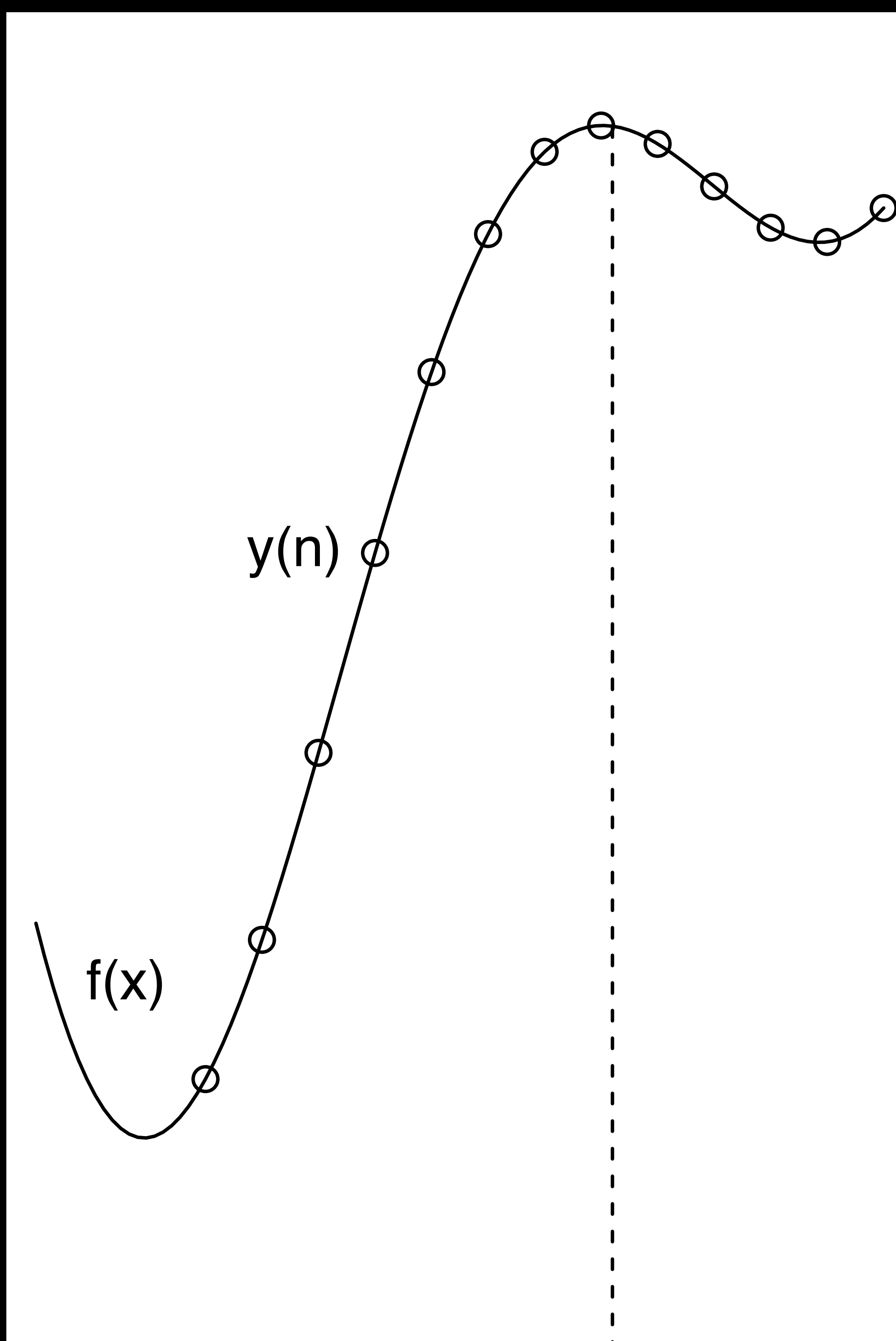
Parallel Bayesian Optimization, for Metrics Optimization at Yelp

Peter Frazier
Assistant Professor
Operations Research & Information Engineering
Cornell University

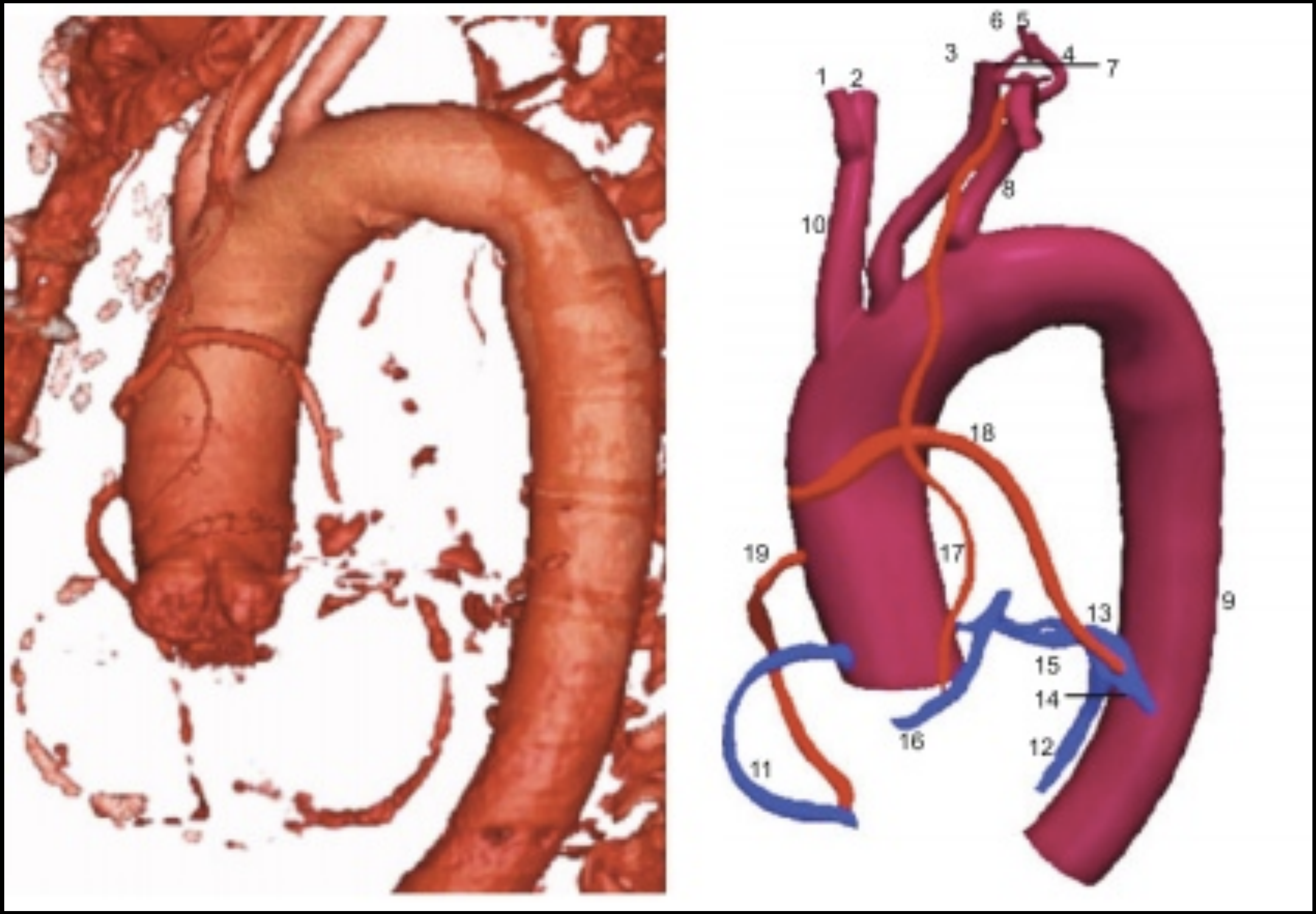
Joint work with:
Jialei Wang (Cornell), Scott Clark (Yelp, SigOpt), Eric Liu (Yelp, SigOpt),
Deniz Oktay (Yelp, MIT), Norases Vesdapunt (Yelp, Stanford)

Consider optimizing an "expensive" function.



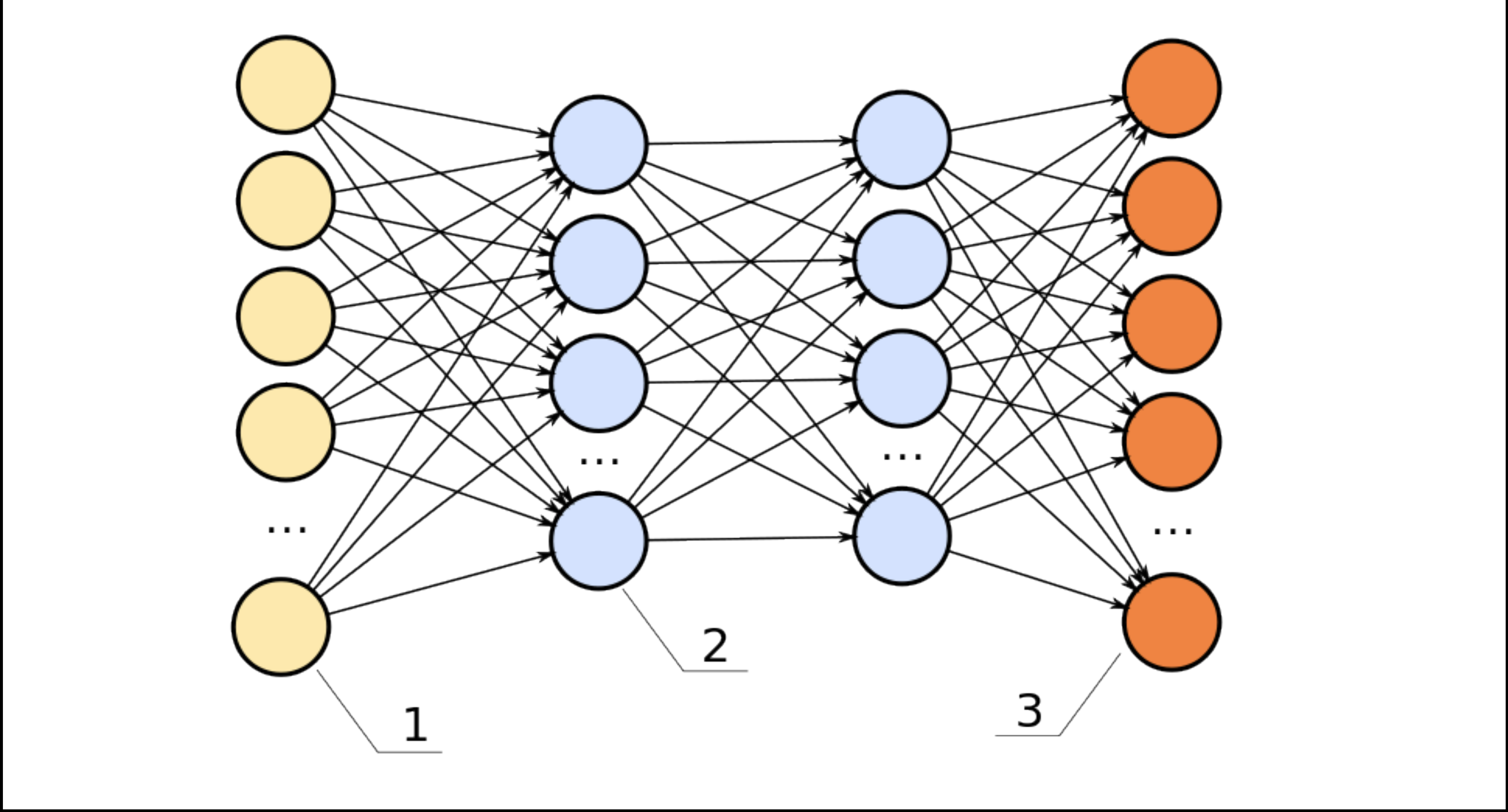
- We'd like to optimize an objective function, $f : \mathbb{R}^d \to \mathbb{R}$.
- f's feasible set is simple, e.g., box constraints.
- f is continuous but lacks special structure, e.g., concavity, that would make it easy to optimize.
- f is derivative-free: evaluations do not give gradient information.
- f is "expensive" to evaluate --- the # of times we can evaluate it is severely limited.

