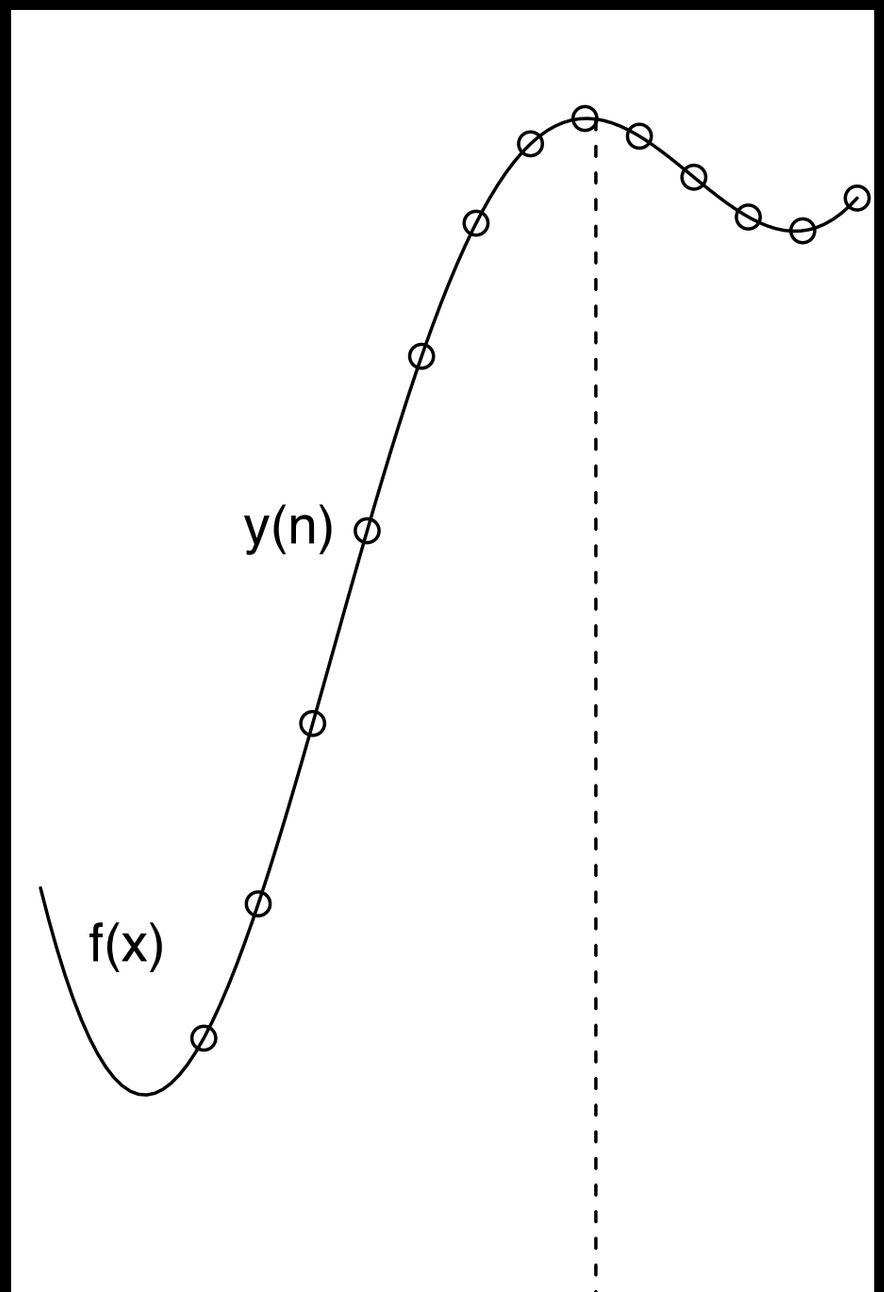
Bayesian Optimization

Peter Frazier
Associate Professor, Operations Research & Information Engineering, Cornell
Staff Data Scientist, Uber

