Grey-Box Bayesian Optimization

Peter I. Frazier
Cornell University
Uber


Raúl Astudillo

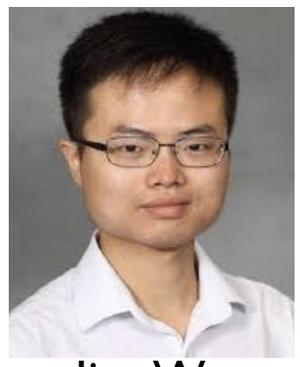

Scott Clark


Matthias Poloczek


Saul Toscano-Palmerin

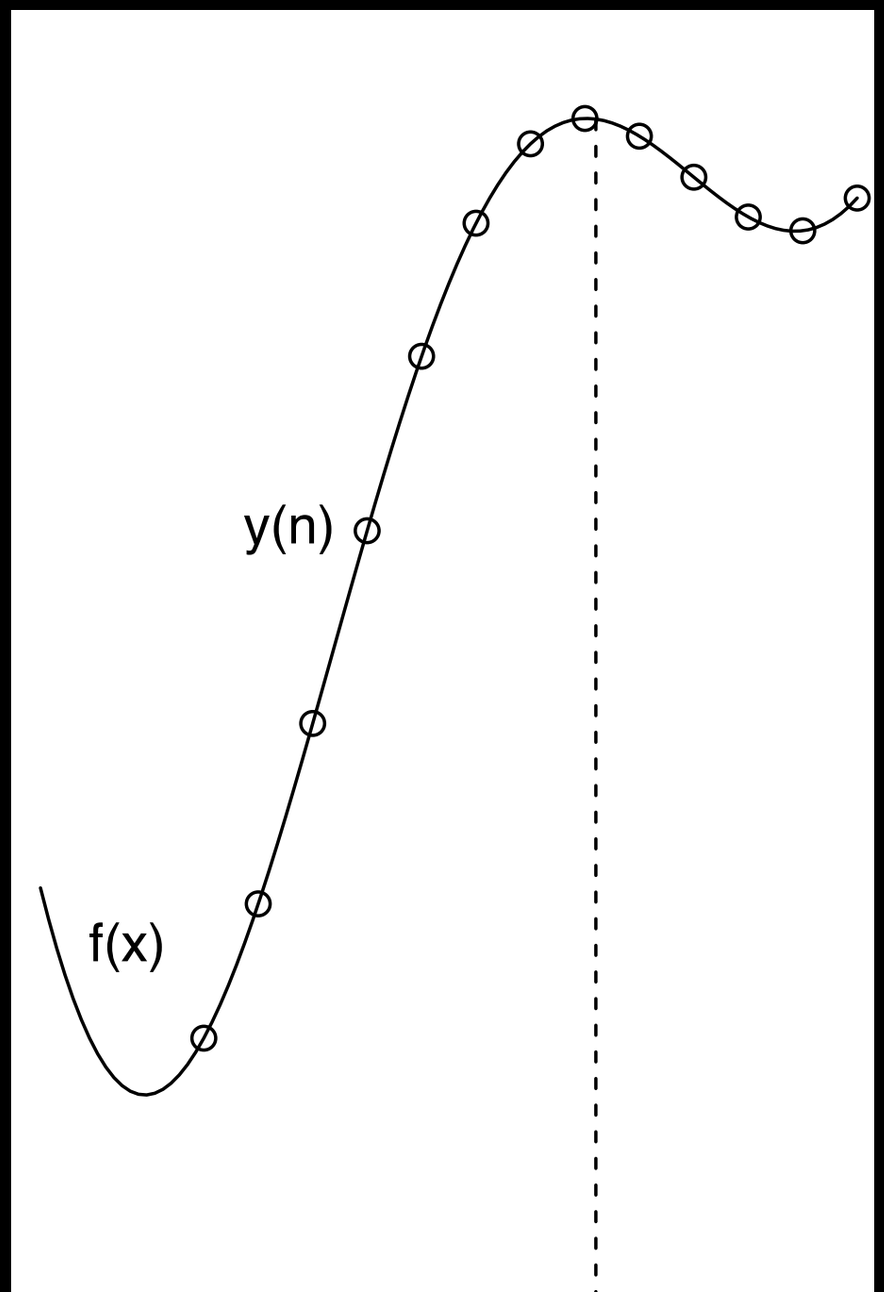

Jialei Wang


Andrew Wilson


Jian Wu

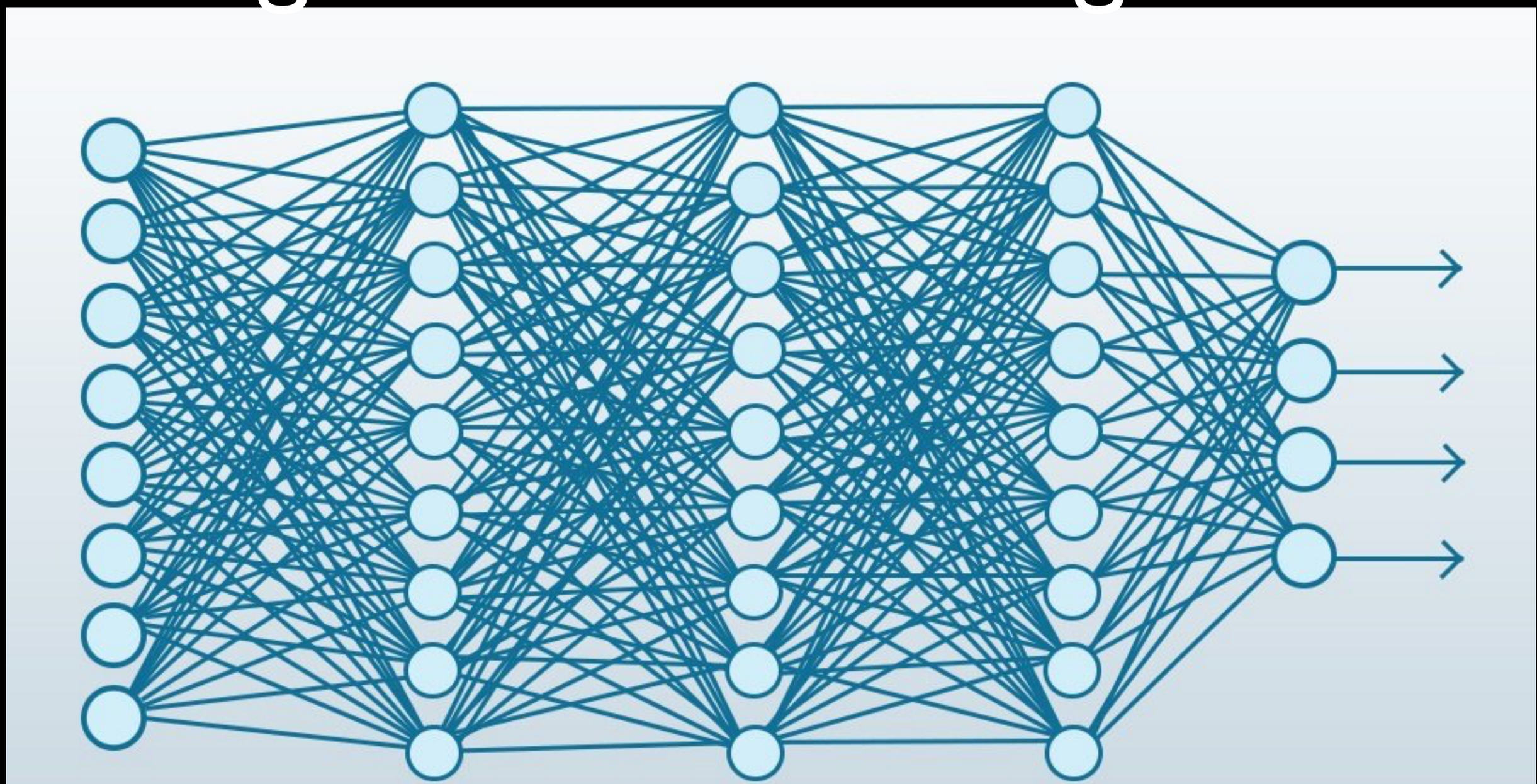
Funded by AFOSR and NSF

Bayesian Optimization traditionally considers this black-box optimization problem



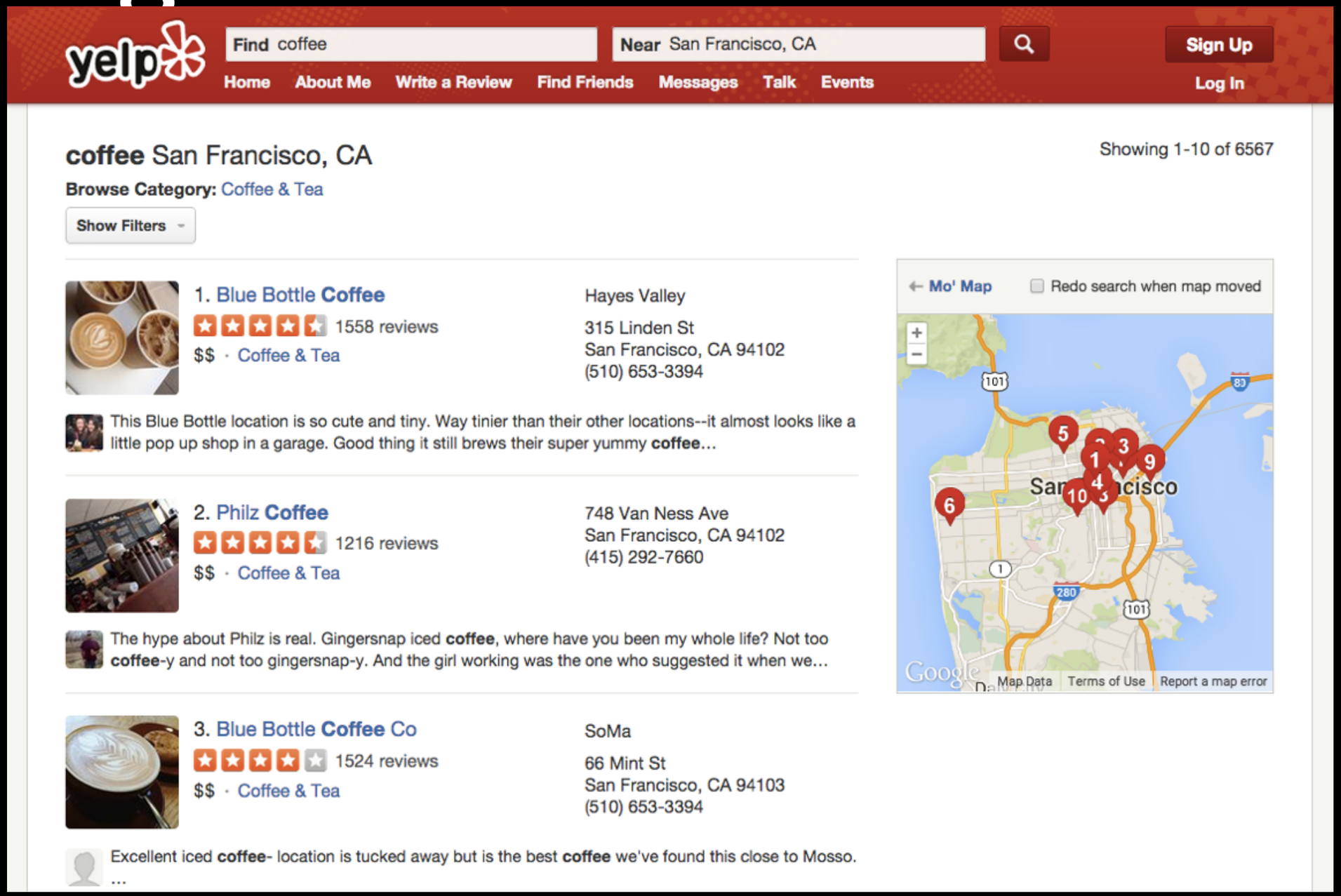
- We'd like to optimize $F : \mathbb{R}^d \rightarrow \mathbb{R}$, where $d < 20$.
- F's feasible set A is simple, e.g., box constraints.
- F is continuous but lacks special structure, e.g., concavity, that would make it easy to optimize.
- F is derivative-free: evaluations do not give gradient information.
- F is expensive to evaluate: the # of times we can evaluate it is severely limited.
- F may be noisy. If noise is present, we'll assume it is independent and normally distributed, with common but unknown variance.