Optimization of expensive functions arises when optimizing physics-based models

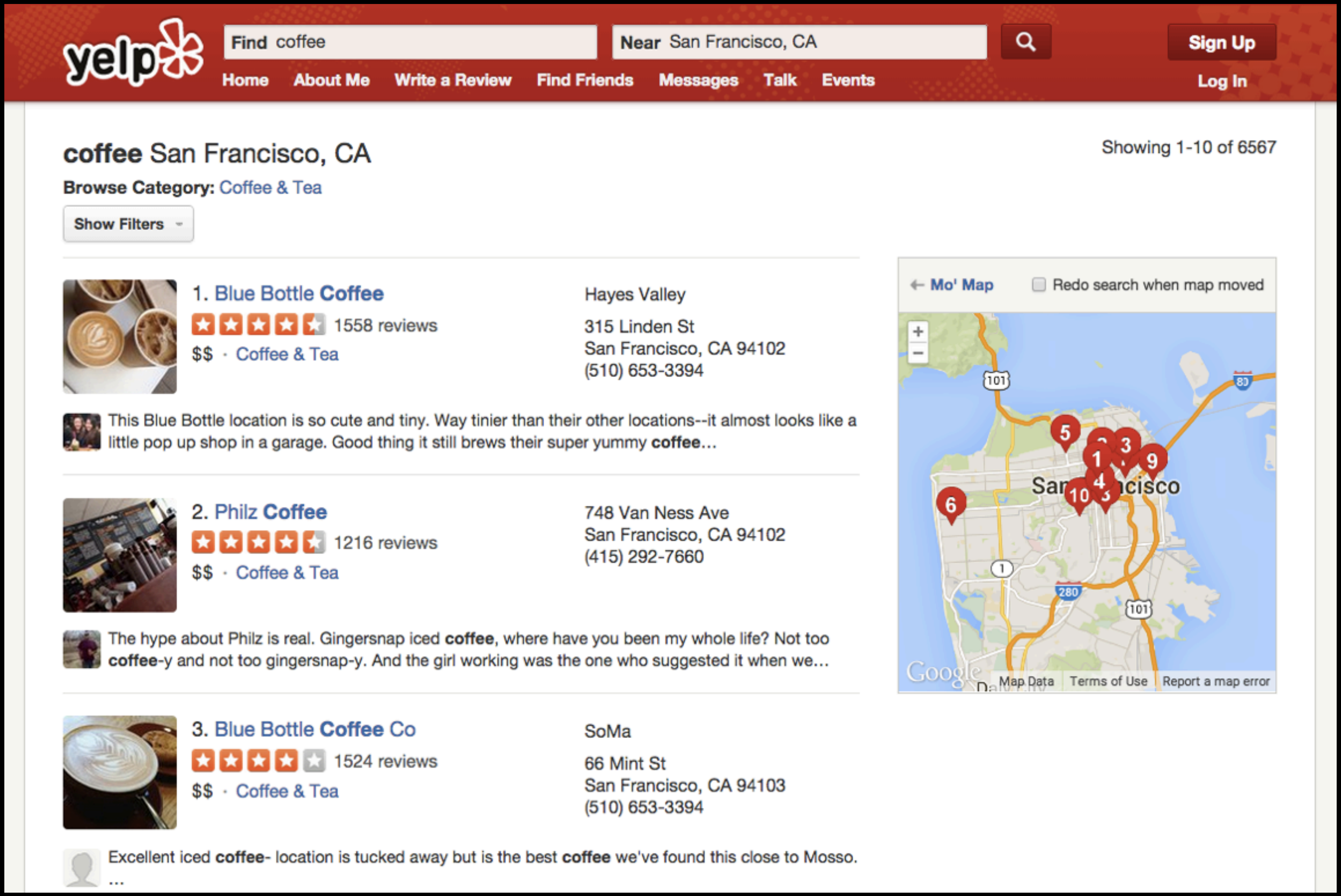


joint work with Alison Marsden, UCSD

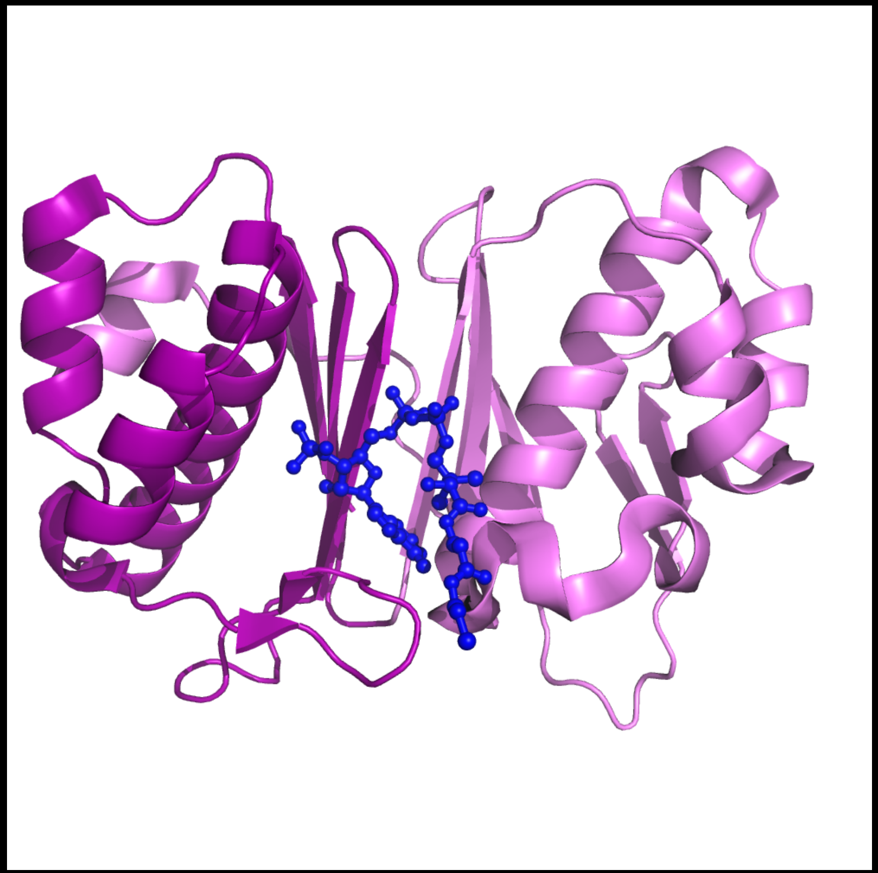
Optimization of expensive functions arises when fitting machine learning models



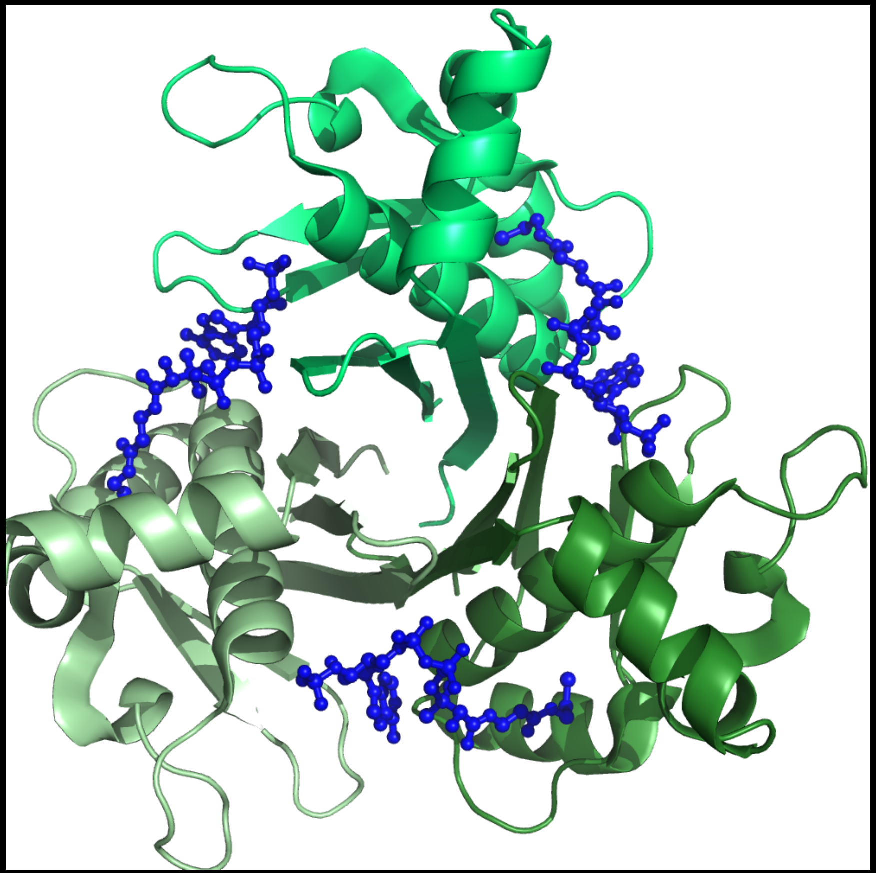
Optimization of expensive functions arises when tuning websites with A/B testing



Optimization of expensive functions arises in drug and materials discovery



Sfp
(a protein-modifying enzyme)



AcpS
(another protein-modifying enzyme)

ongoing work with Mike Burkart and Nathan Gianneschi, UCSD

Bayesian Optimization looks like this

Elicit a prior distribution on the function f (typically a Gaussian process prior).

while (budget is not exhausted) {

Find the point to sample whose value of information is the largest.

Sample that point.

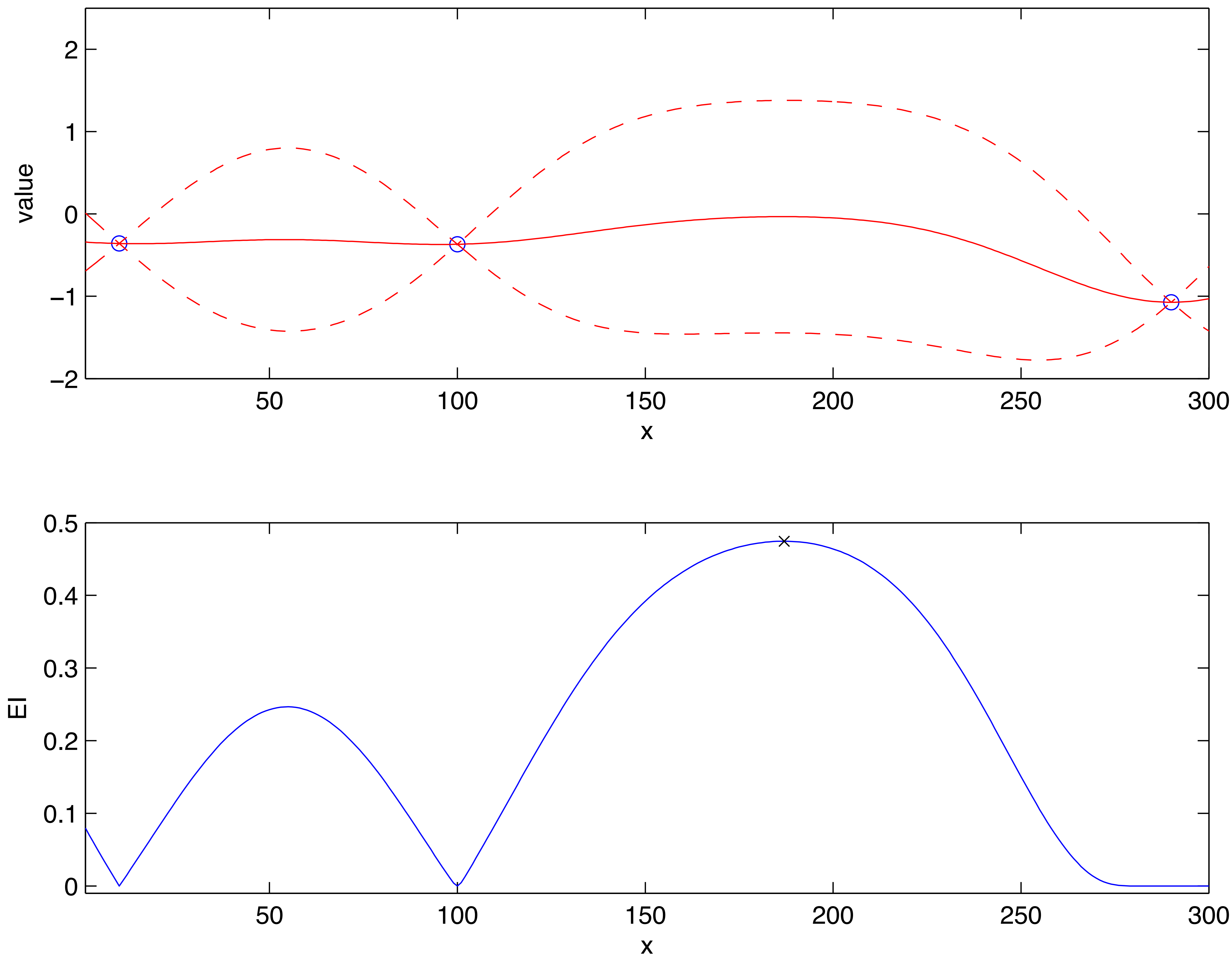
Update the posterior distribution.

