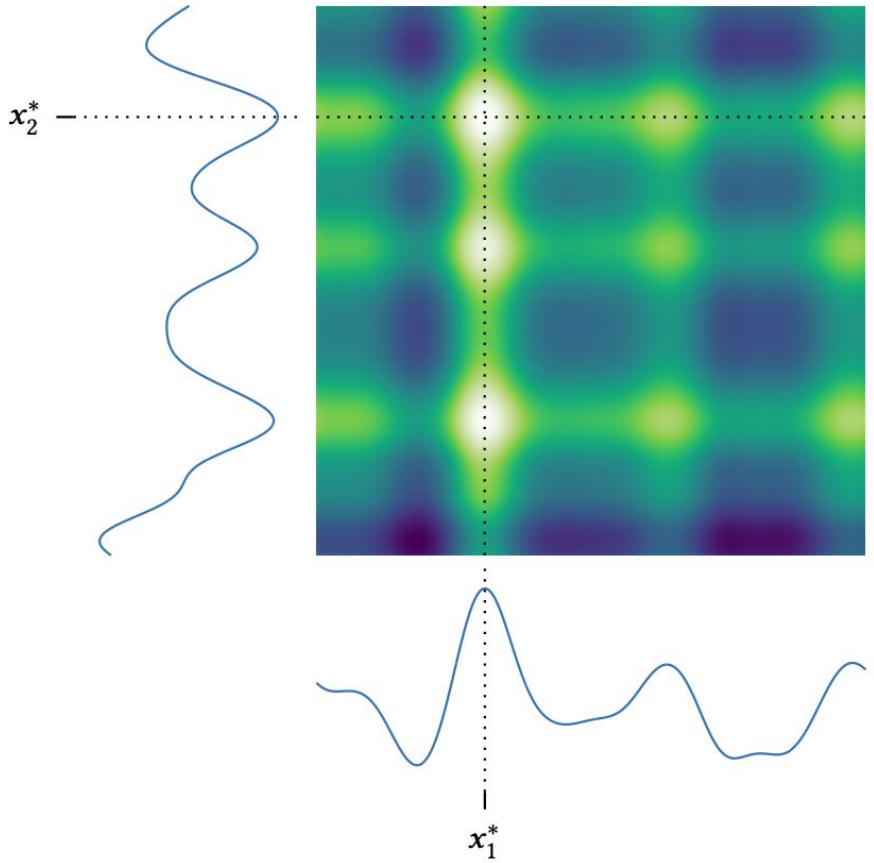
Every time I collaborate on a new application, there is always some interesting “twist” I did not anticipate!

*High-dimensional*

Bayesian optimization remains a challenge.

Better modeling?

Local optimization?



Can we build “expert-in-the-loop” systems for human–computer *collaborative* design / discovery?

# Thank you!