Bayesian Optimization is useful for fitting machine learning models

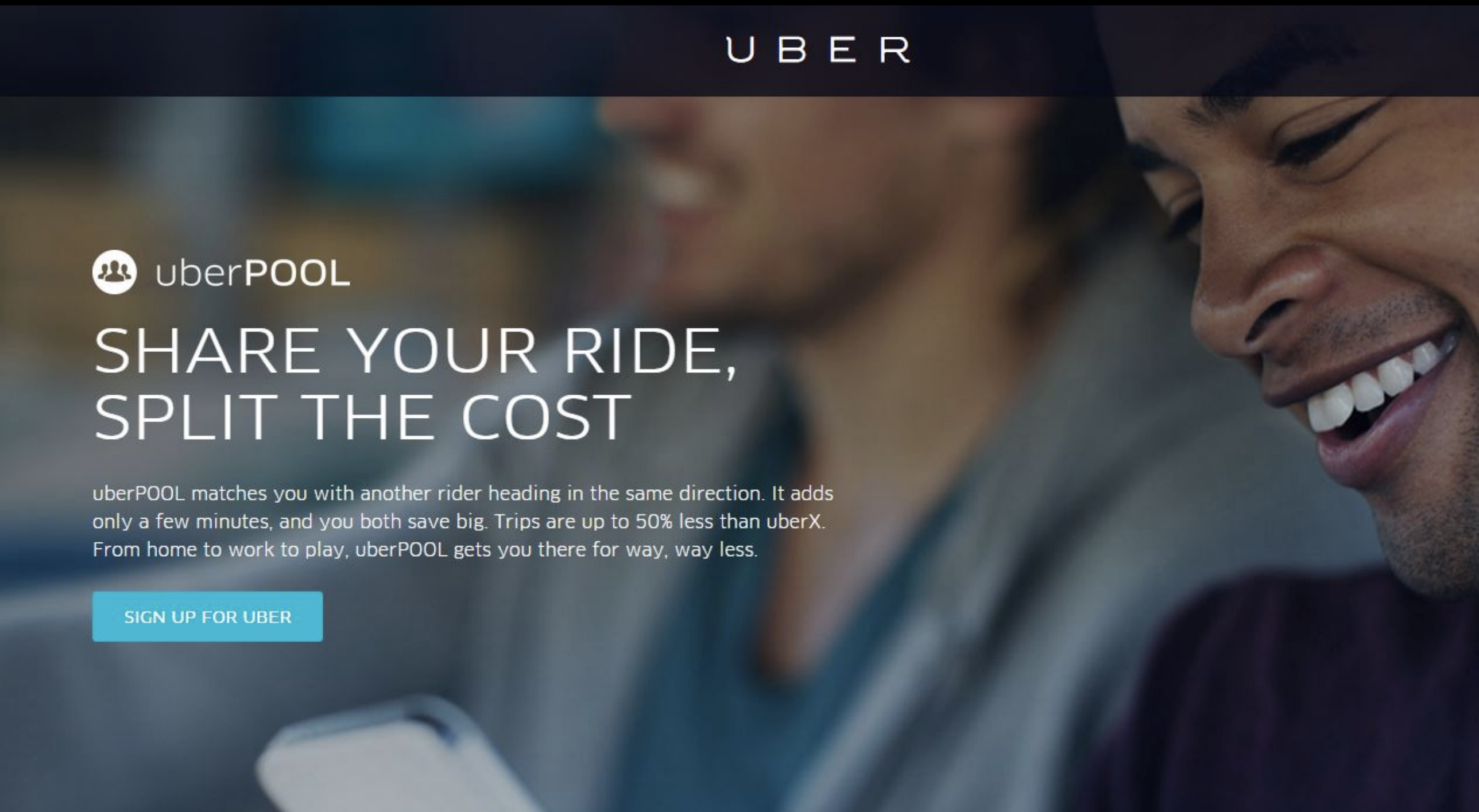


Bayesian Optimization is useful for tuning algorithms via backtesting

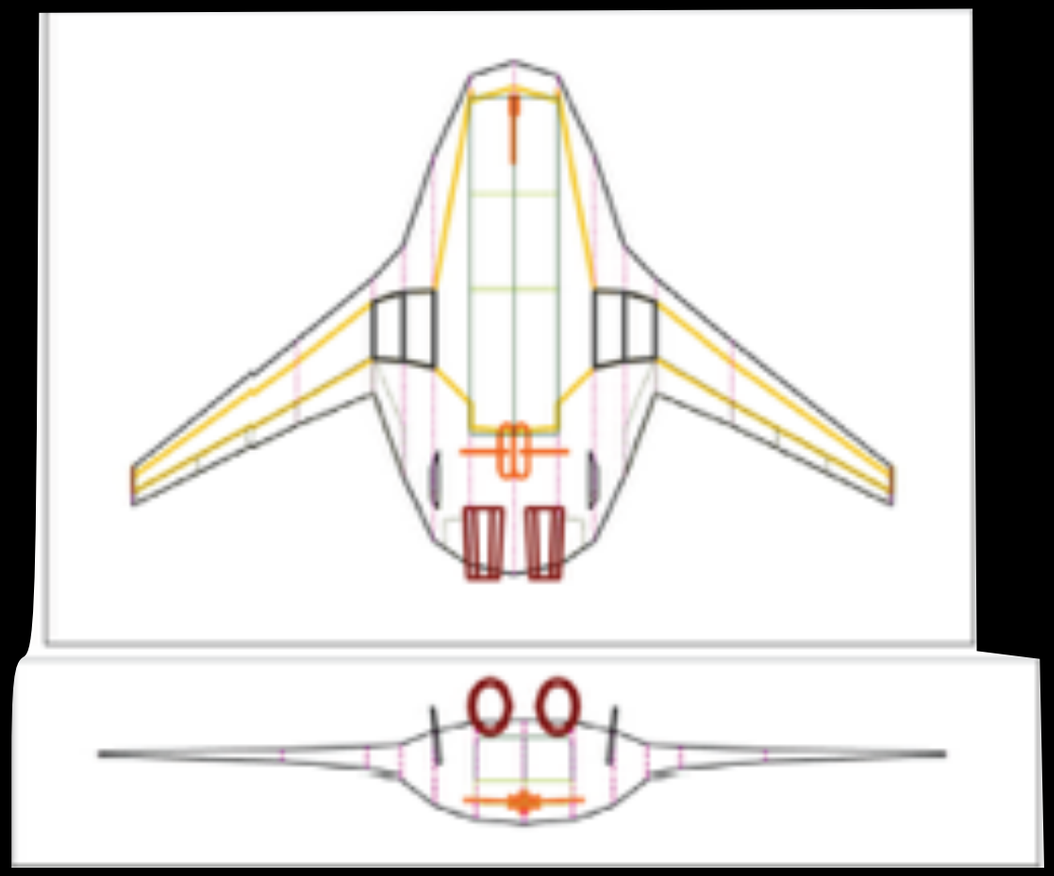
Bayesian Optimization is useful for tuning websites with A/B testing

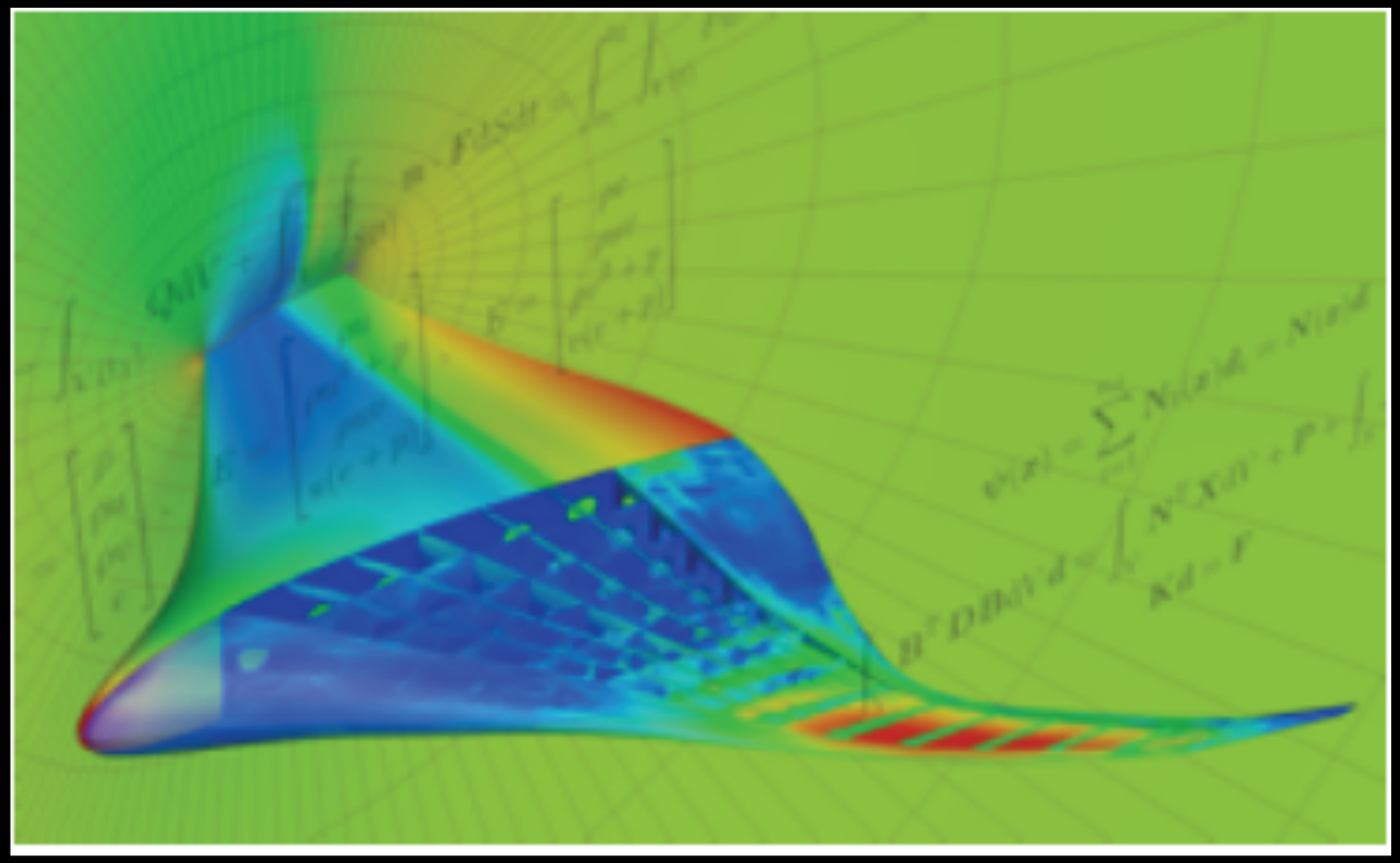


Bayesian Optimization is useful for tuning transportation markets



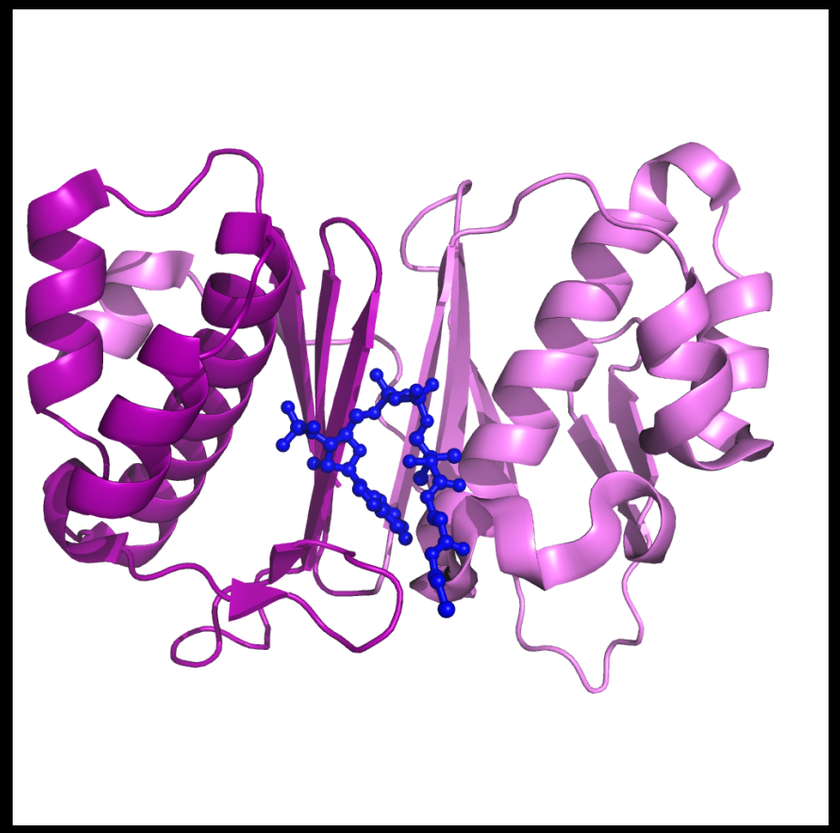
Bayesian Optimization is useful for optimizing physics-based models



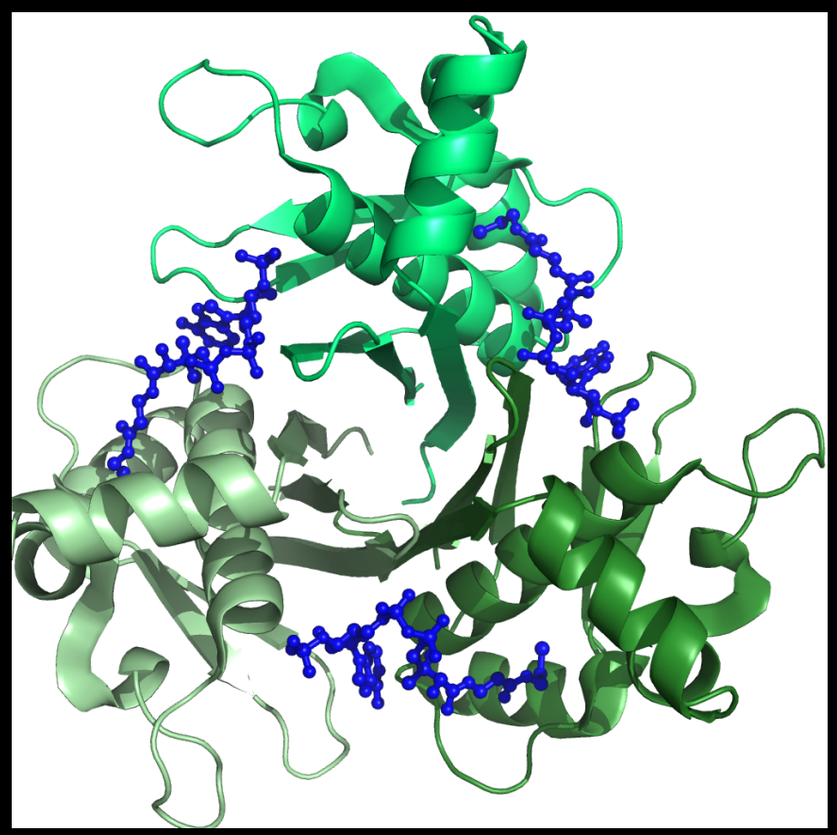


ongoing work with Karen Wilcox (MIT), Joaquim Martins (Michigan), Doug Allaire (Texas A&M), David Wolpert (Sante Fe Institute), Youssef Marzouk (MIT), and Marc Mignolet (Arizona State University),

Bayesian Optimization is useful for drug and materials discovery



Sfp
(a protein-modifying enzyme)



AcpS
(another protein-modifying enzyme)

ongoing work with Mike Burkart and Nathan Gianneschi, UCSD

There are other derivative-free ways to optimize black-box functions

- There are many derivative-free optimization methods.
  - Bayesian optimization, Surrogate-based methods, Trust region methods, Pattern search methods, Evolutionary algorithms, …
- Bayesian optimization works well when:
  - we have prior information
  - the number of evaluations we can do is **extremely** limited:
    e.g., 5 evaluations for a 3 dimensional problem.

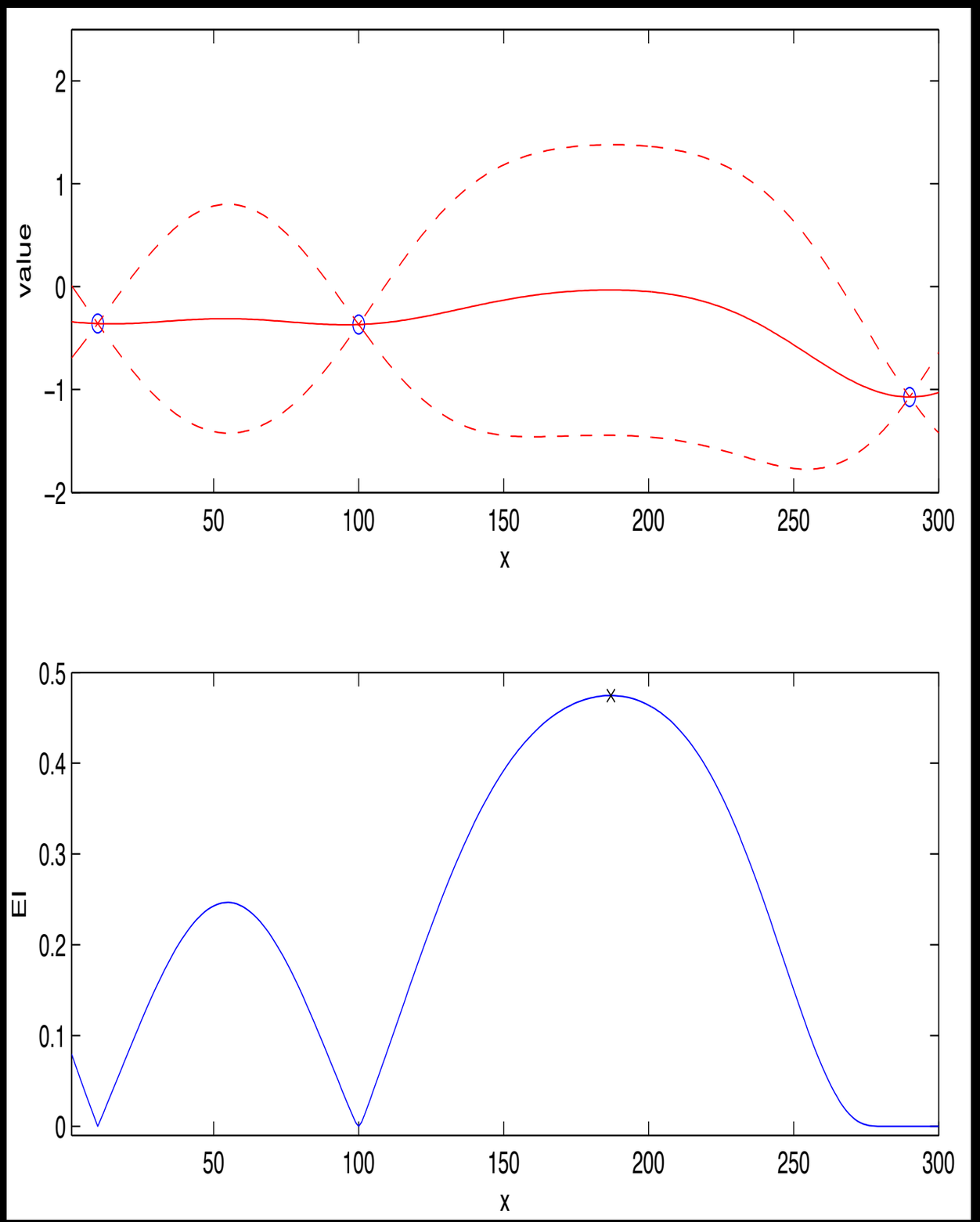
Here's how Bayesian Optimization works, at a high level