}

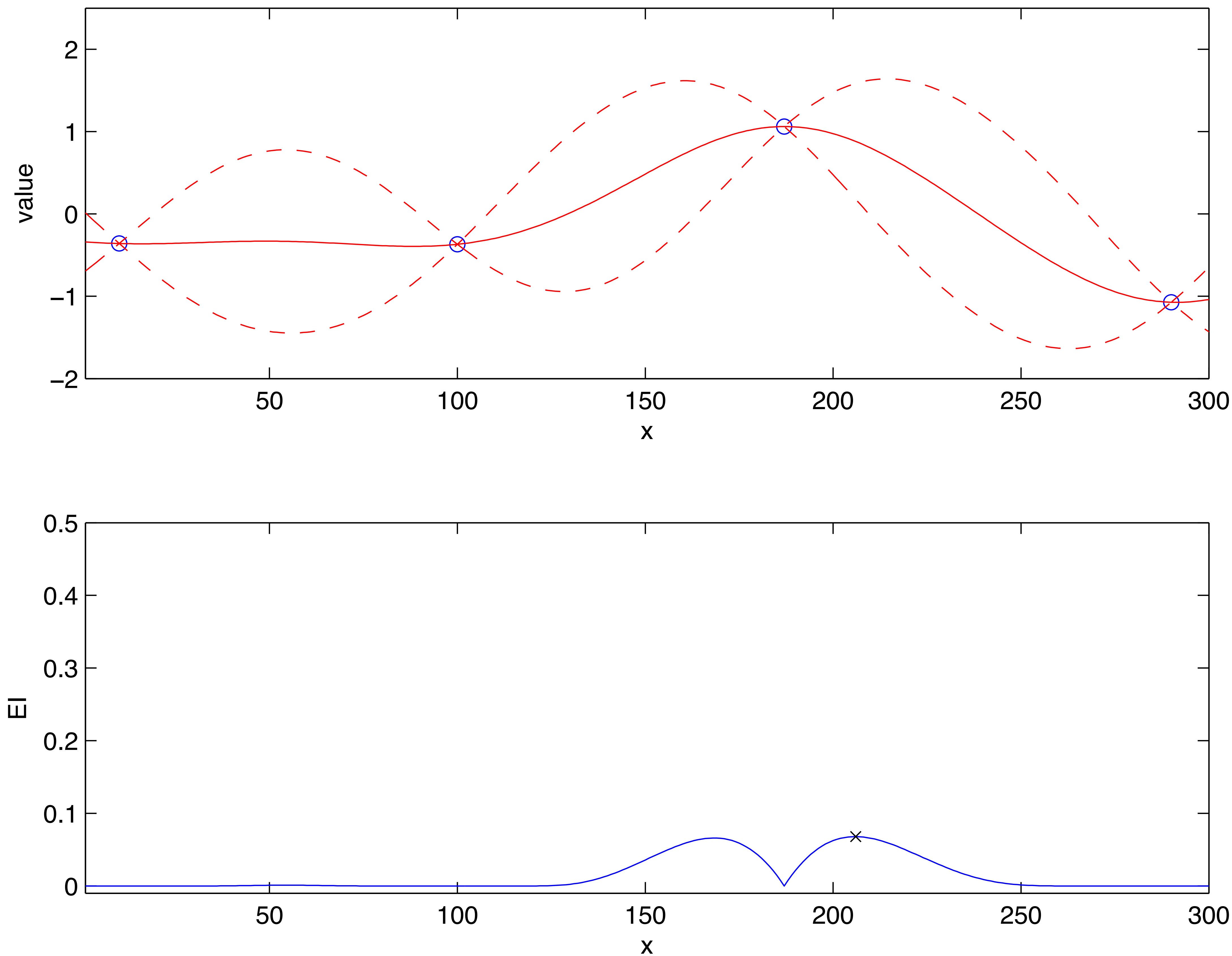
Background: Expected Improvement

- Efficient Global Optimization (EGO) [Jones, Schonlau & Welch 1998; Mockus 1972] is a well-known Bayesian optimization method.
- It does one function evaluation at a time.
- It measures the value of information for each potential measurement using "Expected Improvement."

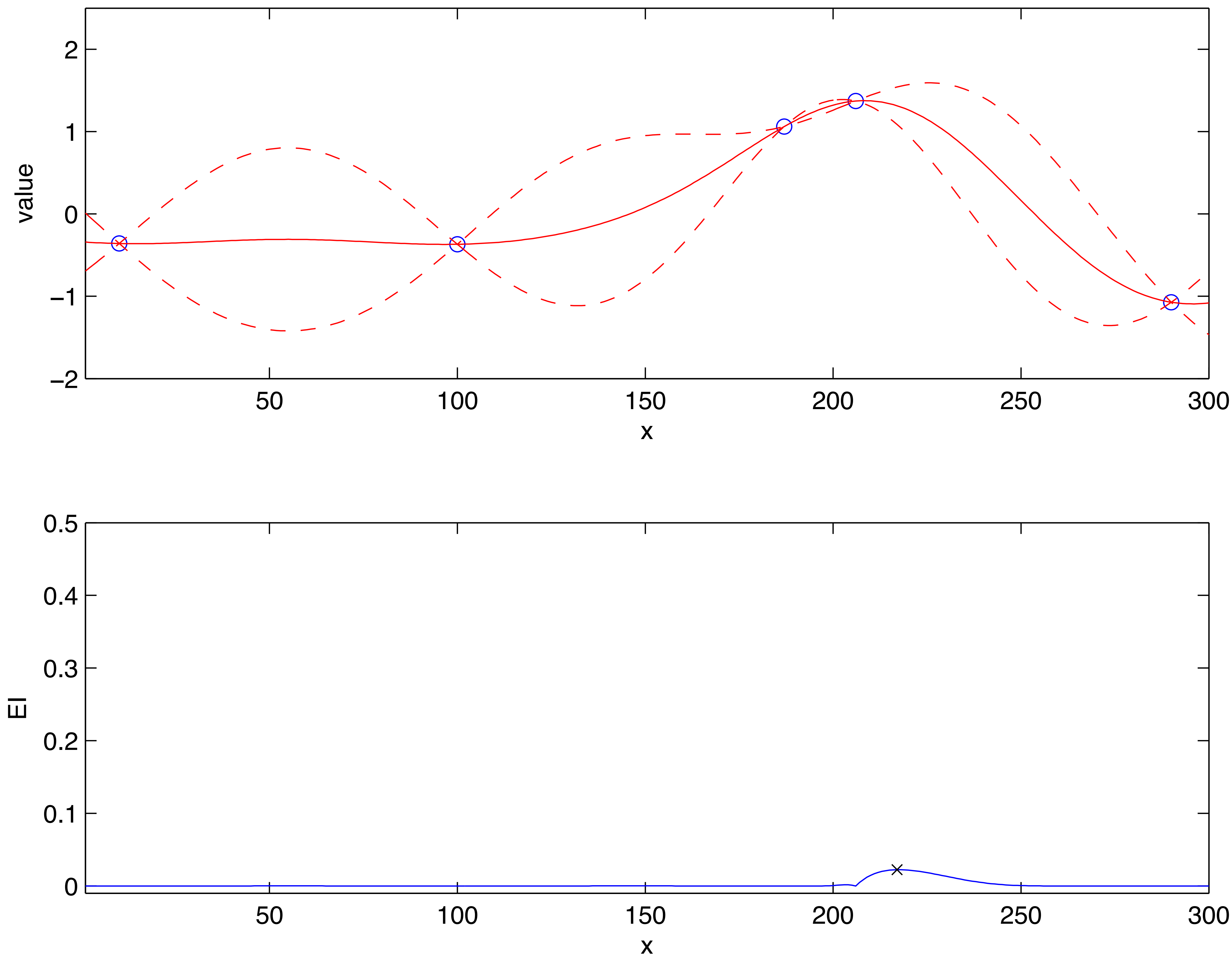
Background: Expected Improvement



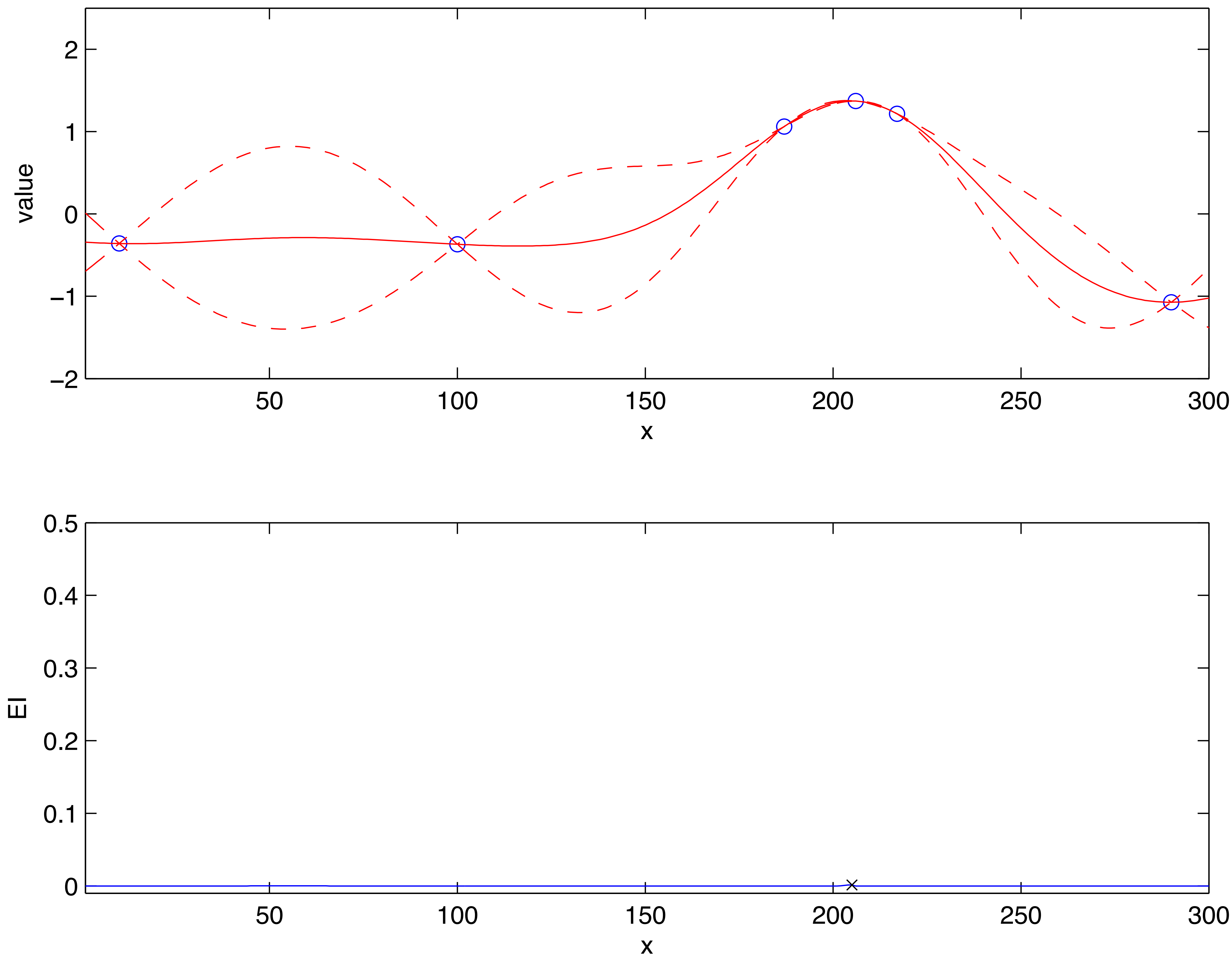
Background: Expected Improvement



Background: Expected Improvement



Background: Expected Improvement



Almost all existing Bayesian Optimization methods take **one** measurement at a time

- EGO / expected improvement take one measurement at a time.
- So do earlier algorithms [Kushner, 1964, Mockus et al., 1978, Mockus, 1989].
- So do later methods [Calvin and Zilinskas, 2005, Villemonteix et al., 2009, Frazier et al., 2009, Huang et al., 2006]
- One exception is a collection of methods by David Ginsbourger and co-authors, and also by Ryan Adams (more later).

We extend Bayesian Optimization to **parallel** function evaluations.


Parallel computer


Parallel A/B tests

- What if we evaluate the function at multiple points simultaneously?
- This happens in parallel computing, A/B testing on the web, and laboratory experiments.
- We use decision theory.
- This was also suggeted by Ginsbourger et al., 2007.

We generalize to multiple function evaluations using a decision-theoretic approach

- We've evaluated $x^{(1)},\ldots x^{(n)}$, and observed $f(x^{(1)}),\ldots,f(x^{(n)})$.
- Once sampling stops, we will select the best point found.
- What is the Bayes-optimal way to choose the set of points $x_1,\ldots,x_q$ to evaluate next?
- In general, we would need to solve a dynamic program.
- When this is the last stage of measurements, the dynamic program becomes a simpler two-stage optimization problem.

We generalize to multiple function evaluations using a decision-theoretic approach

- We've evaluated $x^{(1)},\ldots,x^{(n)}$, & observed $f(x^{(1)}),\ldots,f(x^{(n)})$.
- The **best** value observed is
$$f_n^* = \max\{f(x^{(1)}, \ldots, f(x^{(n)})\}$$
- If we measure at new points $x_1,\ldots,x_q$, and then stop, then the expected value of our new solution is
$$E_n[\max(f_n^*, \max_{i=1,\ldots,q} f(x_i))]$$

We generalize to multiple function evaluations using a decision-theoretic approach

- The **expected improvement** is
  E[value of new solution] - value of old solution
- We write this as
  $$\mathrm{EI}_n(x_1,\ldots,x_q) = E_n[\max(f_n^*, \max_{i=1,\ldots,q} f(x_i))] - f_n^*$$
- Our algorithm will be to sample at the set of points with largest expected improvement
  $$\mathrm{argmax}_{x_1,\ldots,x_q} \mathrm{EI}(x_1,\ldots,x_q)$$

Our approach is Bayes-optimal for one stage of function evaluations

- If we have one stage of function evaluations left, then evaluating

$$\mathrm{argmax}_{x_1,\ldots,x_q} \mathrm{EI}(x_1,\ldots,x_q)$$

  is Bayes-optimal.

- If we have more than one stage left, it is not, but we feel that it is a well-motivated heuristic.

q-EI lacks an easy-to-compute expression

$$\mathrm{EI}_n(x_1, \ldots, x_q) = E_n[(\max_{i=1,\ldots,q} f(x_i) - f_n^*)^+]$$

- When q=1 (no parallelism), this is the expected improvement of Jones et al., 1998, which has a closed-form expression.
- When q=2, Ginsbourger et al., 2007 gives an expression using bivariate normal cdfs.
- When q > 2,
  Ginsbourger et al., 2007 proposes Monte Carlo estimation;
  Chevalier and Ginsbourger, 2013 proposes exact evaluation using repeated calls to high-dimensional multivariate normal cdfs.
  Both are difficult to optimize.

q-EI is hard to optimize

- From Ginsbourger, 2009: "*directly optimizing the q-EI becomes extremely expensive as q and d (the dimension of the inputs) grow.*"
- Rather than optimizing the q-EI, Ginsbourger et al., 2007 and Chevalier and Ginsbourger, 2013 propose other schemes.

Our contribution

- Our 1st contribution is an efficient method for solving

$$\mathrm{argmax}_{x_1,\ldots,x_q} \mathrm{EI}(x_1, \ldots, x_q)$$

- This makes the single-batch Bayes-optimal algorithm implementable, not just conceptual.
- Our 2nd contribution is a high-quality open source implementation, currently in use at Yelp.

Our approach to solving $\mathrm{argmax}_{x_1,\ldots,x_q} \mathrm{EI}(x_1,\ldots,x_q)$

1. Construct an unbiased estimator of
$$\nabla \mathrm{EI}(x_1,\ldots,x_q)$$
using infinitesimal perturbation analysis (IPA).
2. Use multistart stochastic gradient ascent to find an approximate solution to
$$\mathrm{argmax}_{x_1,\ldots,x_q} \mathrm{EI}(x_1,\ldots,x_q)$$

Here's how we estimate ∇EI

- $Y=[f(x_1),\ldots,f(x_q)]'$ is multivariate normal.
- Y's mean vector m and covariance matrix C depend on $x_1,\ldots,x_q$.
- Y=m+CZ, where Z is a vector of independent standard normals.
- $EI(x_1,\ldots,x_q) = E[h(Y)]$ for some function h.
- If our problem is well-behaved, then we can switch derivative and expectation:

  $$\nabla EI(x_1,\ldots,x_q) = E[\nabla h(m+cZ)]$$

Here's how we estimate ∇EI

- $Y=[f(x_1),\ldots,f(x_q)]'$ is multivariate normal.
- Y's mean vector m and covariance matrix C depend on $x_1,\ldots,x_q$.
- Y=m+CZ, where Z is a vector of independent standard normals.
- $EI(x_1,\ldots,x_q) = E[h(Y)]$ for some function h.
- If our problem is well-behaved, then we can switch derivative and expectation:

$$\nabla EI(x_1,\ldots,x_q) = E[\nabla h(m+cZ)]$$

This is our gradient estimator, $g(x_1,\ldots,x_q,Z)$

Our gradient estimator is unbiased, given mild sufficient conditions

Theorem

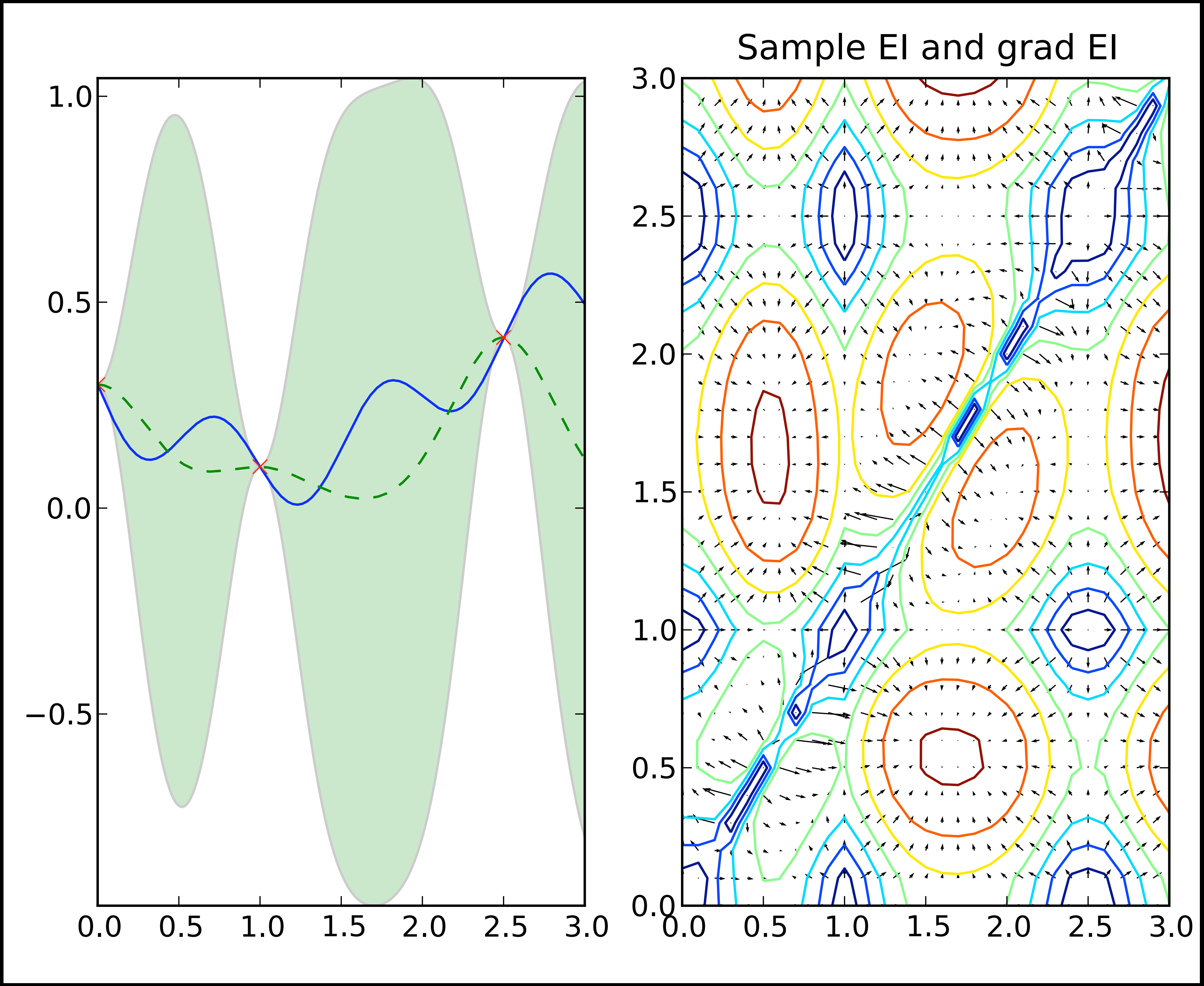
Let $\vec{m}(\vec{x}_1,\ldots,\vec{x}_q)$ and $C(\vec{x}_1,\ldots,\vec{x}_q)$ be the mean vector and Cholesky factor of the covariance matrix of $(f(\vec{x}_1),\ldots,f(\vec{x}_q))$ under the posterior distribution at time n. If the following conditions hold

- *$\vec{m}(\cdot)$ and $C(\cdot)$ are three times continuously differentiable in a neighborhood of $\vec{x}_1,\ldots,\vec{x}_q$.*
- *$C(\vec{x}_1,\ldots,\vec{x}_q)$ has no duplicated rows.*

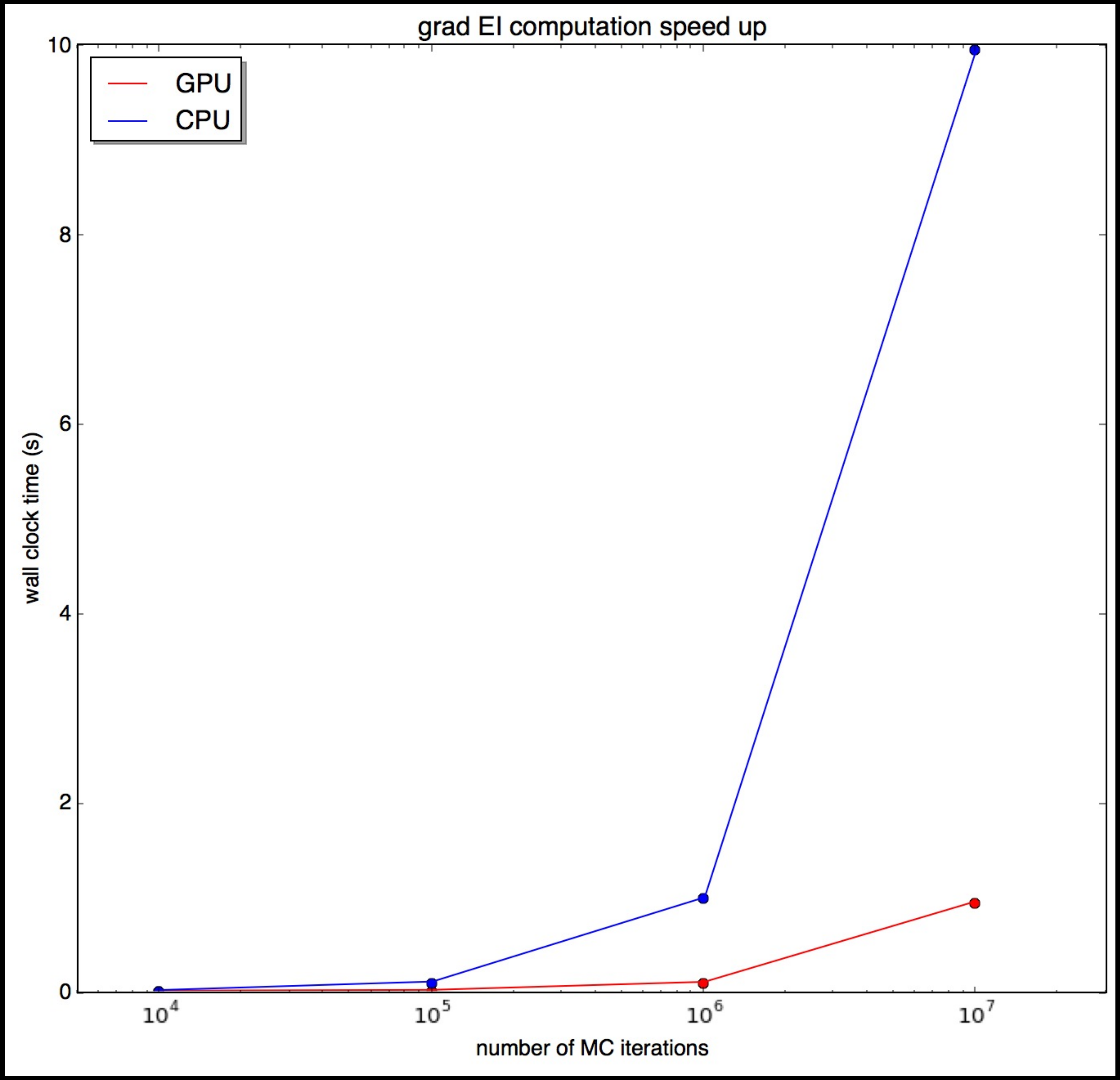
then

$$\nabla \mathrm{EI}(\vec{x}_1,\ldots,\vec{x}_q) = \mathbb{E}_n\left[g(\vec{x}_1,\ldots,\vec{x}_q,\vec{Z})\right].$$