Choose an appropriate Bayesian prior on f

How? Typically we:
(1) assume a Gaussian process prior whose kernel depends on parameters
(2) sample f at a few randomly selected points
(3) use the samples to choose the parameters of the kernel

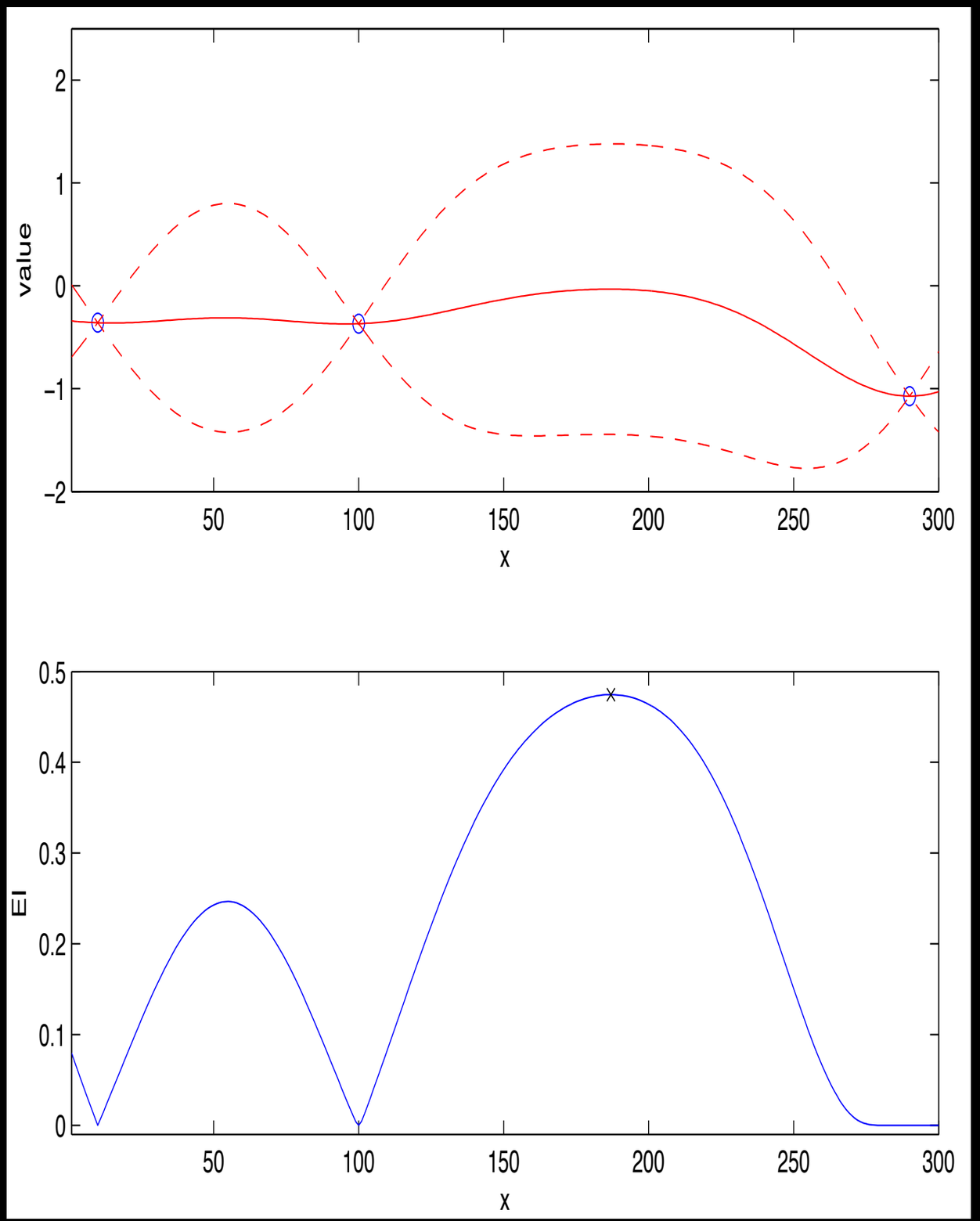
```
while (budget is not exhausted) {
        Find x that maximizes acquisition(x,posterior)
        Sample x & observe F(x)
        Update the posterior distribution on F
      }
```

Let me tell you about a **classical** Bayesian optimization method:
Efficient Global Optimization (EGO) [Jones, Schonlau & Welch 1998]



- We've evaluated $x^{(1)},\ldots x^{(n)}$, & observed $f(x^{(1)}),\ldots,f(x^{(n)})$ without noise.
- The best value observed is $f^*=\max(f(x^{(1)}),\ldots,f(x^{(n)}))$.
- If we evaluate at x, we observe f(x).
- The *improvement* is $\{f(x)-f^*\}^+$
- The *expected improvement* is $EI(x)=E[\{f(x)-f^*\}^+]$

Let me tell you about a **classical** Bayesian optimization method:
Efficient Global Optimization (EGO) [Jones, Schonlau & Welch 1998]

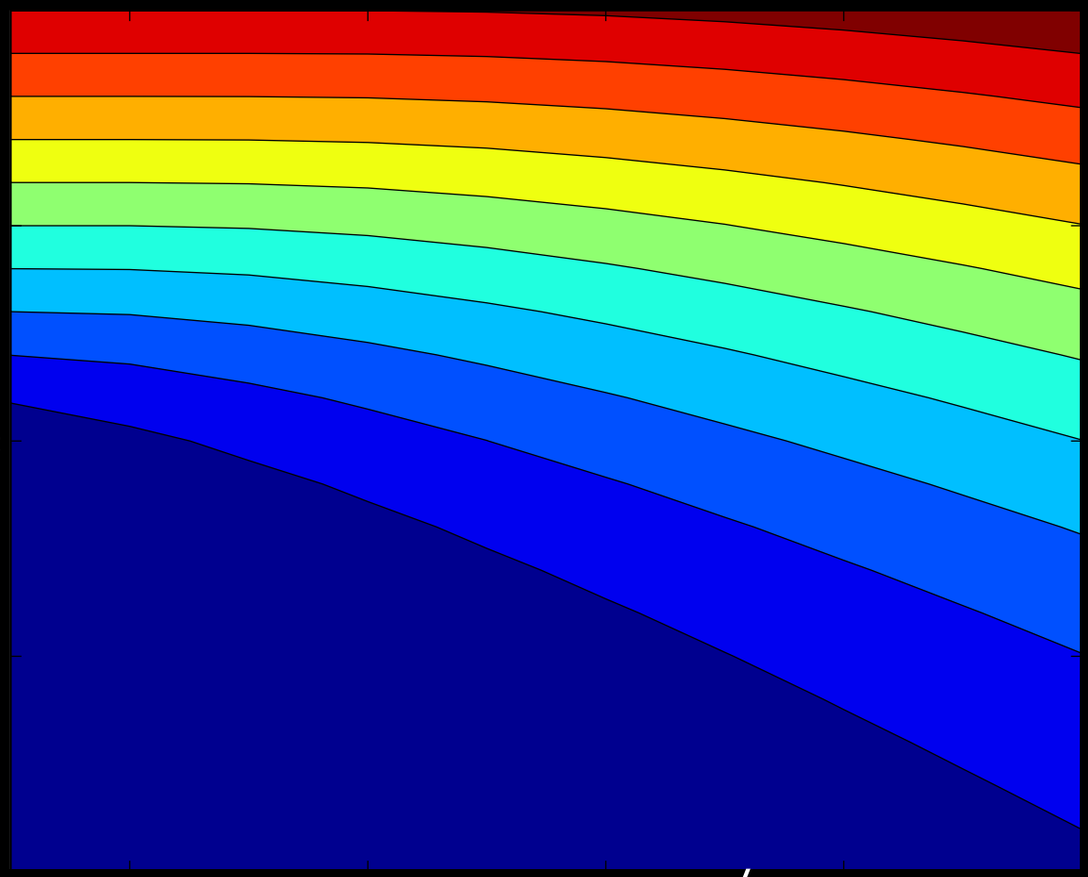


- The *expected improvement* is $EI(x)=E[\{f(x)-f^*\}^+]$
- We evaluate at the point with the largest EI.
- This is **optimal** (with respect to expected reward under the posterior) **if**:

1. this is our last evaluation
2. our solution can only be a previously evaluated point

Expected improvement has an analytic expression that trades exploration vs. exploitation

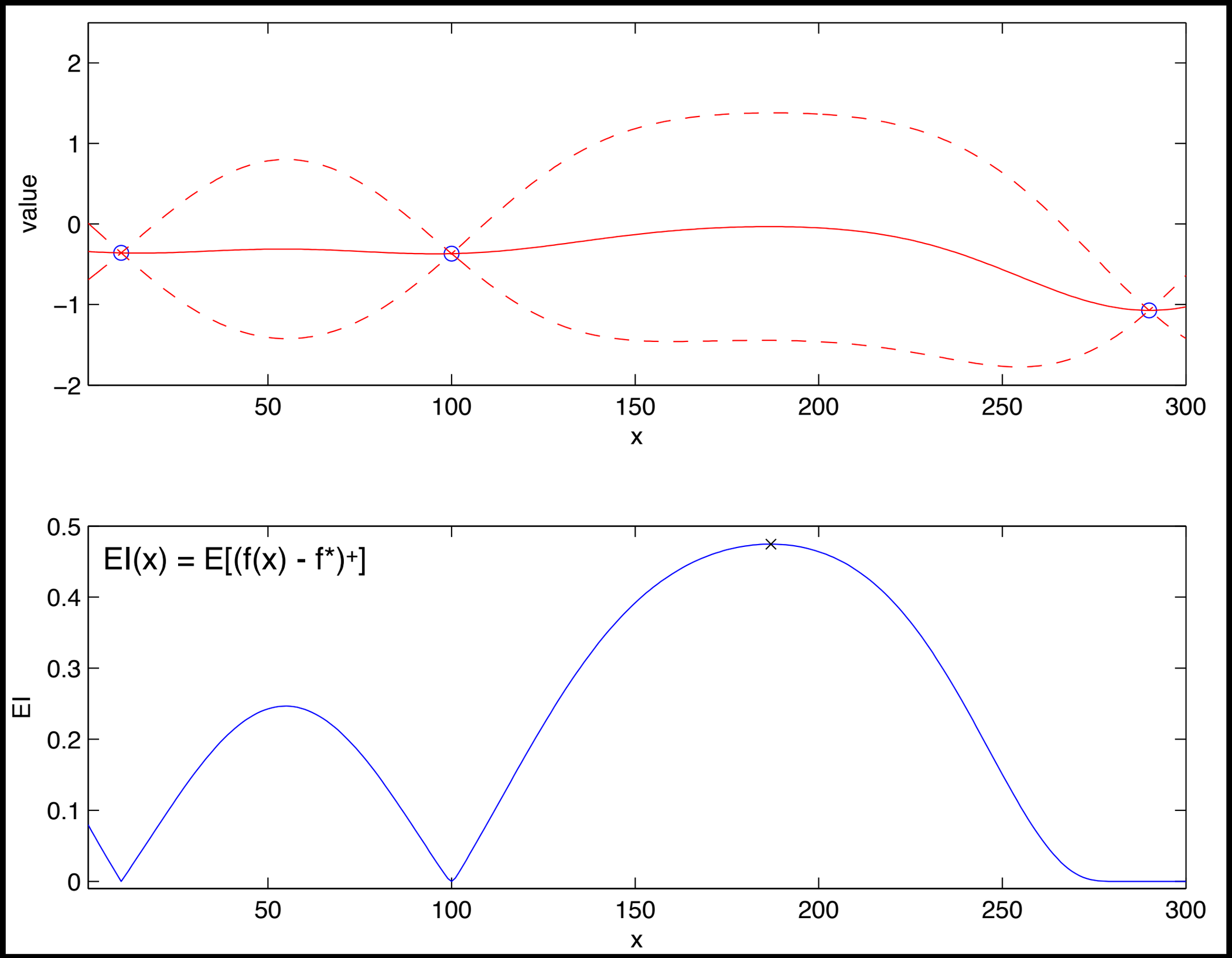
Estimated quality,
$\Delta(x) = E[f(x)]-f^*$



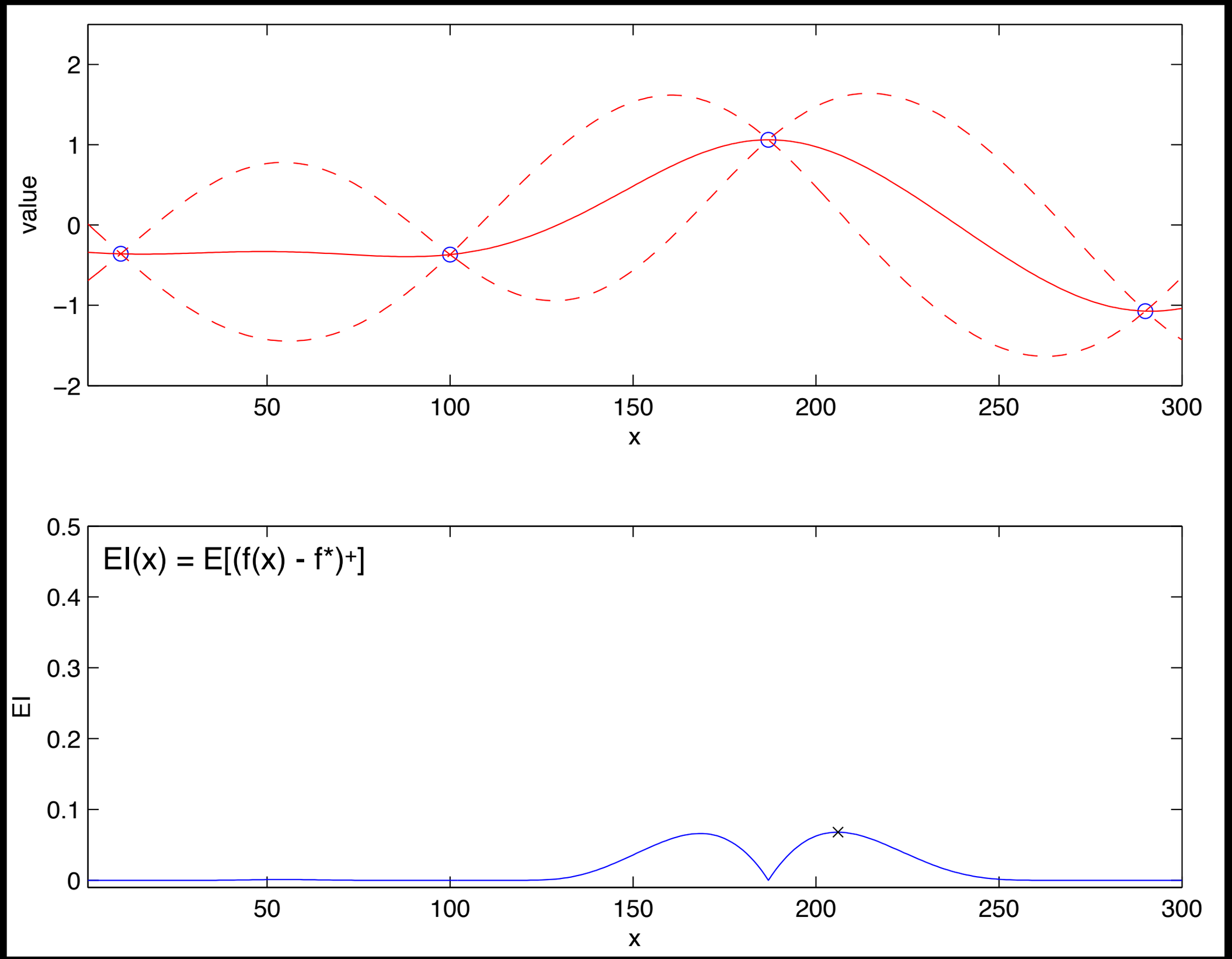
Uncertainty, $\sigma(x)=\sqrt{Var[f(x)]}$

$$EI(x) = \Delta(x)^+ + \sigma(x)\varphi(\Delta(x)/\sigma(x)) - |\Delta(x)|\Phi(-|\Delta(x)|/\sigma(x))$$