Here's what ∇EI looks like



Estimating ∇EI can be parallelized on a GPU

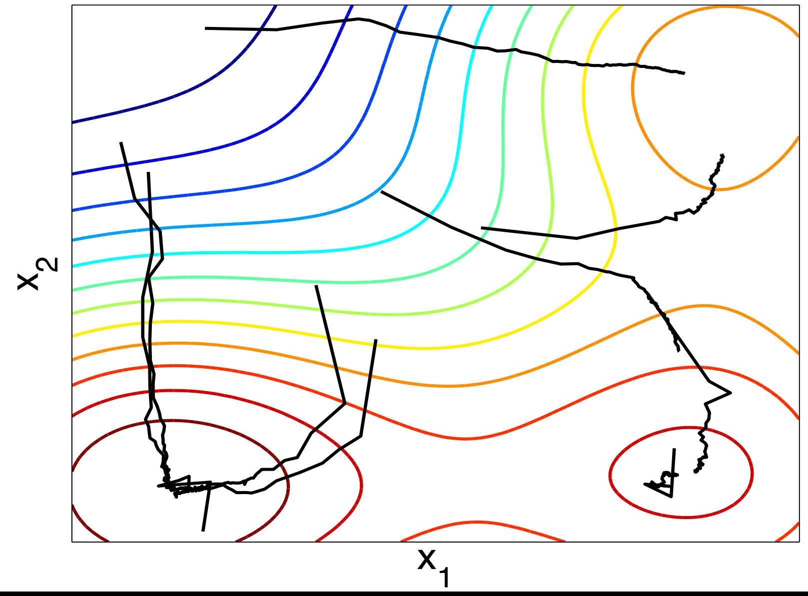


We use this gradient estimator in multistart stochastic gradient ascent

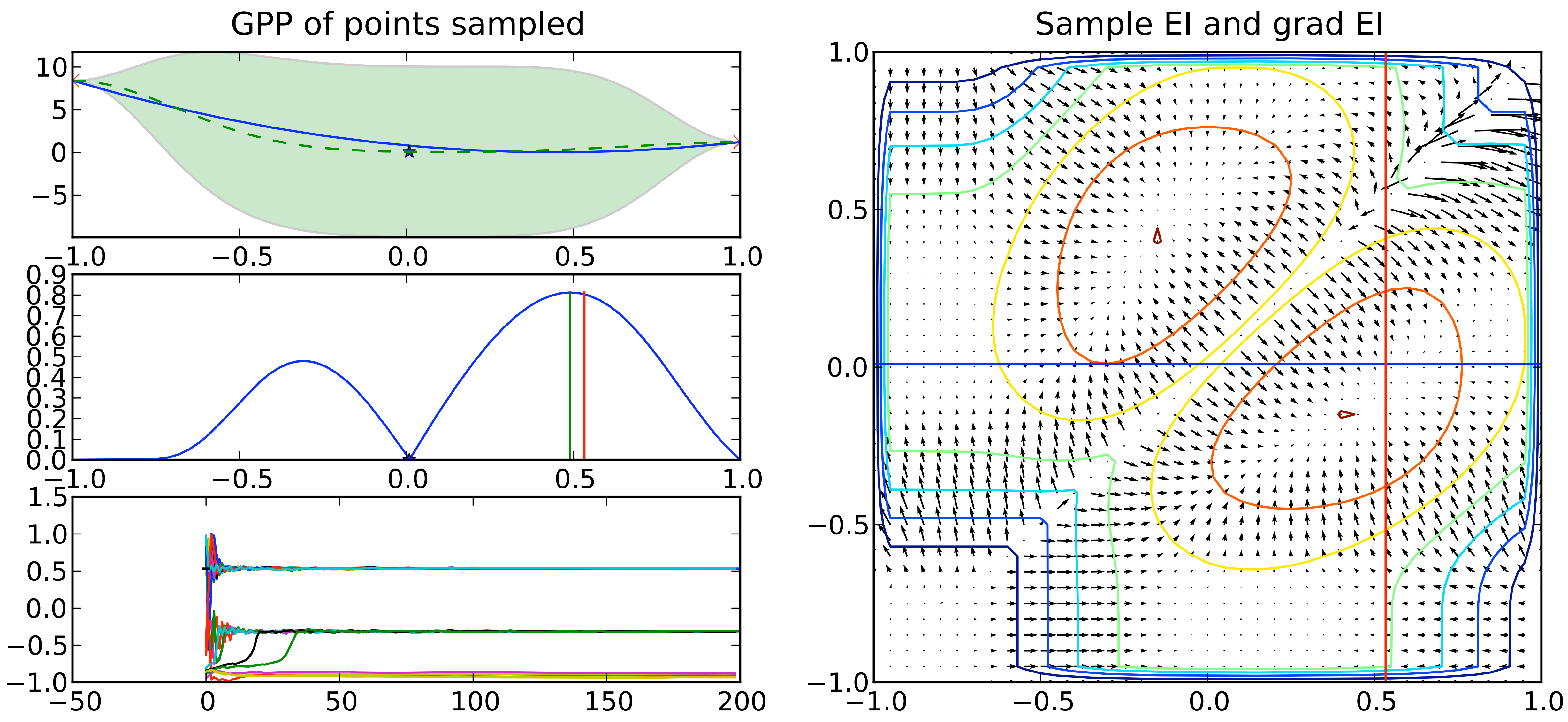
1. Select several starting points, uniformly at random.
2. From each starting point, iterate using the stochastic gradient method until convergence.

$$(\vec{x}_1, \dots, \vec{x}_q) \leftarrow (\vec{x}_1, \dots, \vec{x}_q) + \alpha_n g(\vec{x}_1, \dots, \vec{x}_q, \omega),$$

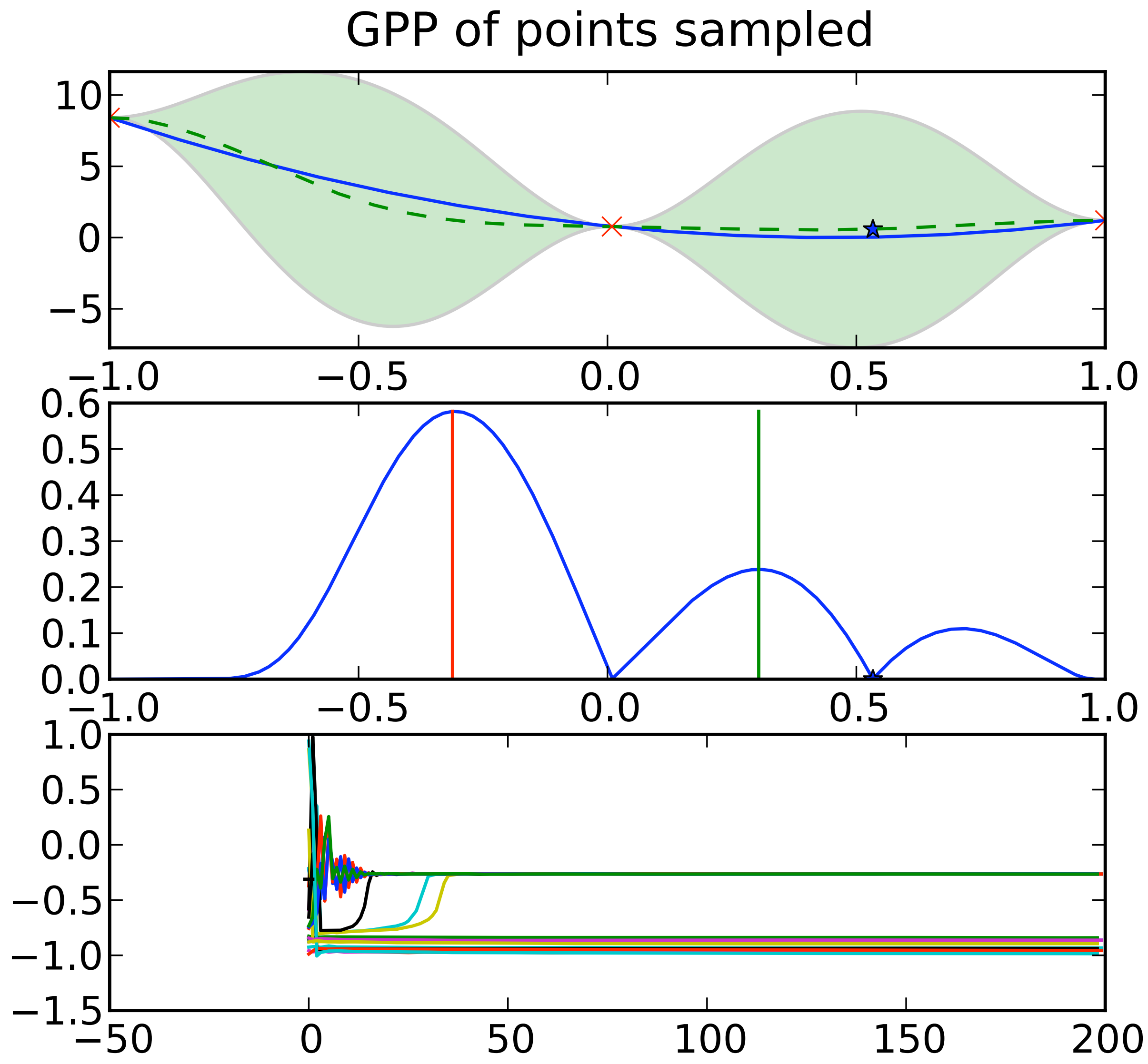
   where (α_n) is a stepsize sequence.
3. For each starting point, average the iterates to get an estimated stationary point. (Polyak-Ruppert averaging)
4. Select the estimated stationary point with the best estimated value as the solution.



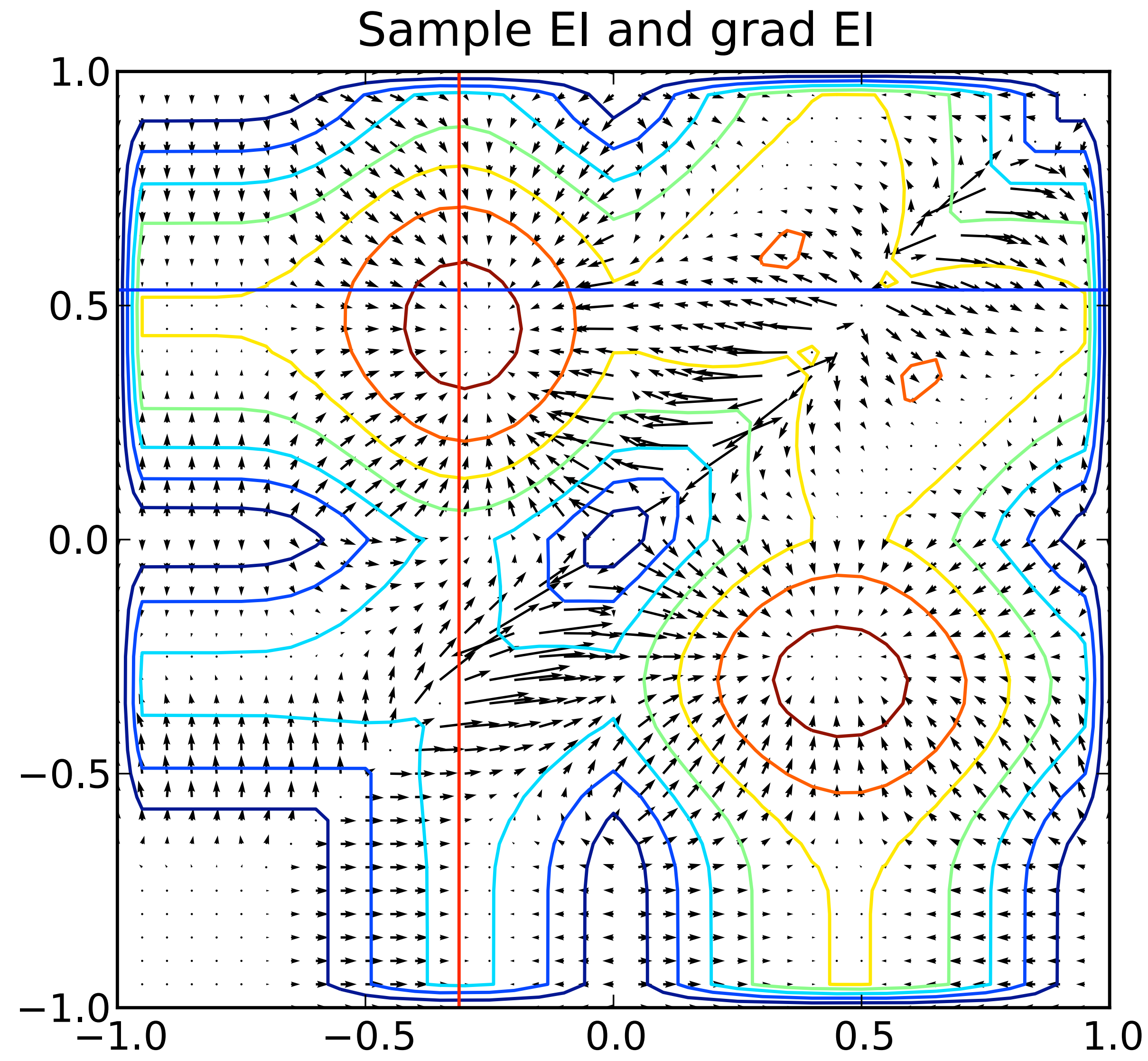
Animation



Animation

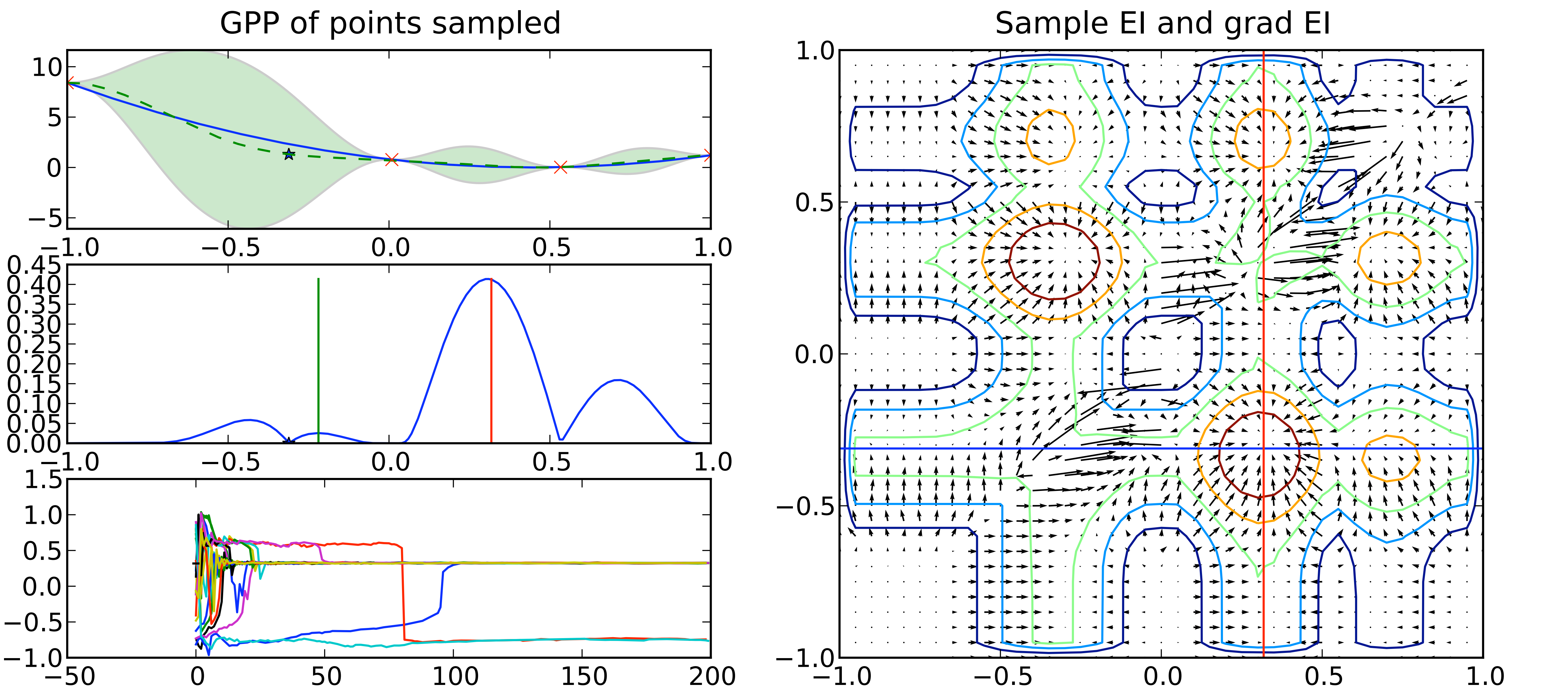


GPP of points sampled
10
5
0
−5
−1.0
−0.5
0.0
0.5
1.0
0.6
0.5
0.4
0.3
0.2
0.1
0.0
1.0
0.5
0.0
−0.5
−1.0
−1.5
−50
0
50
100
150
200

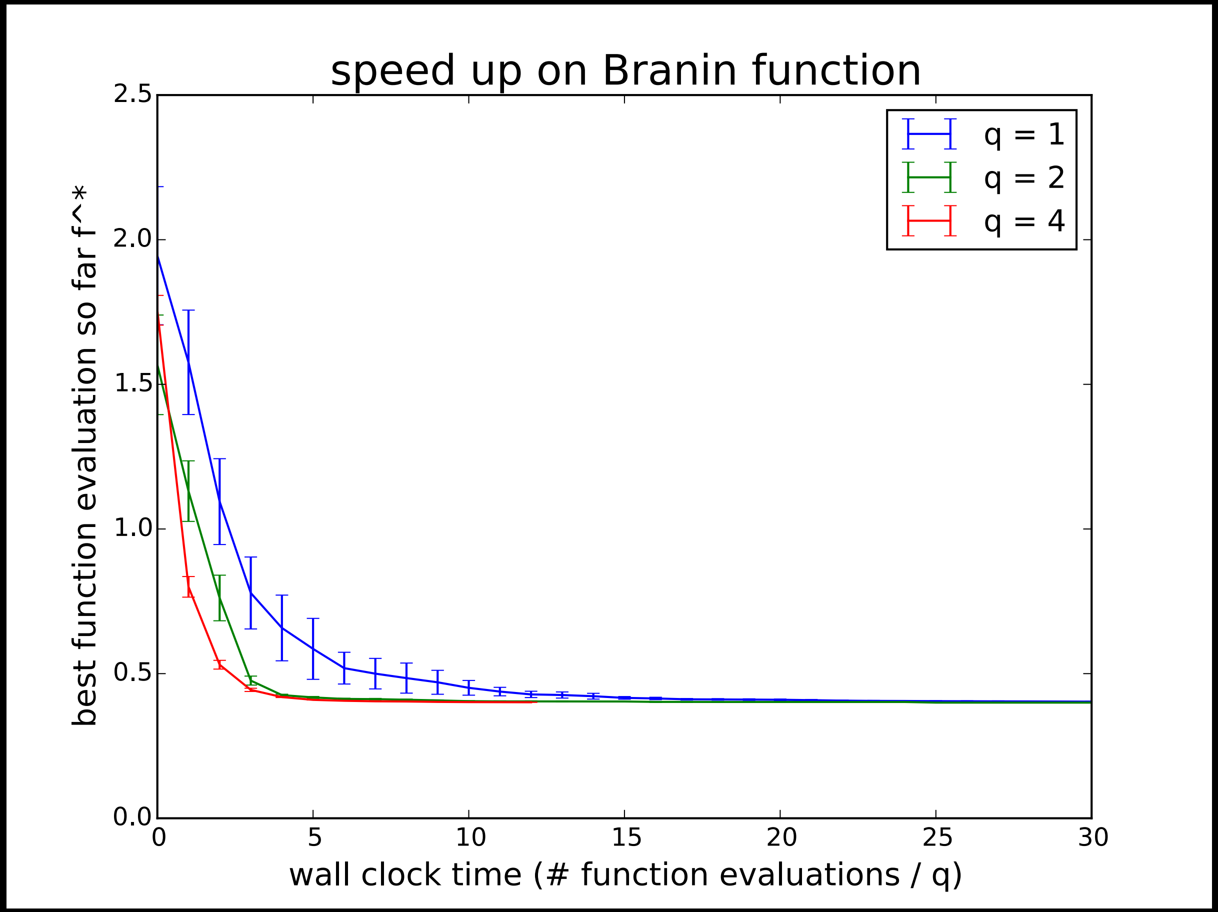


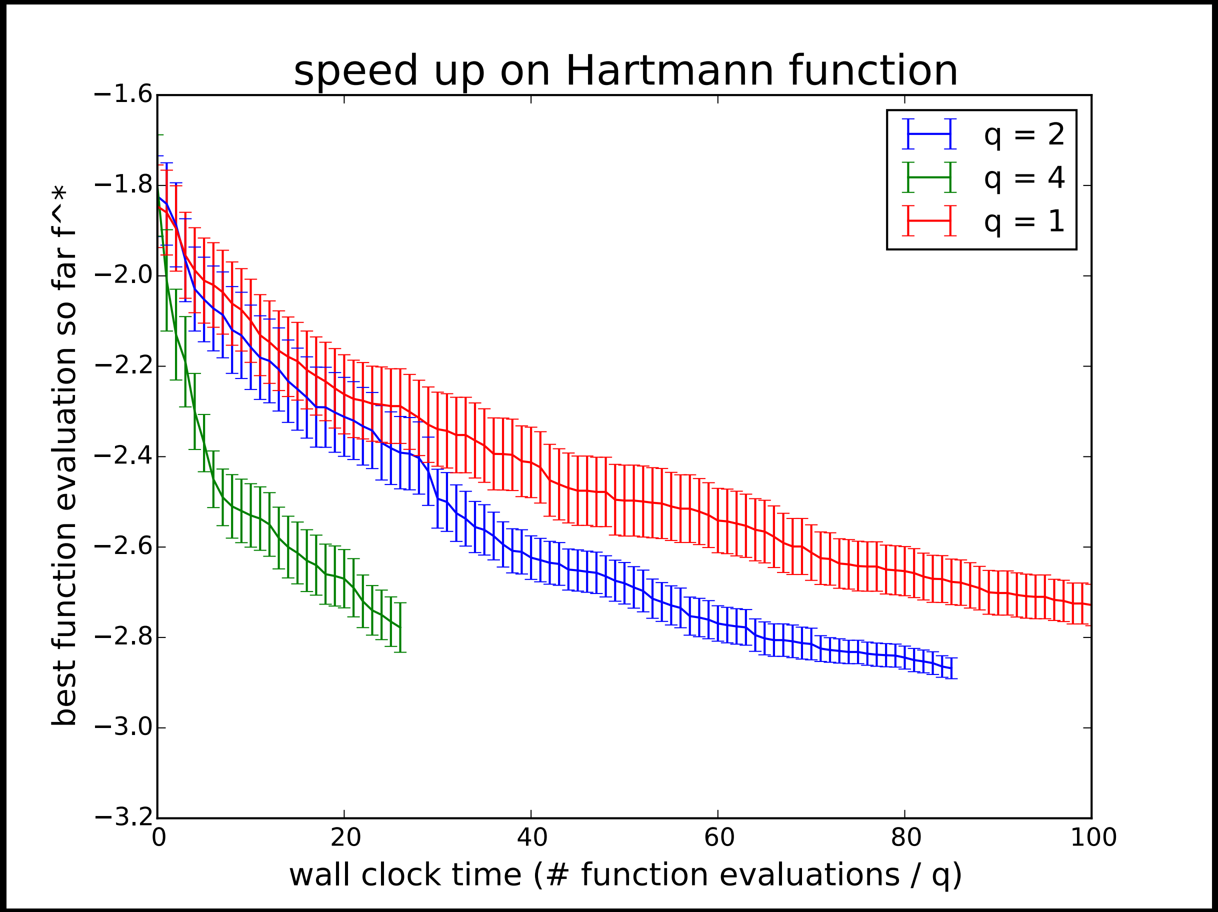
Sample EI and grad EI
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0