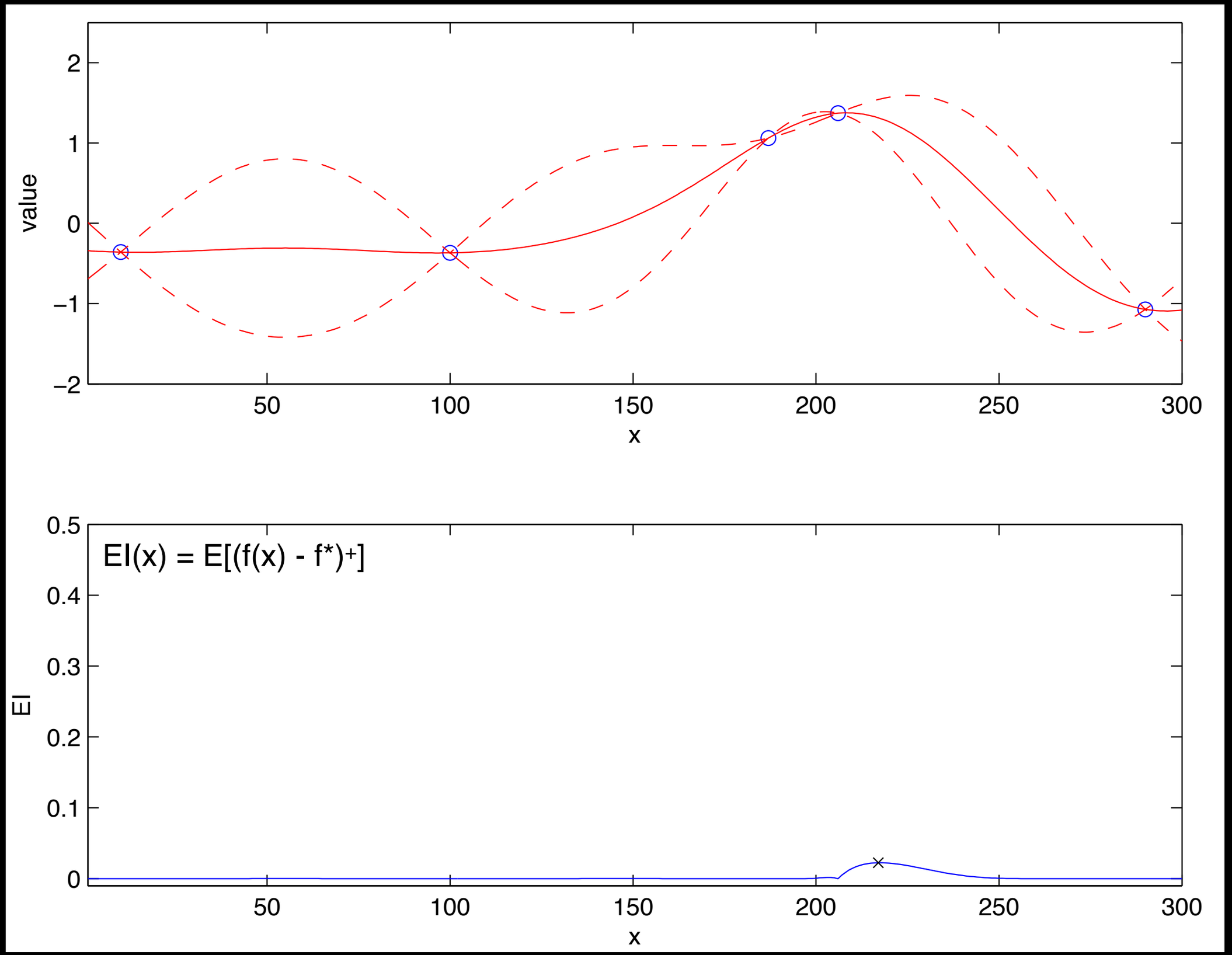
This is EGO, a **classical** Bayesian optimization method, optimizing a 1-dimensional objective



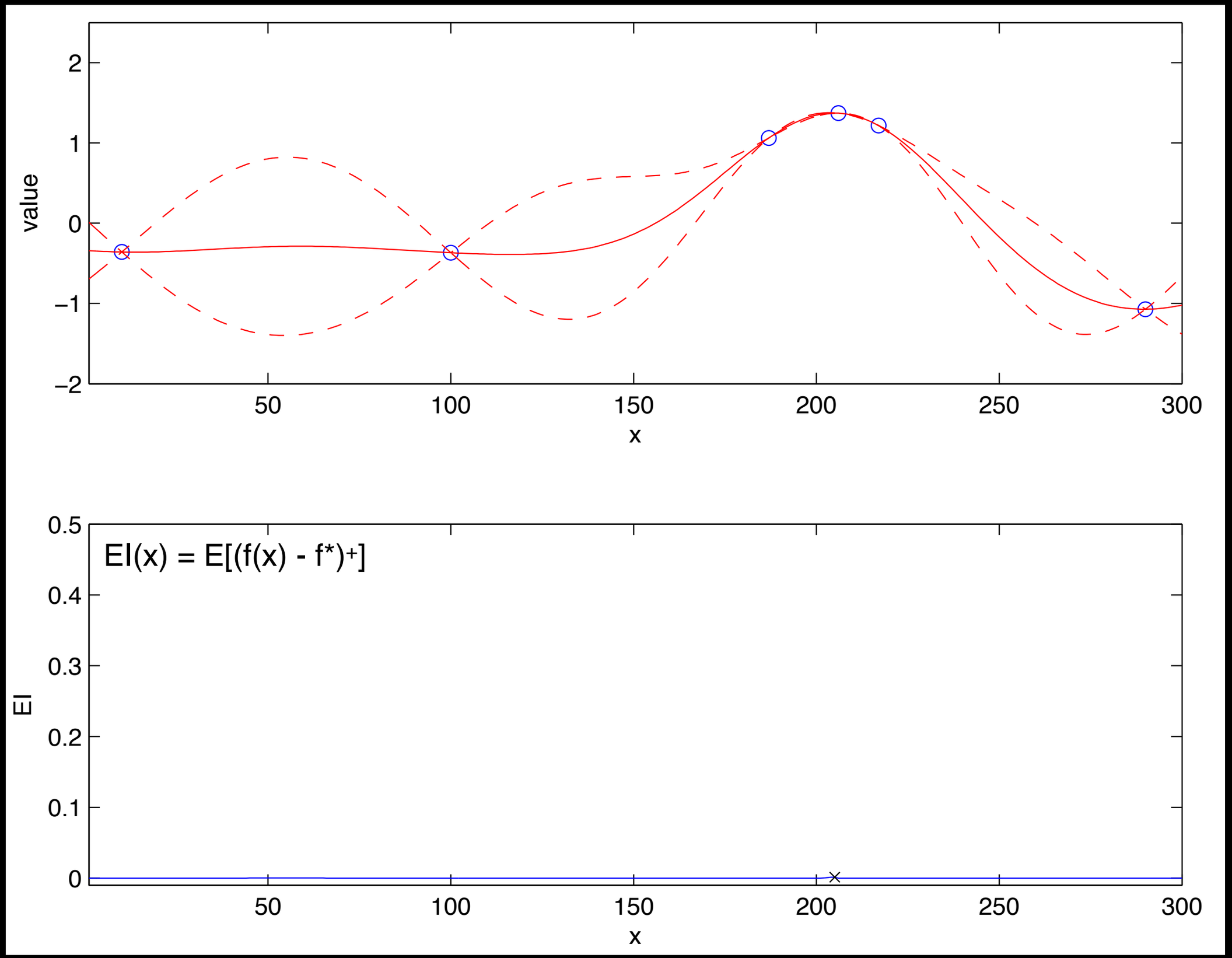
This is EGO, a **classical** Bayesian optimization method, optimizing a 1-dimensional objective



This is EGO, a **classical** Bayesian optimization method, optimizing a 1-dimensional objective



This is EGO, a **classical** Bayesian optimization method, optimizing a 1-dimensional objective



Bayesian Optimization has been very successful

Venture Capital Dispatch

An inside look from VentureWire at high-tech startups and their investors.

BANKS
Barry Silbert's DCG Slows Down Bitcoin Deals

UNCATEGORIZED
Alternative Lending 'Land Grab' Squeezes Margins

7:40 am ET
Jun 16, 2015 BIG DATA

SigOpt Raises $2 Million to 'Optimize Anything' From Jet Engines to Snacks

PREVIOUS
Kleiner Perkins Launches New Seed Fund to Lure Youth

NEXT
The Daily Startup: Foundation Rebukes VCs for Failing to Put Skin in the Game

ARTICLE | COMMENTS

ANDREESSEN HOROWITZ BUSINESS INTELLIGENCE SOFTWARE

Email Print

By LORA KOLODNY CONNECT

SigOpt founder and CEO Scott Clark. — Felicia Flee Kieselhorst

San Francisco-based startup **SigOpt Inc.** has raised $2 million for its subscription software that promises to help businesses "optimize anything and everything" more quickly than relying on traditional A/B testing or crowdsourced user testing, said cofounder and CEO Scott Clark. **Andreessen Horowitz** and **Data Collective** co-led the investment.

SigOpt's system asks for a user to enter any number of data points about a product or service, as well as parameters that describe what they're aiming to achieve with it.

The system then applies machine learning, algorithms and other sophisticated math to show the user experiments their business may want to try next to get closer to achieving

SEARCH VENTURE CAPITAL DISPATCH GO

About Venture Capital Dispatch

Produced by the editors of **Dow Jones VentureWire**, Venture Capital Dispatch tracks the fast-moving developments at the intersection of high-tech innovation and venture capital finance. Featuring the VentureWire reporting team in the Silicon Valley, New York, Boston and Shanghai tech centers, Venture Capital Dispatch provides insight into the newest start-ups and latest trends in venture capital investing. Write us at **VCdispatch@dowjones.com**. For more information on Dow Jones products covering venture capital and other financial markets, go to **http://pevc.dowjones.com**.

Follow Venture Capital Dispatch on Twitter

Like Venture Capital Dispatch on Facebook

Venture Capital Dispatch Bureau


Mike Billings
Reporter, Wall Street Journal


Lizette Chapman
Reporter, Wall Street Journal


Yuliya Chernova
Reporter, Wall Street Journal


Deborah Gage
Reporter, Wall Street Journal


Russ Garland
Reporter, Wall Street Journal


Brian Gormley
Reporter, Wall Street Journal


Timothy Hay
Reporter, Wall Street Journal


Lora Kolodny
Reporter, Wall Street Journal

Bayesian optimization is used at Uber, Facebook, Google, Yelp, Netflix, & elsewhere

Bayesian optimization is used at Uber, Facebook, Google, Yelp, Netflix, & elsewhere

Developer tools | Research

Open-sourcing Ax and BoTorch: New AI tools for adaptive experimentation

May 01, 2019 Written by Eytan Bakshy, Max Balandat, Kostya Kashin

How can researchers and engineers explore large configuration spaces that have complex trade-offs when it may take hours or days to evaluate any given configuration? This challenge frequently arises across many domains, including tuning hyperparameters for machine learning (ML) models, finding optimal product settings through A/B testing, and designing next-generation hardware.

Today we are open-sourcing two tools, Ax and BoTorch, that enable anyone to solve challenging exploration problems in both research and production — without the need for large quantities of data.

- Ax is an accessible, general-purpose platform for understanding, managing, deploying, and automating adaptive experiments.
- BoTorch, built on PyTorch, is a flexible, modern library for Bayesian optimization, a probabilistic method for data-efficient global optimization.

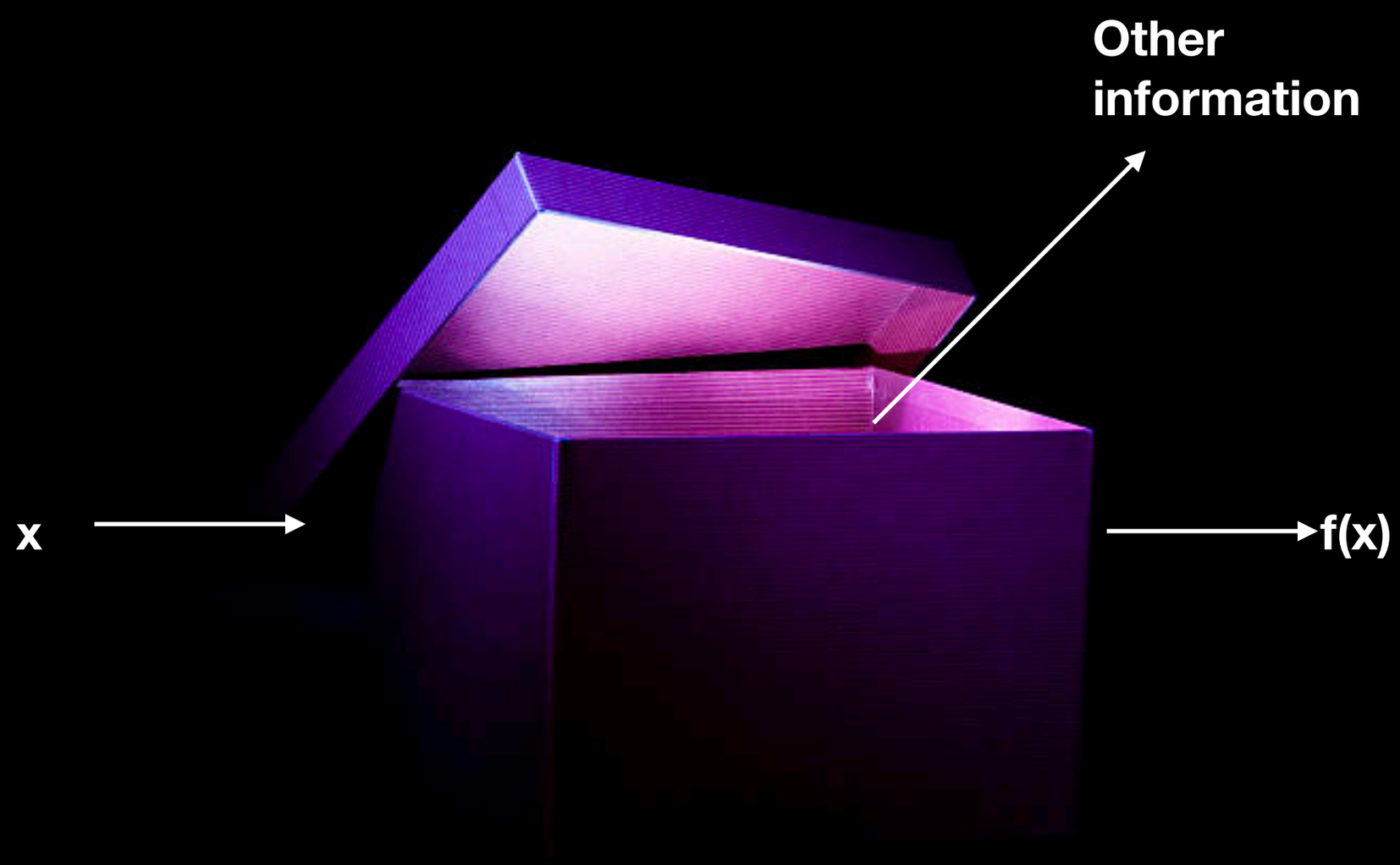
Bayesian Optimization treats the objective function as a black box

Goal: Solve $\min_x f(x)$

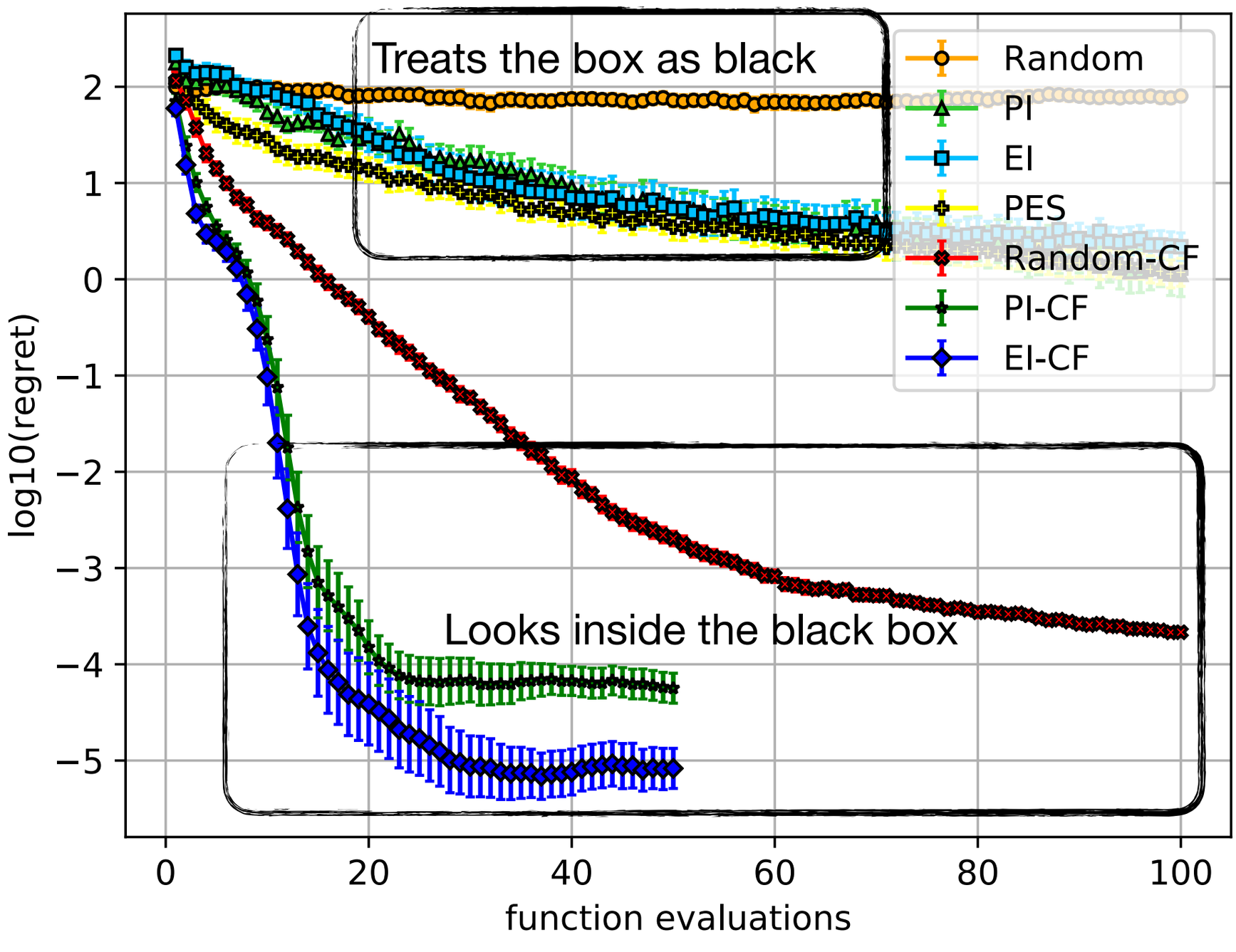
where f(x) is a non-convex derivative-free time-consuming black-box

Kushner 1964; Mockus 1989; Jones, Schonlau & Welch 1998

We can do better by looking inside the box



We can do a lot better by looking inside the box



- Astudillo & F., "Bayesian Optimization of Composite Functions", ICML 2019

- Wu, Toscano-Palmerin, Wilson, F., “Practical Multi-fidelity Bayesian Optimization of Iterative Machine Learning Algorithms” UAI 2019

- Toscano-Palmerin, F. "Bayesian Optimization with Expensive Integrands", in submission, arxiv 1803.08661

- Wu, Poloczek, Wilson, Frazier, "Bayesian Optimization with Gradients" NIPS 2017

- Poloczek, Wang, F., “Multi-Information Source Optimization” Neural Information Processing Systems NIPS 2017

- **Astudillo & F., "Bayesian Optimization of Composite Functions", ICML 2019**

- Wu, Toscano-Palmerin, Wilson, F., “Practical Multi-fidelity Bayesian Optimization of Iterative Machine Learning Algorithms” UAI 2019

- Toscano-Palmerin, F. "Bayesian Optimization with Expensive Integrands", in submission, arxiv 1803.08661

- Wu, Poloczek, Wilson, Frazier, "Bayesian Optimization with Gradients" NIPS 2017

- Poloczek, Wang, F., “Multi-Information Source Optimization” Neural Information Processing Systems NIPS 2017

Bayesian Optimization of Composite Functions

Goal: Solve $\max_{x \in \mathcal{X}} f(x),$

where: $f(x) = g(h(x))$

$h : \mathcal{X} \subset \mathbb{R}^d \to \mathbb{R}^m$ is a time-consuming black box

$g : \mathbb{R}^m \to \mathbb{R}$ and its gradient are fast to compute

$\mathcal{X}$ is a simple compact set in d < 20 dimensions, e.g., a hyper-rectangle

Raúl Astudillo

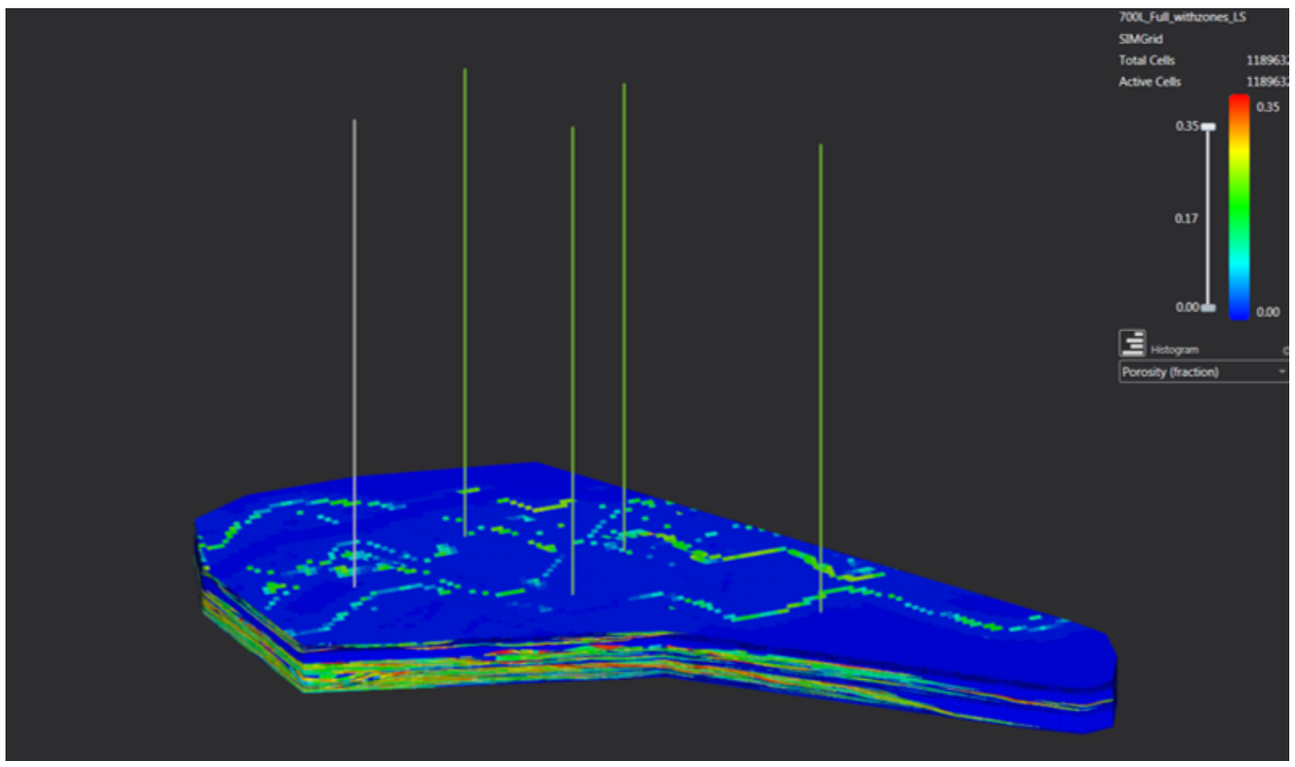
Example: Inverse Problems with Time-Consuming Black Box Forward Models

$$f(x) = g(h(x)) = -\sum_{j=1}^{m}(h_j(x) - y_j)^2,$$

where: x is a parameter vector to calibrate

y is a vector of observational data (e.g., pressures at different test wells)

$h_j(x)$ is the forward model's prediction for the j[th] observation



(Joint work with ExxonMobil)

Other Examples

Inverse Reinforcement Learning (RL)

- y = observed behavior
- h(x) = prediction for observed behavior from RL solver
- g(h(x)) = sum of squared errors (+ regularization)

Materials design

- h(x) = vector of material attributes
- g(h(x)) = performance measure over attributes

Aircraft design

- h(x) = lift and draft as a function of angle of attack and velocity
- g(h(x)) = fuel burn over a mission profile

Related Work

- **Non-Bayesian local/convex optimization of composite functions:**
  Nesterov 2013, Burke & Ferris 1995, Burke 1985, Powell 1983, Fletcher 1982, Anderson and Osborne 1977

- **Non-Bayesian surrogate-based derivative-free nonlinear least squares**:
  Wild 2014; Kelley 2011; Zhang, Conn, Scheinberg 2010

- **Calibration of time-consuming computer models:**
  Overstall & Woods 2013;
  Bliznyuk, Ruppert, Shoemaker, Regis, Wild & Mugunthan 2008

- **BayesOpt with time-consuming constraints:**
  Schonlau, Welch, & Jones, 1998; Gardner, Kusner, Xu, Weinberger, & Cunningham, 2014; Picheny, Gramacy, Wild, & Le Digabel 2016.

- **BayesOpt for sums / multi-task BayesOpt:**
  Swersky, Snoek & Adams, 2013

Standard Bayesian Optimization

while (budget is not exhausted) {

Fit a Gaussian process to observations x, f(x)

Find x that maximizes EI(x) = $E[\{f(x)-f^*\}^+]$

Observe f(x)

}

*Recall: objective is f(x) = g(h(x))

Our Approach

while (budget is not exhausted) {

Fit multi-output Gaussian process regression to observations x, h(x)

Find x that maximizes a new acquisition function, EI-CF(x) = $E[\{g(h(x))-f^*\}^+]$

Observe h(x), f(x)

}

*Recall: objective is f(x) = g(h(x))

Let's see why it works with an **example**

- x is a parameter of a simulator,
- $h(x)$ is simulator's prediction under x,
- y is our observed data.