Animation



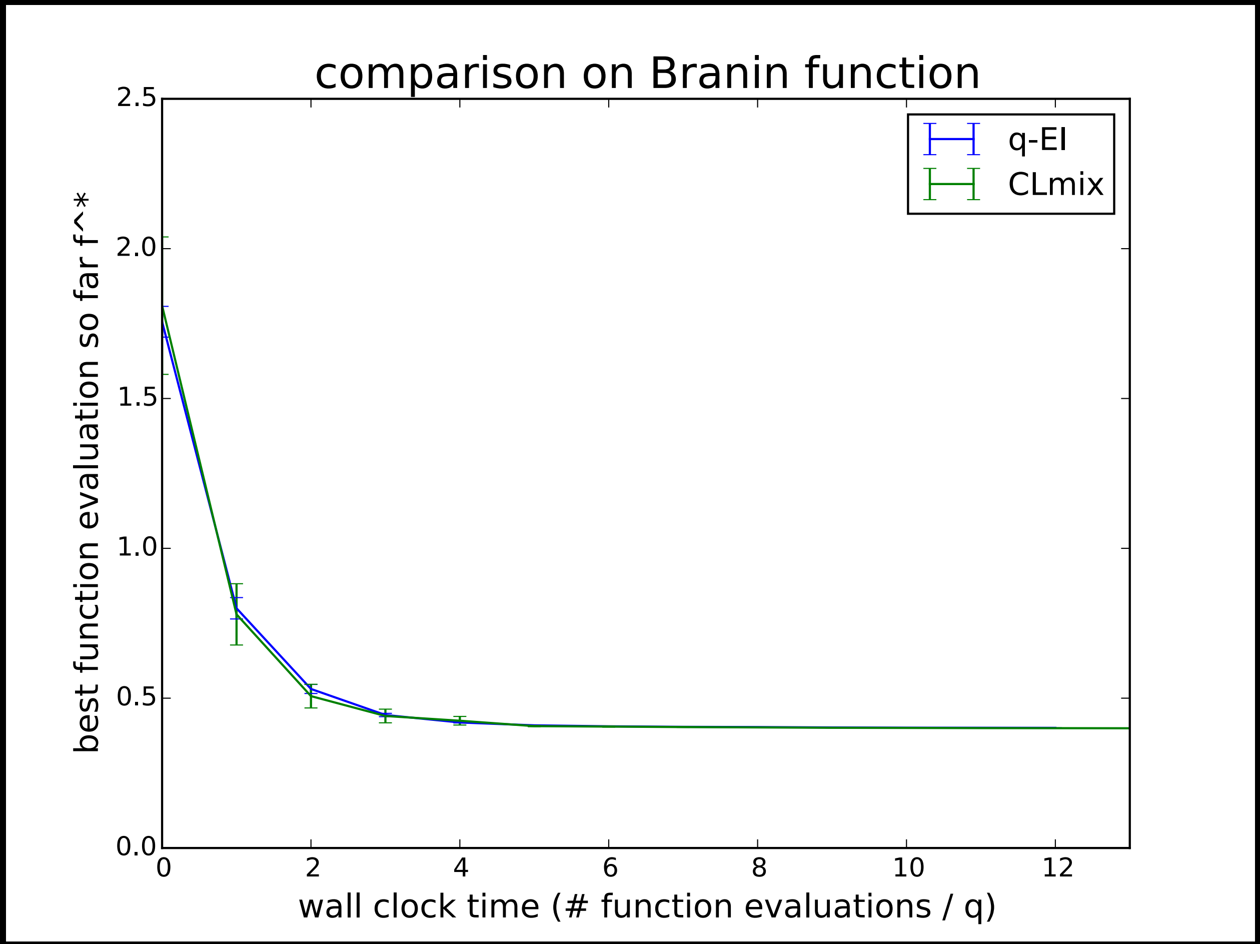
The method works: adding parallelism improves performance

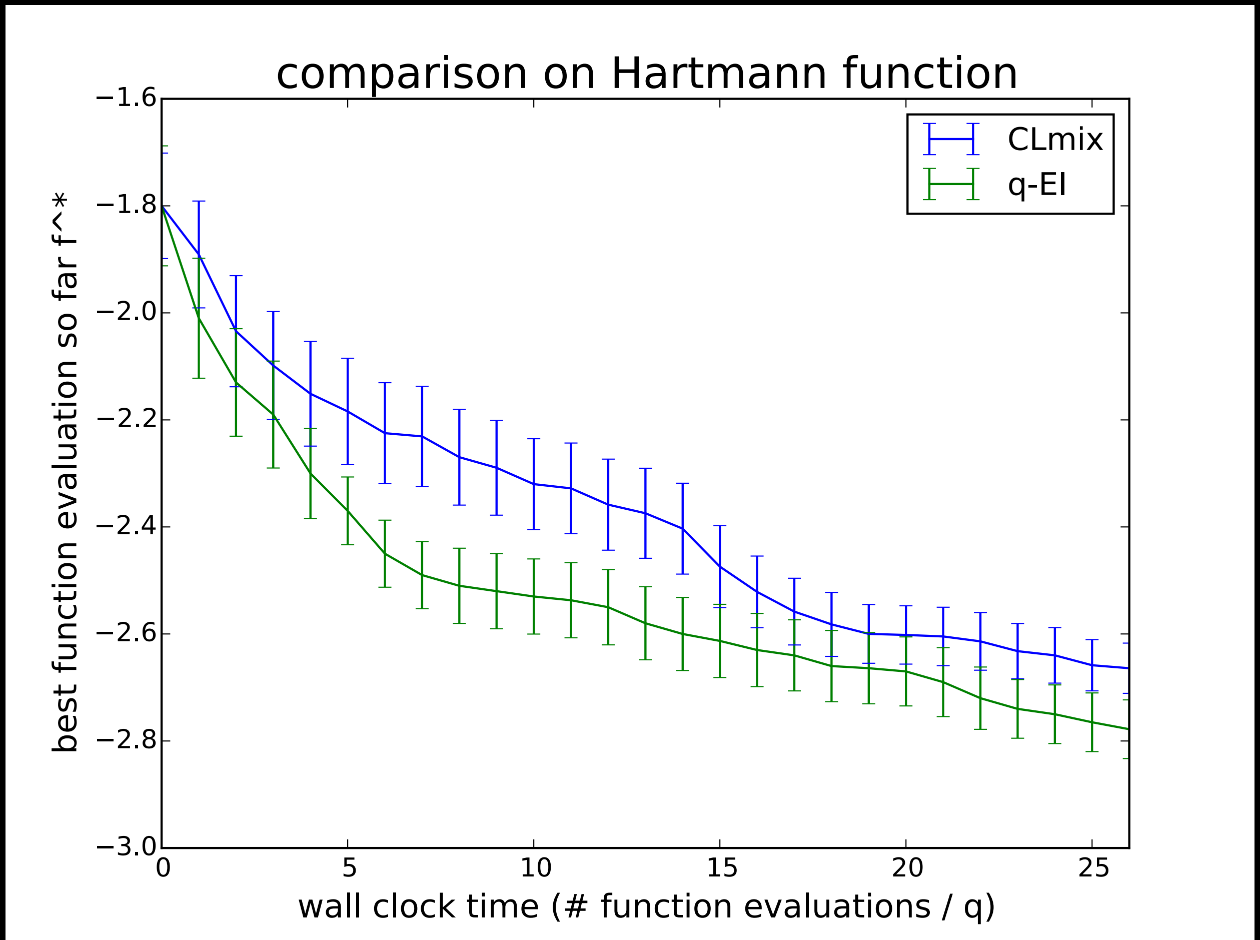




- q=1 (one thread) is the EGO method of [Jones et al., 1998]

The method works: it outperforms an approximation to the Bayes-optimal procedure





- Constant Liar (CL) is a class of algorithms proposed by Chevalier & Ginsbourger 2013.
- CL-mix is the best of the CL algorithms.

Our procedure is only Bayes-optimal for a single batch

- If we do just one batch, then our procedure is Bayes-optimal.
- If we run many batches, starting each new batch after the previous one completes, then our procedure is not optimal.

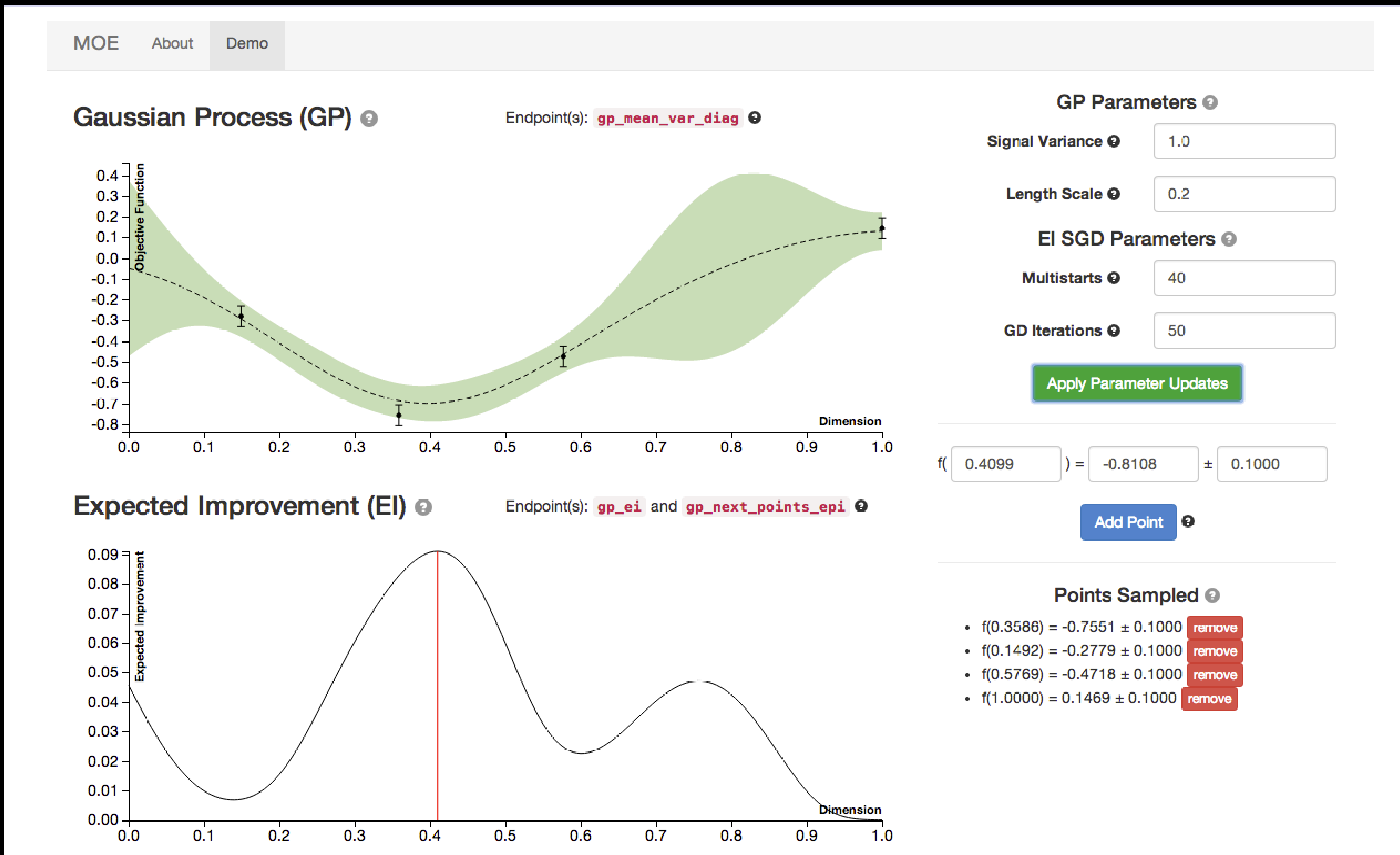
Finding the Bayes-optimal **multi-batch** procedure is hard

- The optimal procedure for $N>1$ batches is the solution to a partially observable Markov decision process (POMDP).
- This is well-understood theoretically, but very hard computationally.
    - The amount of memory required is exponential in d (the problem dimension), q (the batch size), and N (the number of batches).