We want to solve

$$\min_x (h(x) - y)^2.$$

Standard Bayesian Optimization

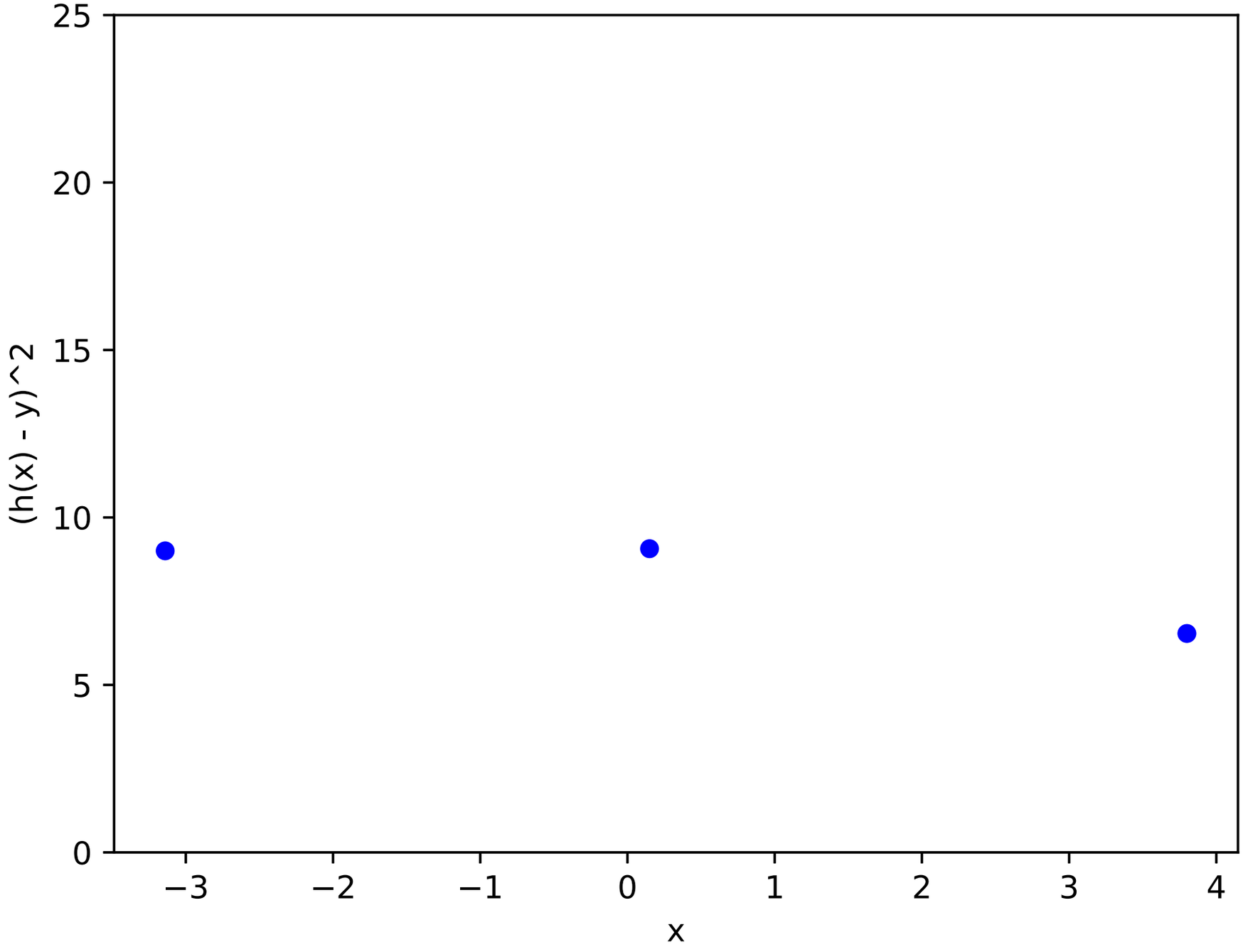


Figure: Evaluations of $(h(x) - y)^2$

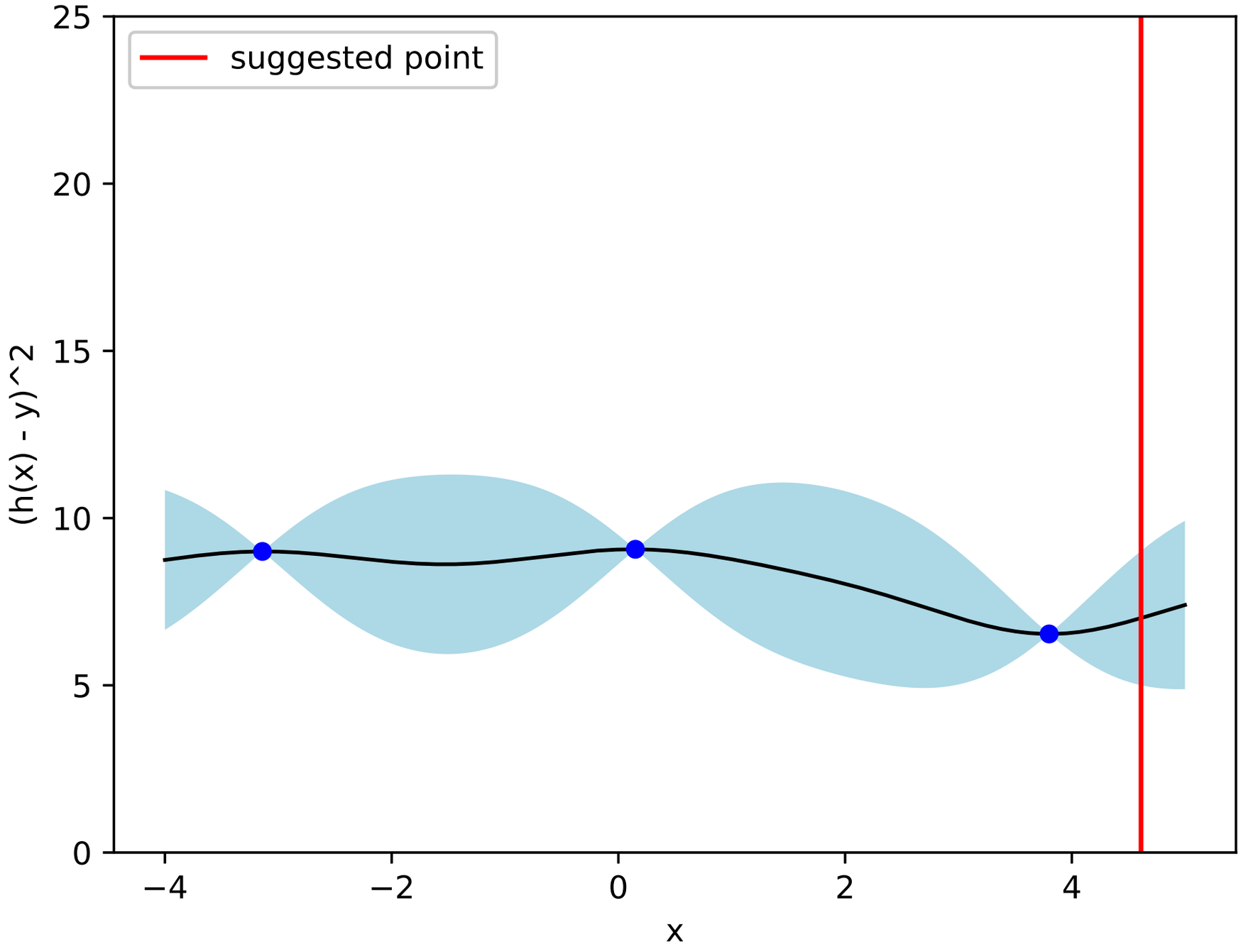


Figure: GP posterior on $(h(x) - y)^2$

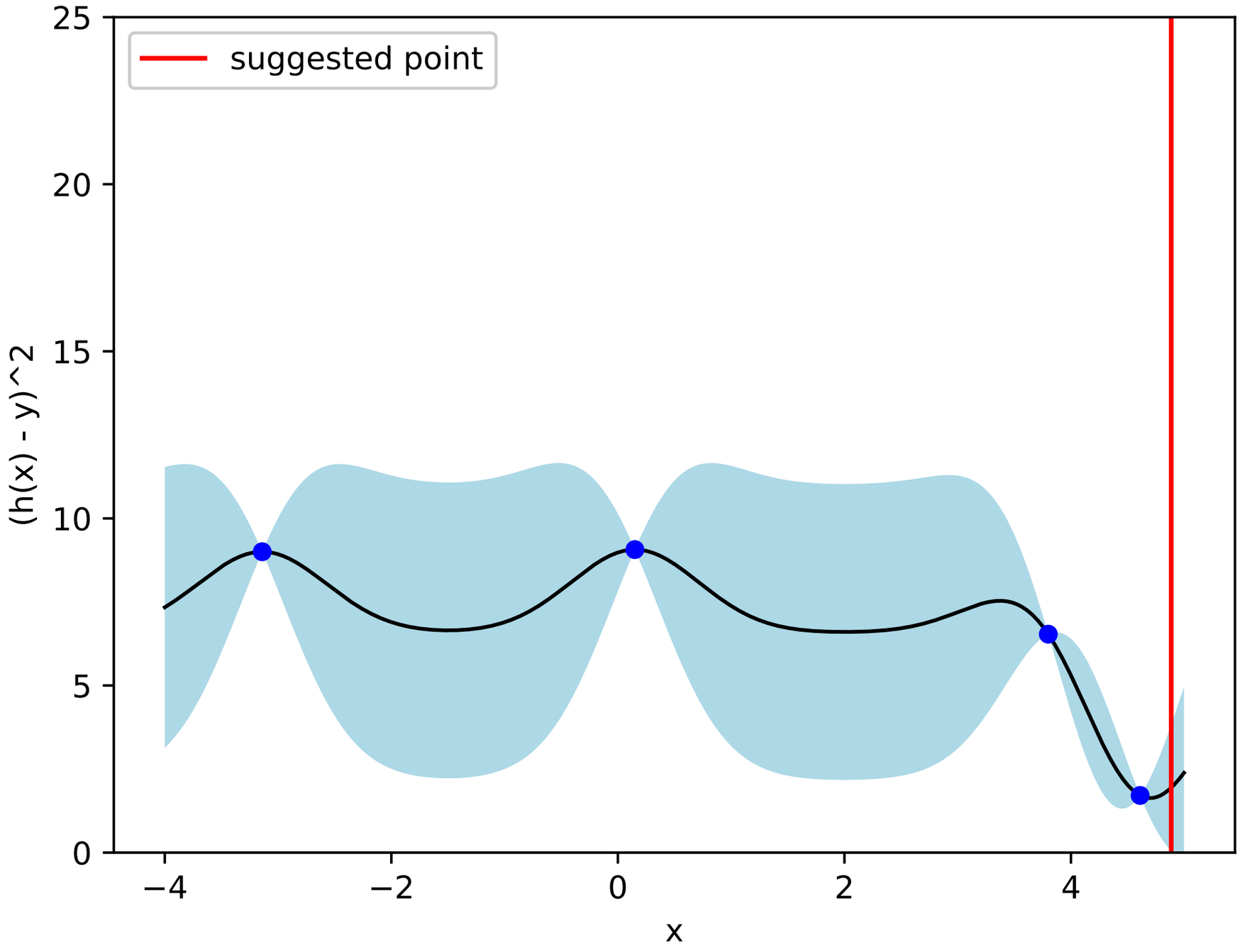
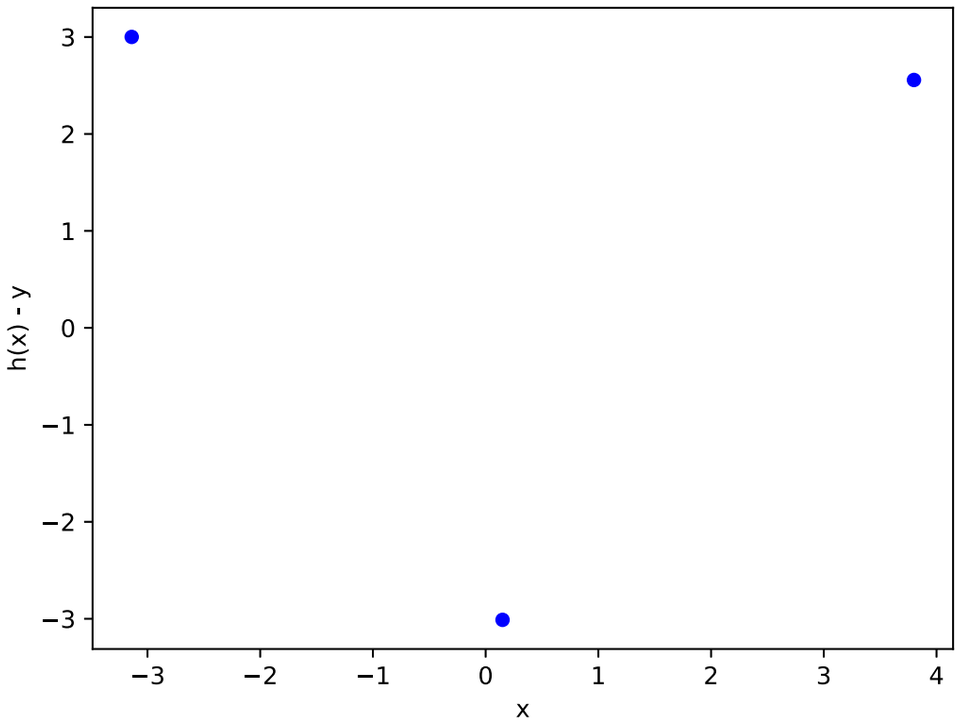
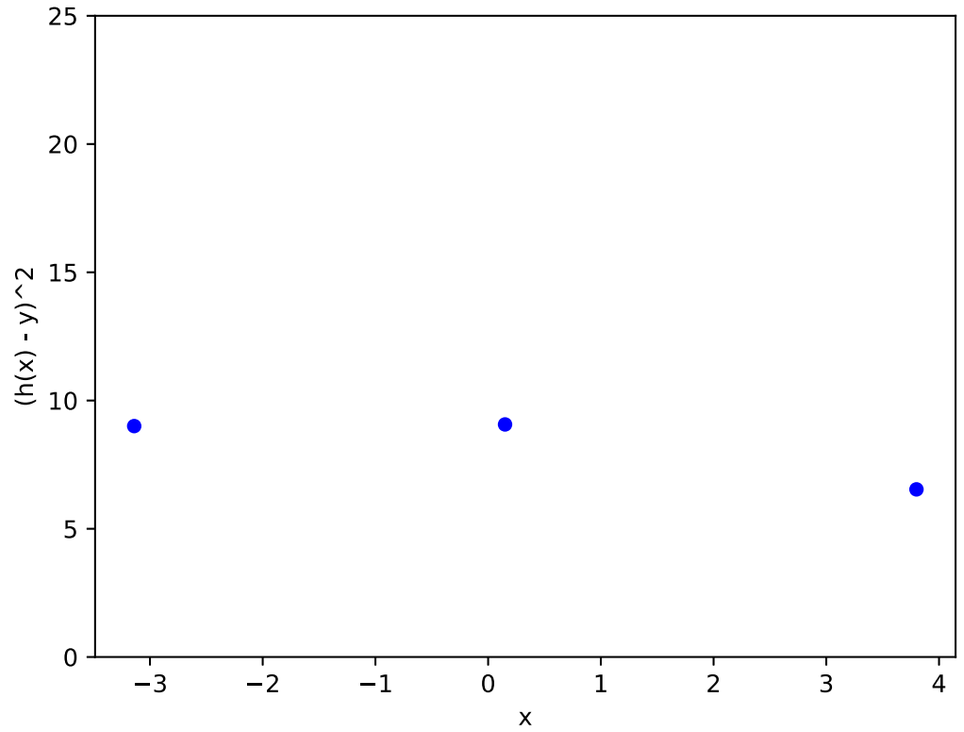


Figure: GP posterior on $(h(x) - y)^2$

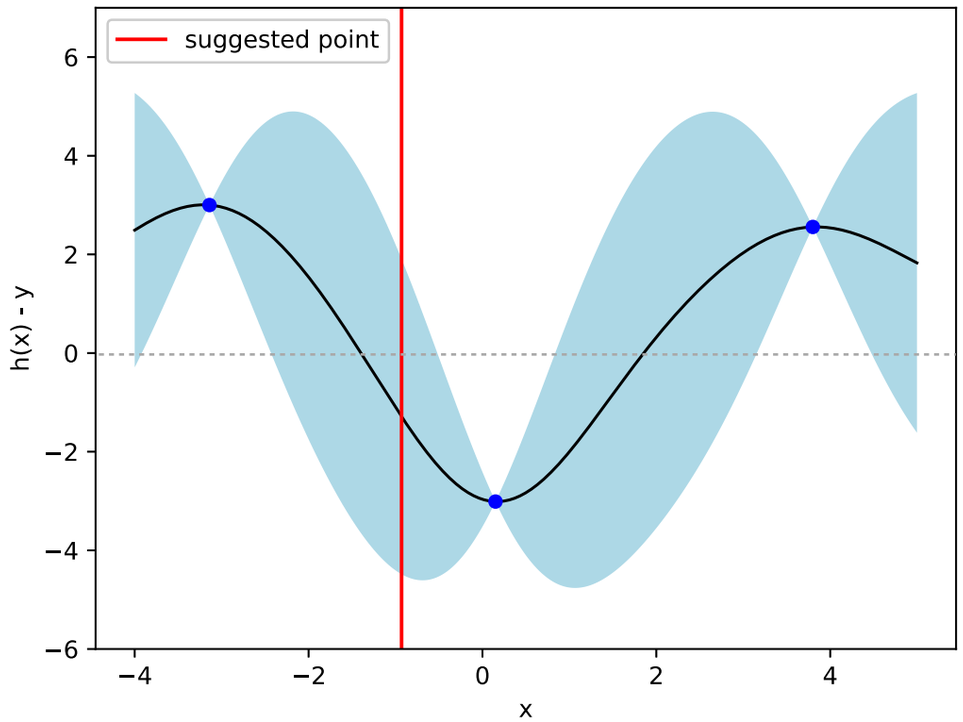
Our Approach



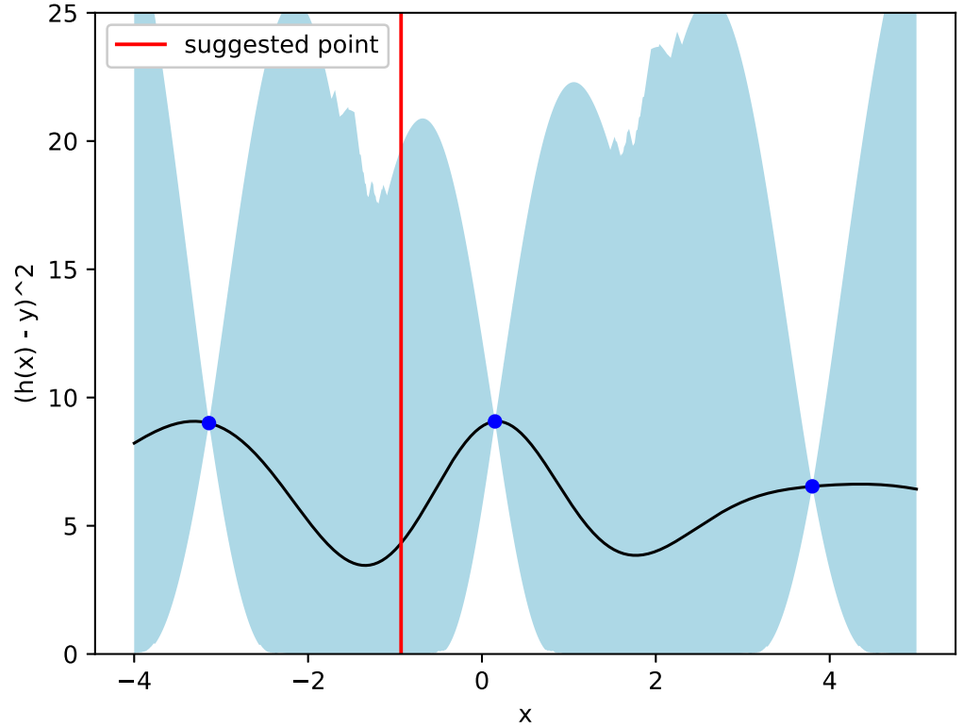
(a) Evaluations of $h(x) - y$



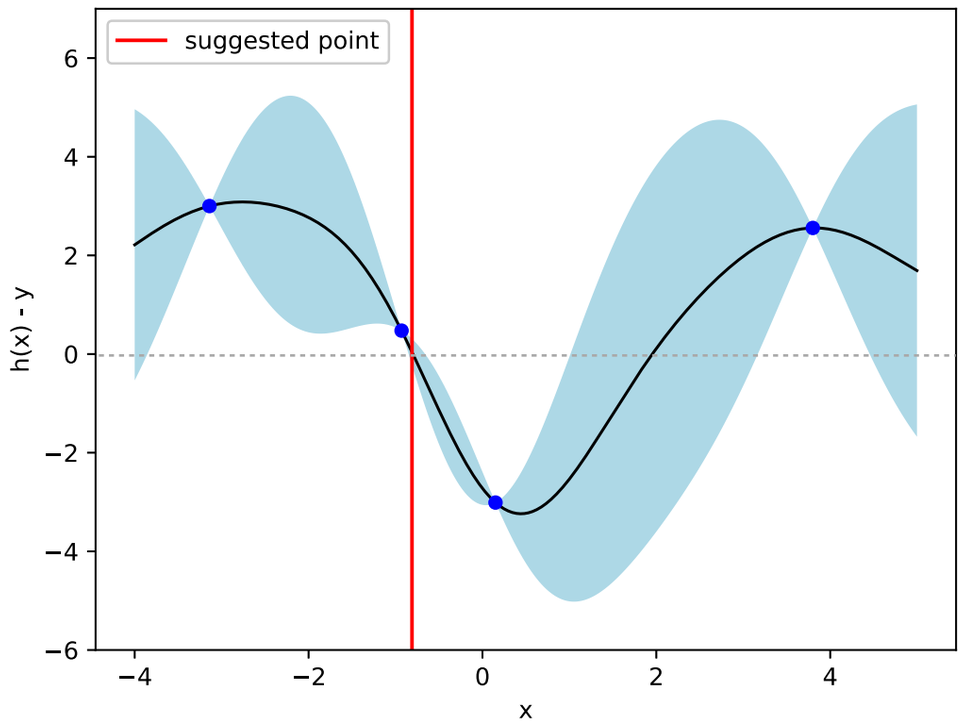
(b) Evaluations of $(h(x) - y)^2$



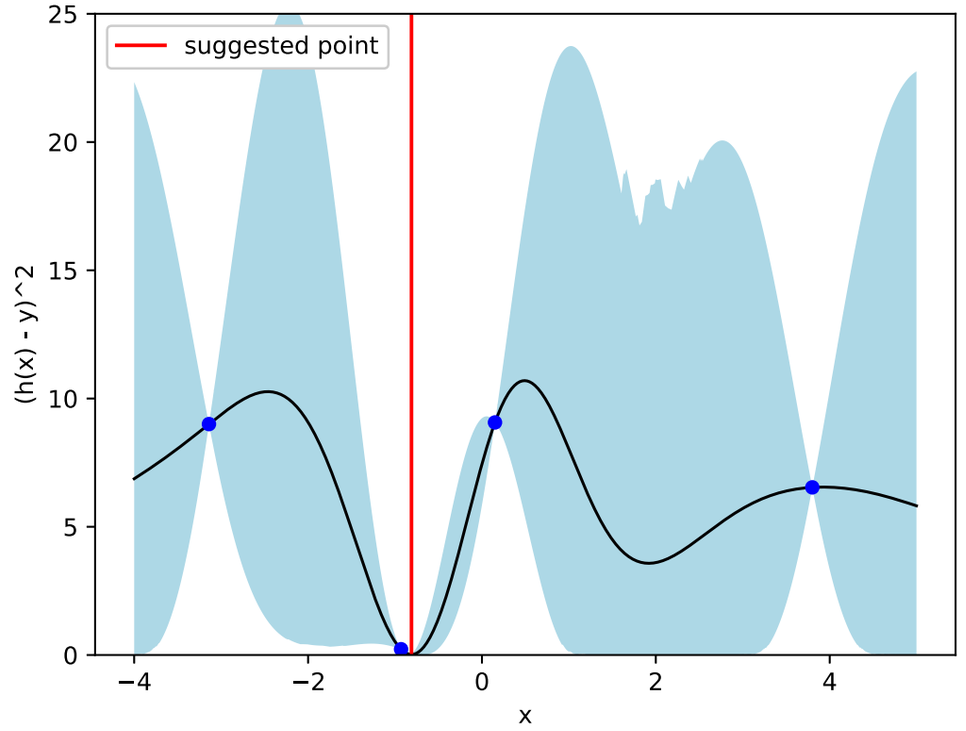
(a) GP posterior on $h(x) - y$



(b) Implied posterior on $(h(x) - y)^2$

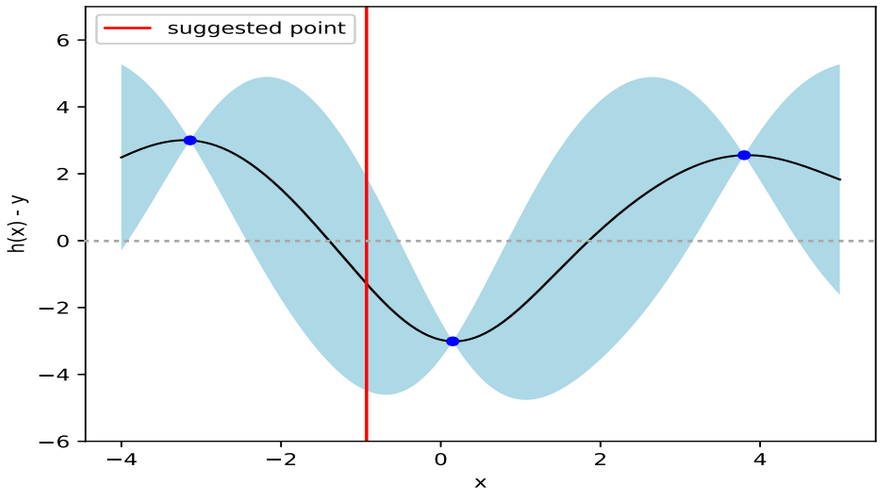


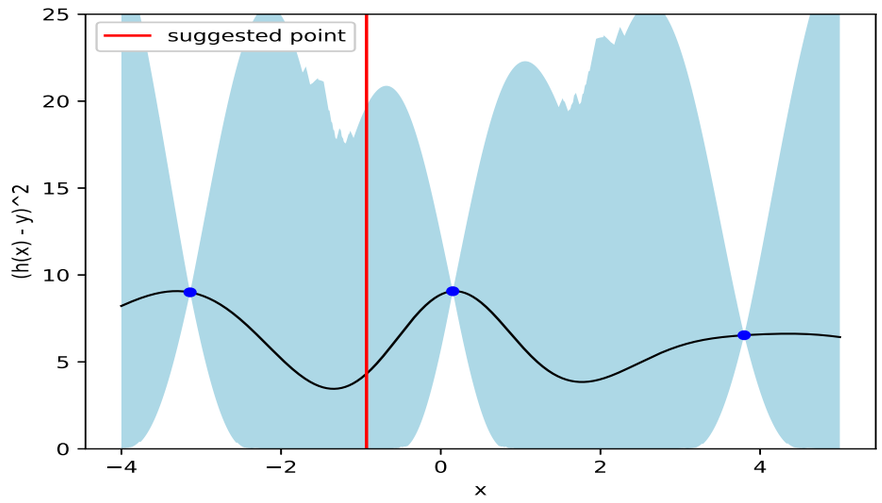
(a) GP posterior on $h(x) - y$

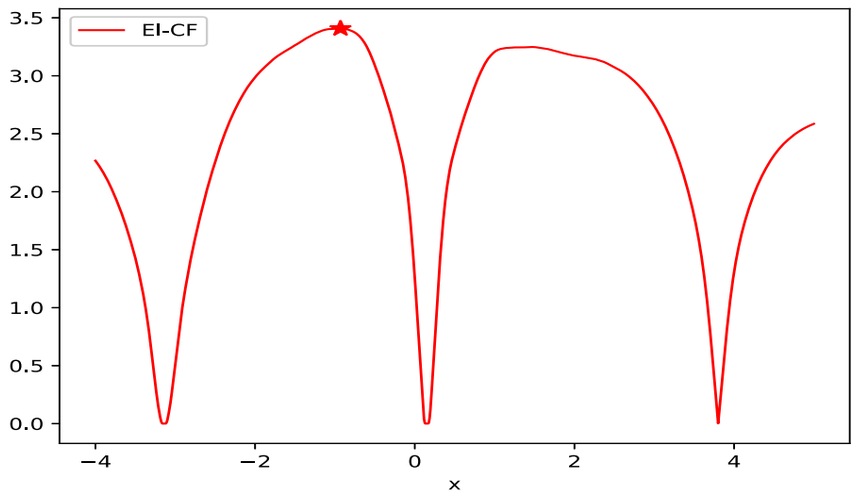


(b) Implied posterior on $(h(x) - y)^2$

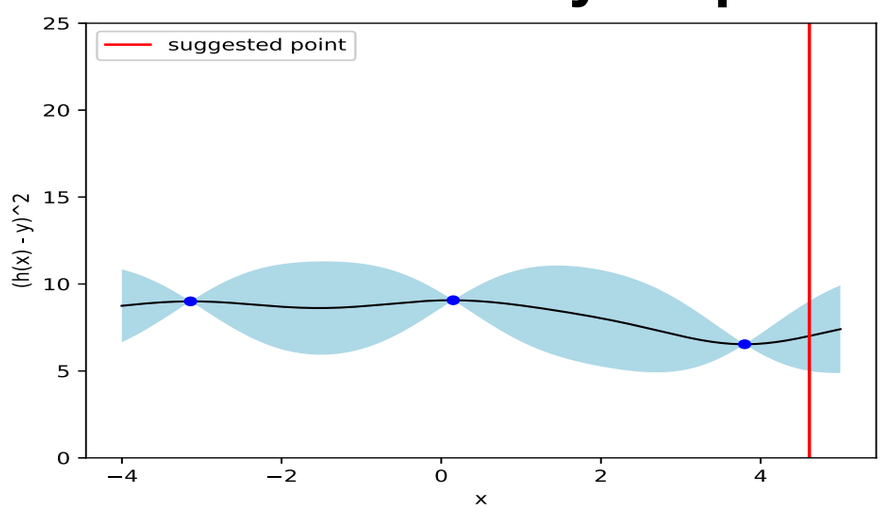
Our Approach

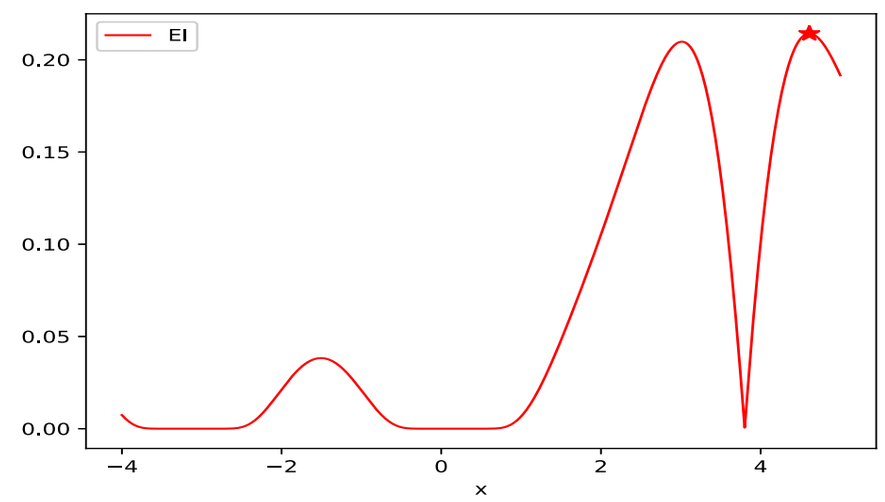






Standard BayesOpt





Implementing our Approach

Challenge: maximizing EI-CF is hard

- In standard Bayesian optimization, f(x) is Gaussian, giving EI a closed form.

Challenge:

- When h is a GP and g is nonlinear, f(x) = g(h(x)) is not Gaussian
- EI-CF has **no closed form**, making it hard to optimize

Calculating EI-CF

To estimate EI-CF(x), repeat the following:

1. Sample h(x) from the Gaussian process posterior
2. Calculate the improvement $\{g(h(x))-f^*\}^+$

Then average the results.

Challenge: maximizing EI-CF is hard

Naive approach: Maximize this simulation-based estimate of EI-CF directly, e.g., with a genetic algorithm

Problem: This will be really slow because we lack gradients and evaluations are noisy

A better way to maximize EI-CF

1. Reparameterization trick

2. Novel stochastic gradient estimator

3. Optimize EI-CF using multi-start stochastic gradient descent

Reparameterization Trick

$h(x) = \mu(x) + C(x)\, Z$

- $\mu(x)$ is the mean of $f(x)$ under the Bayesian posterior probability distribution
- $C(x)$ is the Cholesky decomposition of $h(x)$ under the posterior
- Z is a standard normal random vector

Novel Stochastic Gradient Estimator

$$\nabla_x \text{EI-CF}(x) = \nabla_x E[\{ g(\mu(x)+C(x)Z) - f^* \}^+]$$

$$= E[\nabla_x \{ g(\mu(x)+C(x)Z) - f^* \}^+]$$

$$= E[\gamma(x,Z)],$$

where $\gamma(x,Z)$ is the expression inside the brackets

Exchanging E and ∇ requires regularity conditions (infinitesimal perturbation analysis, Ho & Cao 1991)

Our gradient estimator is unbiased under conditions

Lemma.

Under mild regularity conditions, $\mathrm{EI\text{-}CF}_n$ is differentiable almost everywhere and its gradient, when it exists, is given by

$$\nabla \mathrm{EI\text{-}CF}_n(x) = \mathbb{E}_n\left[\gamma_n(x, Z)\right],$$

where

$$\gamma_n(x, Z) = \begin{cases} 0, \text{ if } g(\mu_n(x) + C_n(x)Z) \leq f_n^*. \\ \nabla g(\mu_n(x) + C_n(x)Z), \text{ otherwise.} \end{cases}$$

Putting it all together: we use multi-start SGD to maximize EI-CF(x)

Run stochastic gradient ascent from multiple randomly chosen starting points

In each iteration of stochastic gradient ascent:

1. Sample a standard normal random vector Z

2. Our stochastic estimator of ∇_x EI-CF(x) is $\gamma(x,Z)$

Under more regularity conditions, stochastic gradient ascent converges to a stationary point of EI-CF(x)

Use Monte Carlo to evaluate EI-CF(x) for each stationary point and choose the best one

What we know theoretically about solution quality

Theorem: If g is continuous, and other regularity conditions hold, EI-CF is **asymptotically consistent**, i.e., it finds the true global optimum as the number of evaluations grows to infinity.

Remark: By construction, evaluating at argmax_x EI-CF(x) is **optimal** (in an average-case sense under the prior on h) if:

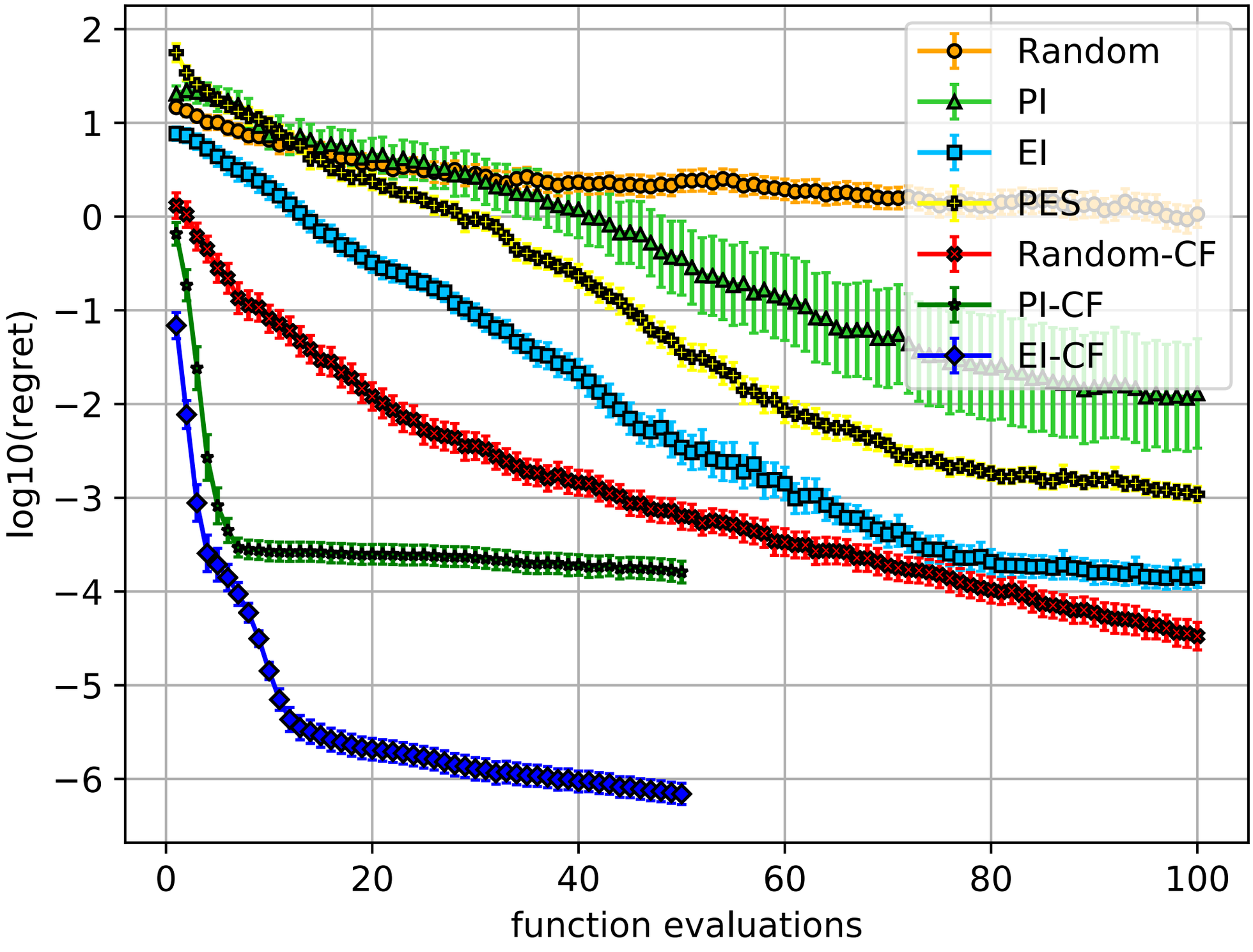
1. this is our last evaluation
2. our solution can only be a previously evaluated point