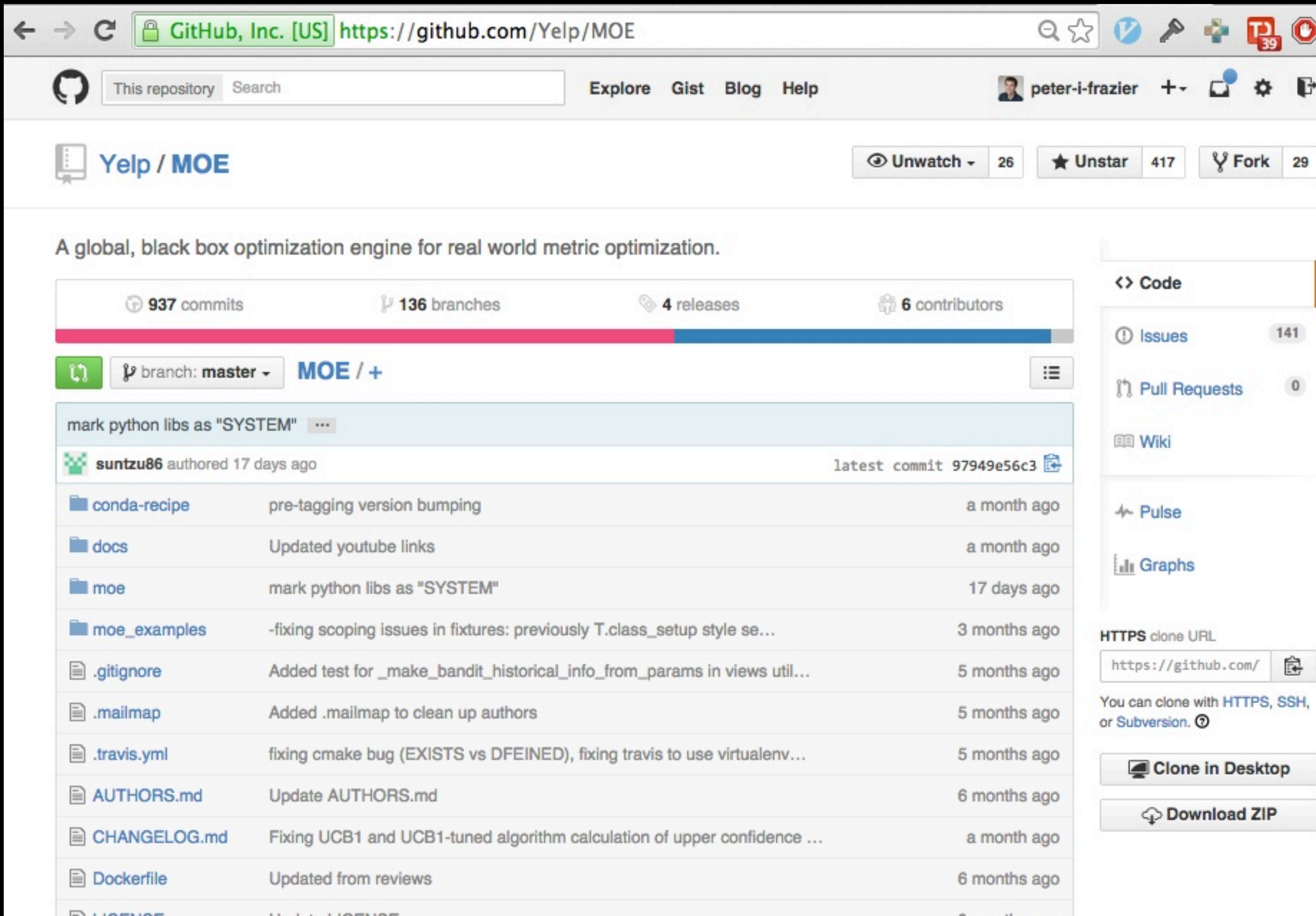
We have found Bayes-optimal multi-batch procedures for other related learning problems

- We have found Bayes-optimal multi-batch procedures, or upper bounds on their value, for these related problems:
  - multiple comparisons [Xie and F., 2013, Hu, F., Xie 2014]
  - stochastic root-finding [Waeber, F., Henderson 2013]
  - ranking & selection (pure exploration MAB) [Xie and F., 2013]
  - information filtering [Zhao and F., 2014]
  - object localization [Jedynak, F., Sznitman, 2012]

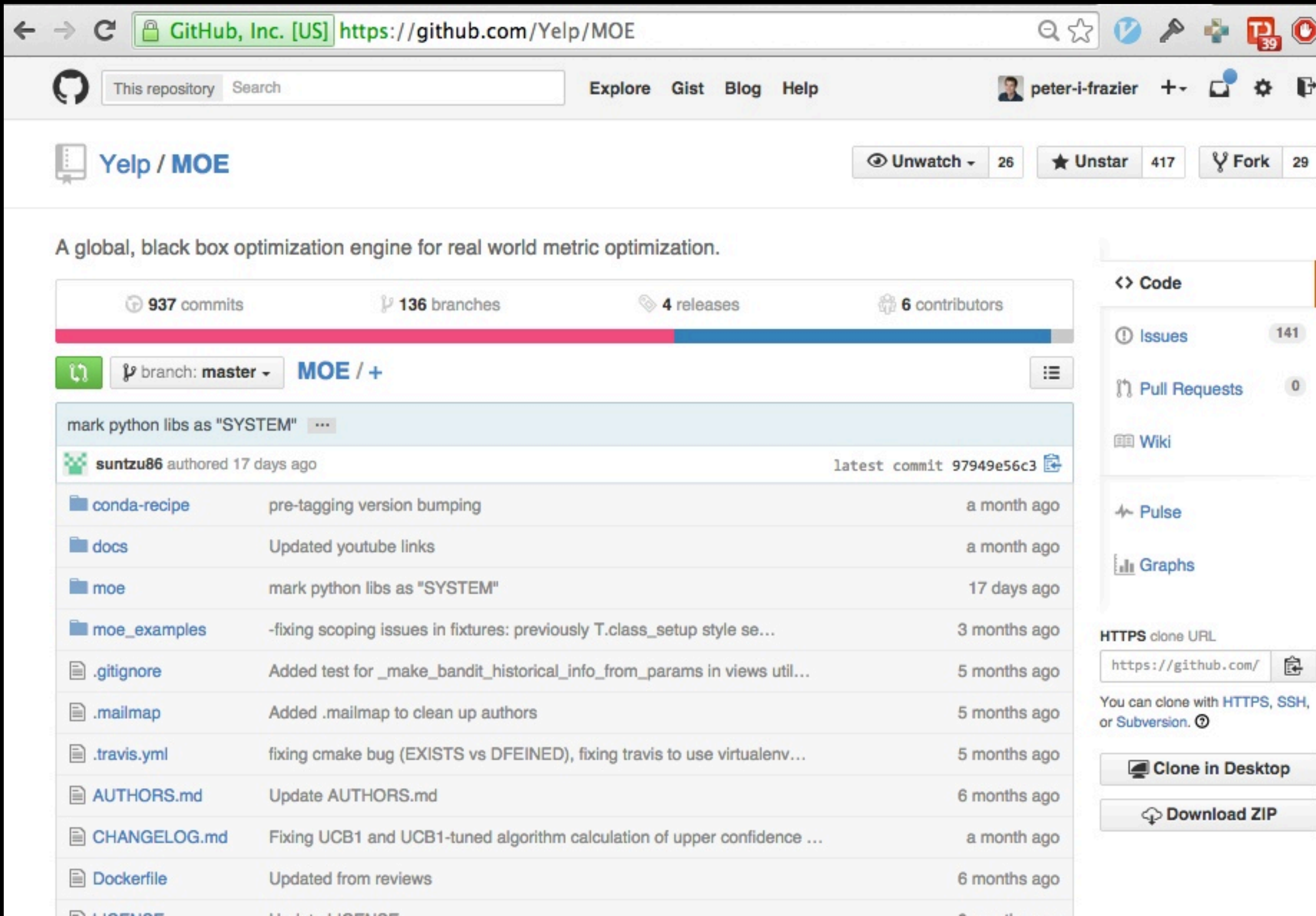
With Yelp, we made a high-quality implementation of some of these methods, called MOE (Metrics Optimization Engine)



MOE is open source

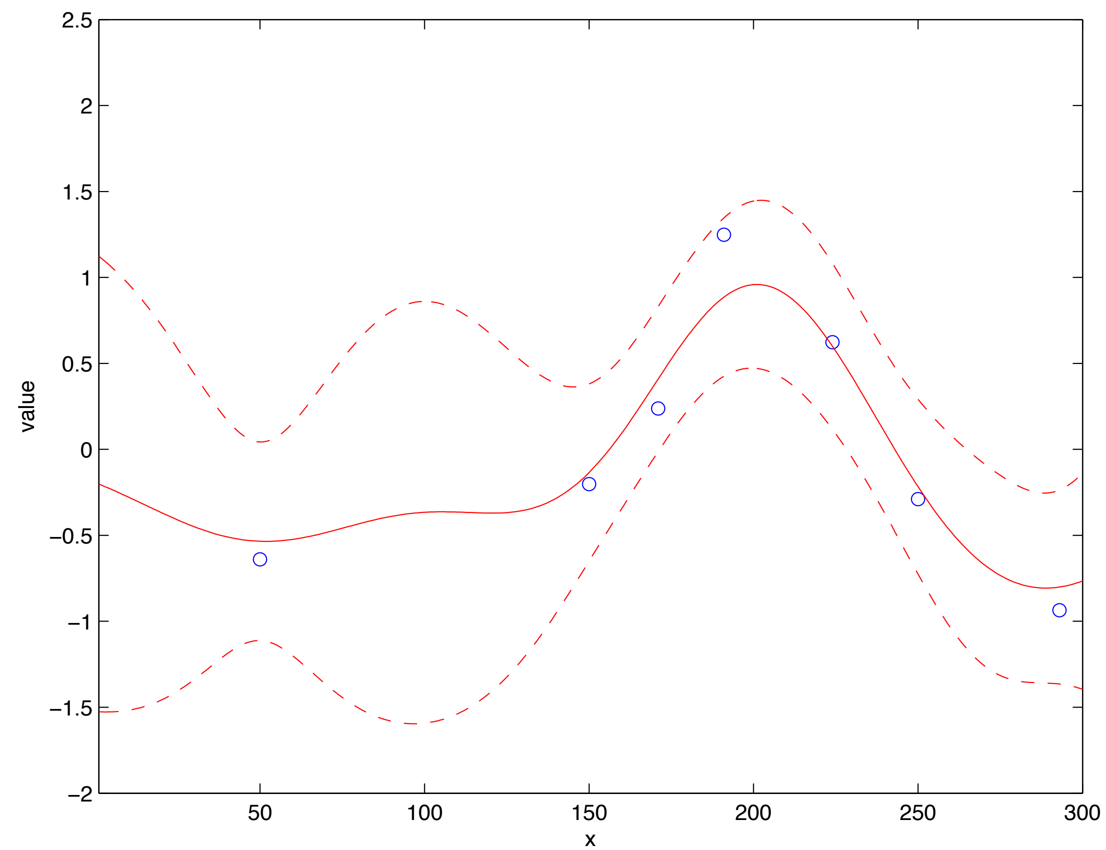


MOE is in production at Yelp & Netflix, and is being considered by Wayfair, Tripadvisor, & others...



This q-EI method can be used in the noisy case, but it loses its decision-theoretic motivation

- We use Gaussian process regression with normally distributed noise.
- The red line is the posterior mean, $\mu_n(x) = \mathbb{E}_n[f(x)]$
- The largest posterior mean is $\mu_n^* = \max_{i=1,\ldots,n} \mu_n(\vec{x}^{(m)})$.



- We use $\mathrm{EI}_n(\vec{x}_1, \ldots, \vec{x}_q) = \mathbb{E}_n\left[\left(\max_{m=1,\ldots,q} \mu_{n+1}(\vec{x}_i) - \mu_n^*\right)^+\right]$
- This ignores that $\mu_{n+1}(x) \neq \mu_n(x)$ for previously evaluated x.
- A more principled approach is possible (e.g., generalize knowledge gradient method to multiple evaluations), but we haven't done it yet.

Thanks!
Any Questions?

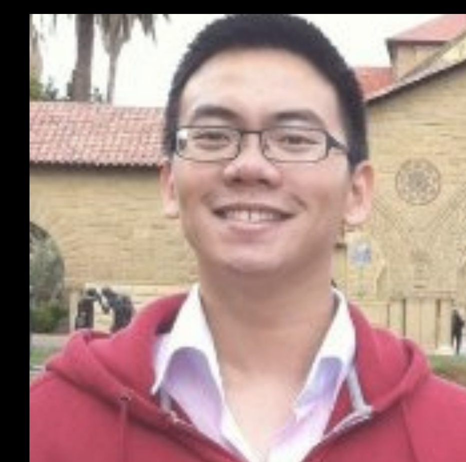
- This was joint work with:

Scott Clark
Cornell PhD '12
Yelp, SigOpt

Eric Liu
Yelp, SigOpt

Jialei Wang
Cornell PhD student
Yelp intern

Deniz Oktay
MIT undergraduate
Yelp intern



Norases Vesdapunt
Stanford undergraduate
Yelp intern