Environmental model test problem (Bliznyuk et al., 2008)

- Models a chemical accident that has caused a pollutant to spill at two locations.
- Given 12 measurements at different geospatial locations, invert the 4 parameters of this simulator.
- We solve

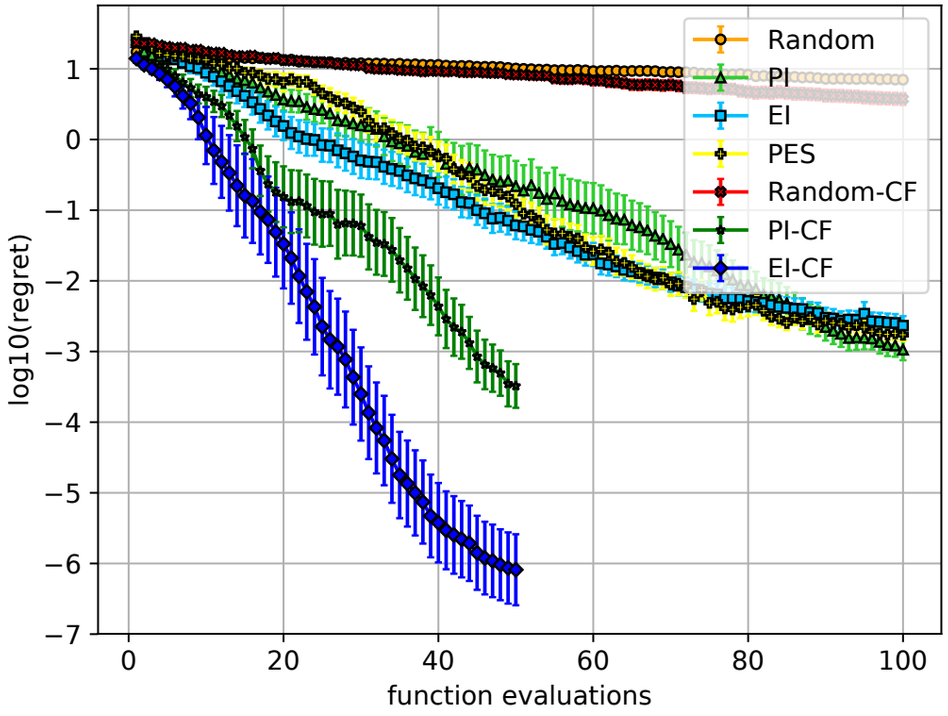
$$\min_{x \in \mathcal{X}} \sum_{j=1}^{12} (s(\theta_j; x^*) - s(\theta_j; x))^2.$$

Environmental model test problem

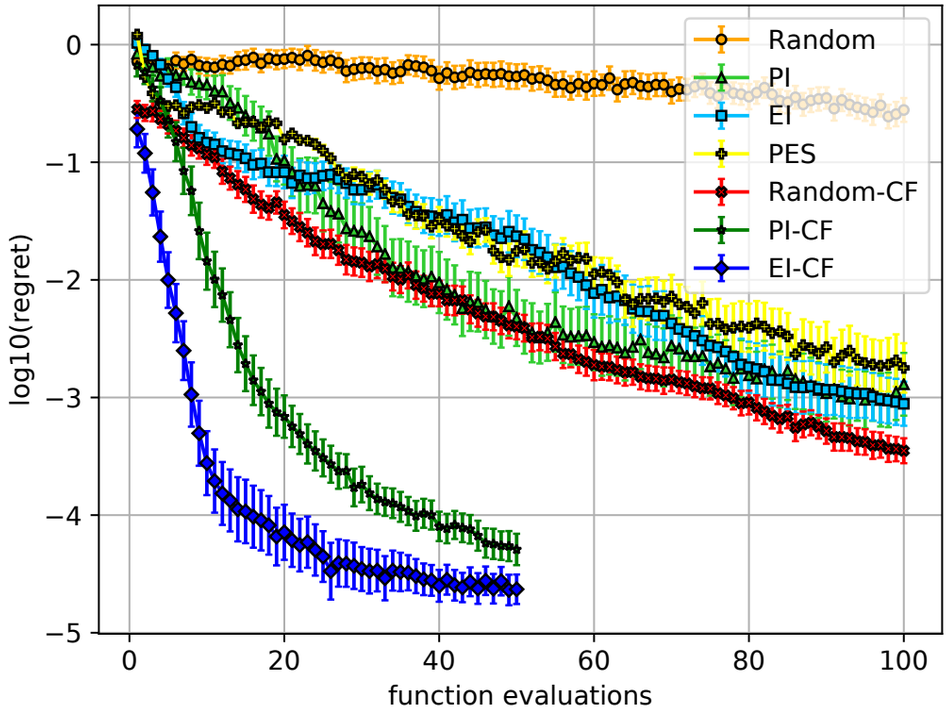


GP-generated test problems

Problem	$\mathcal{X}$	g	m
a	$[0,1]^4$	$g(h(x)) = -\sum_{j=1}^{5}(h_j(x) - y_j^*)^2$	5
b	$[0,1]^3$	$g(h(x)) = -\sum_{j=1}^{4}\exp(h_j(x))$	4



(a)



(b)

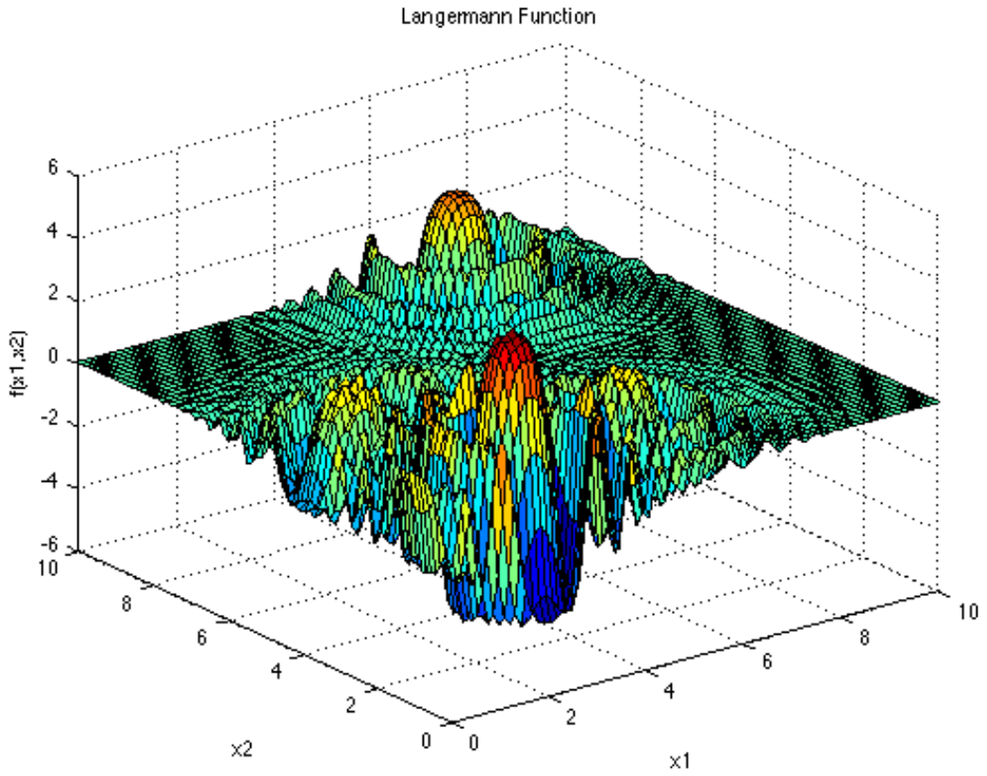
Langermann test problem

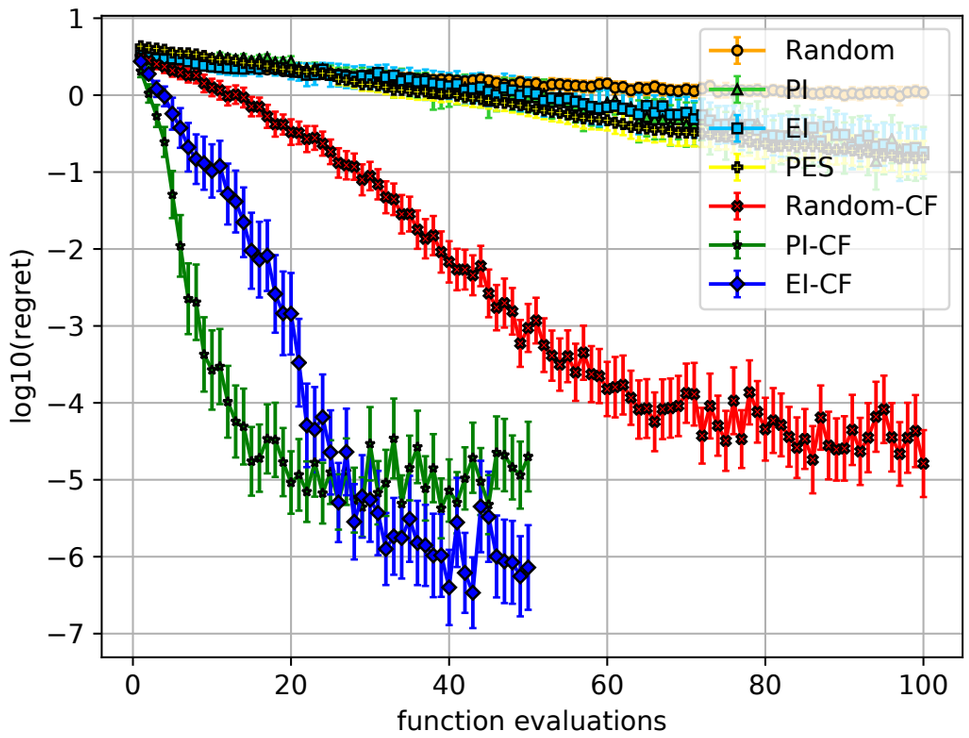
$f(x) = g(h(x))$ where

$$h_j(x) = \sum_{i=1}^{d} (x_i - A_{ij}), \;\; j = 1, \ldots 5,$$

and

$$g(h(x)) = -\sum_{j=1}^{5} c_j \exp(-h_j(x)/\pi) \cos(\pi h_j(x)).$$





5d Rosenbrock test problem

$$f(x) = -\sum_{j=1}^{d-1} 100(x_{j+1} - x_j^2)^2 + (x_j - 1)^2$$

Adapted to our framework by taking $d = 5$ and

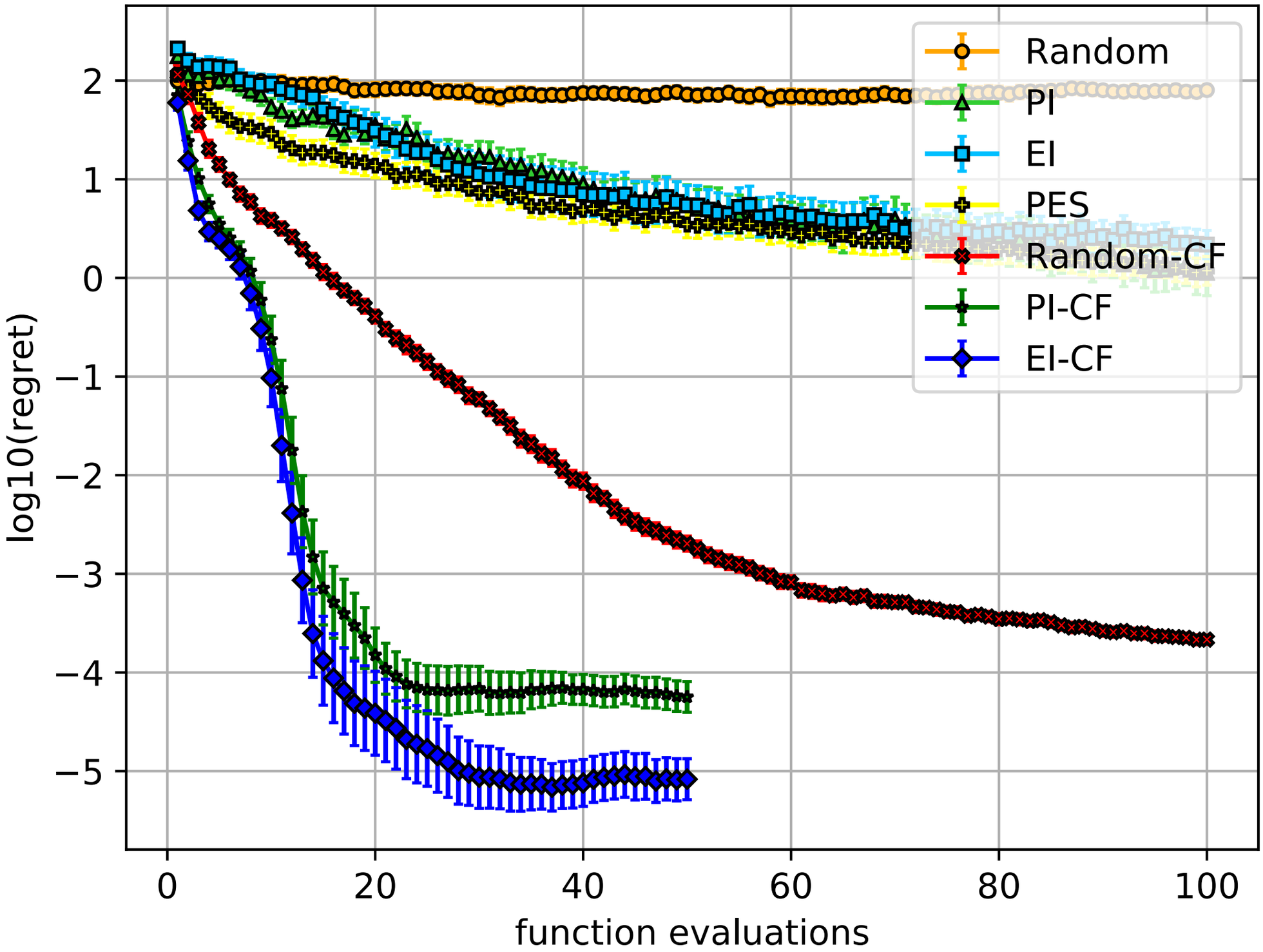
$$h_j(x) = x_{j+1} - x_j^2, \ j = 1, \ldots, 4,$$

$$h_{j+4}(x) = x_j - 1, \ j = 1, \ldots, 4,$$

and

$$g(h(x)) = -\sum_{j=1}^{4} 100 h_j(x)^2 + h_{j+4}(x)^2.$$

5d Rosenbrock test problem



Conclusion

- Exploiting composite objectives can improve BayesOpt performance by 3-6 orders of magnitude.
- There is lots of headroom in looking inside the box
- https://github.com/RaulAstudillo06/BOCF
  + coming soon to Cornell MOE
  https://github.com/wujian16/Cornell-MOE

Thanks!