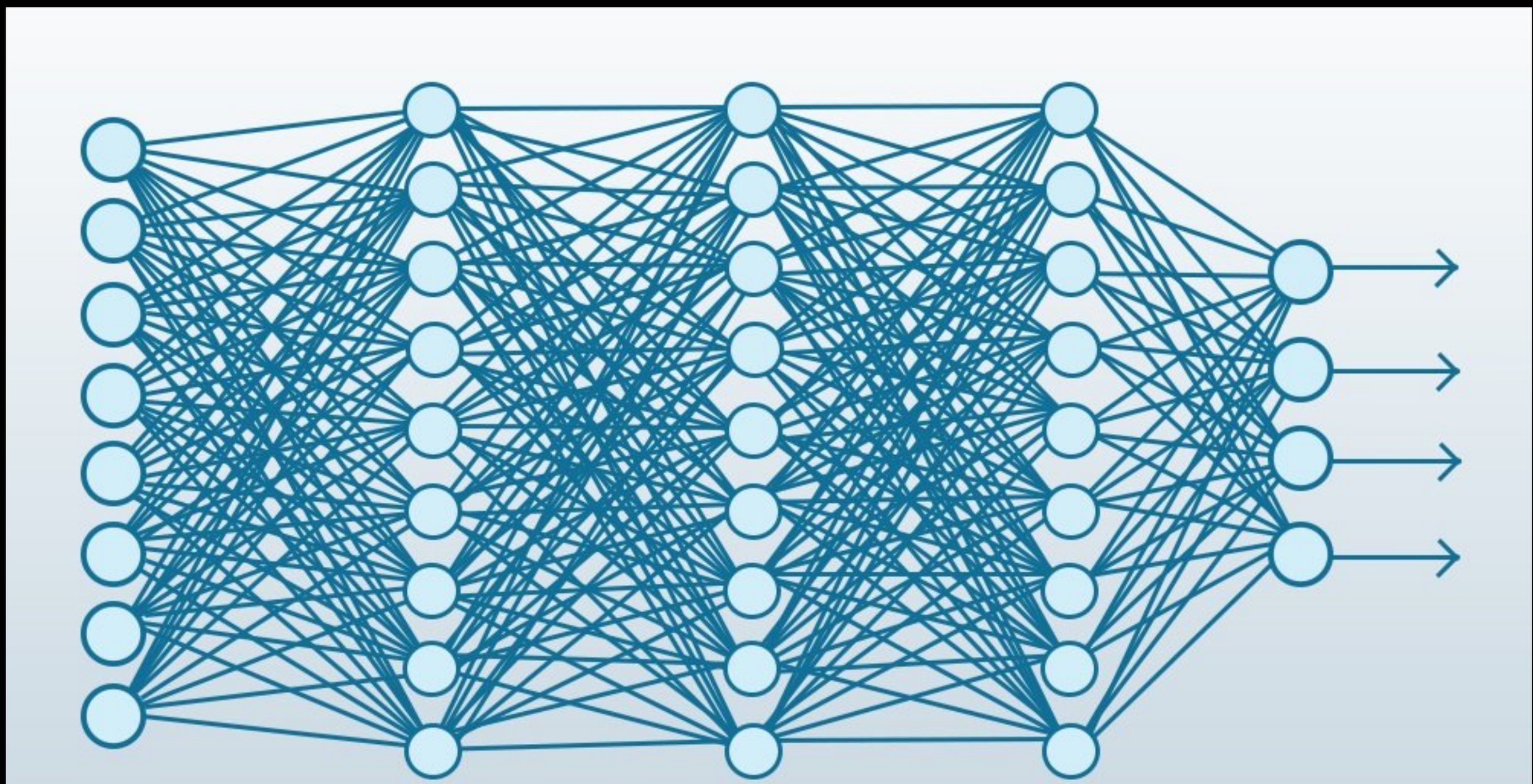
For details & links to code, see
P. Frazier, "A Tutorial on Bayesian Optimization" https://arxiv.org/abs/1807.02811

This is the standard problem in Bayesian Optimization



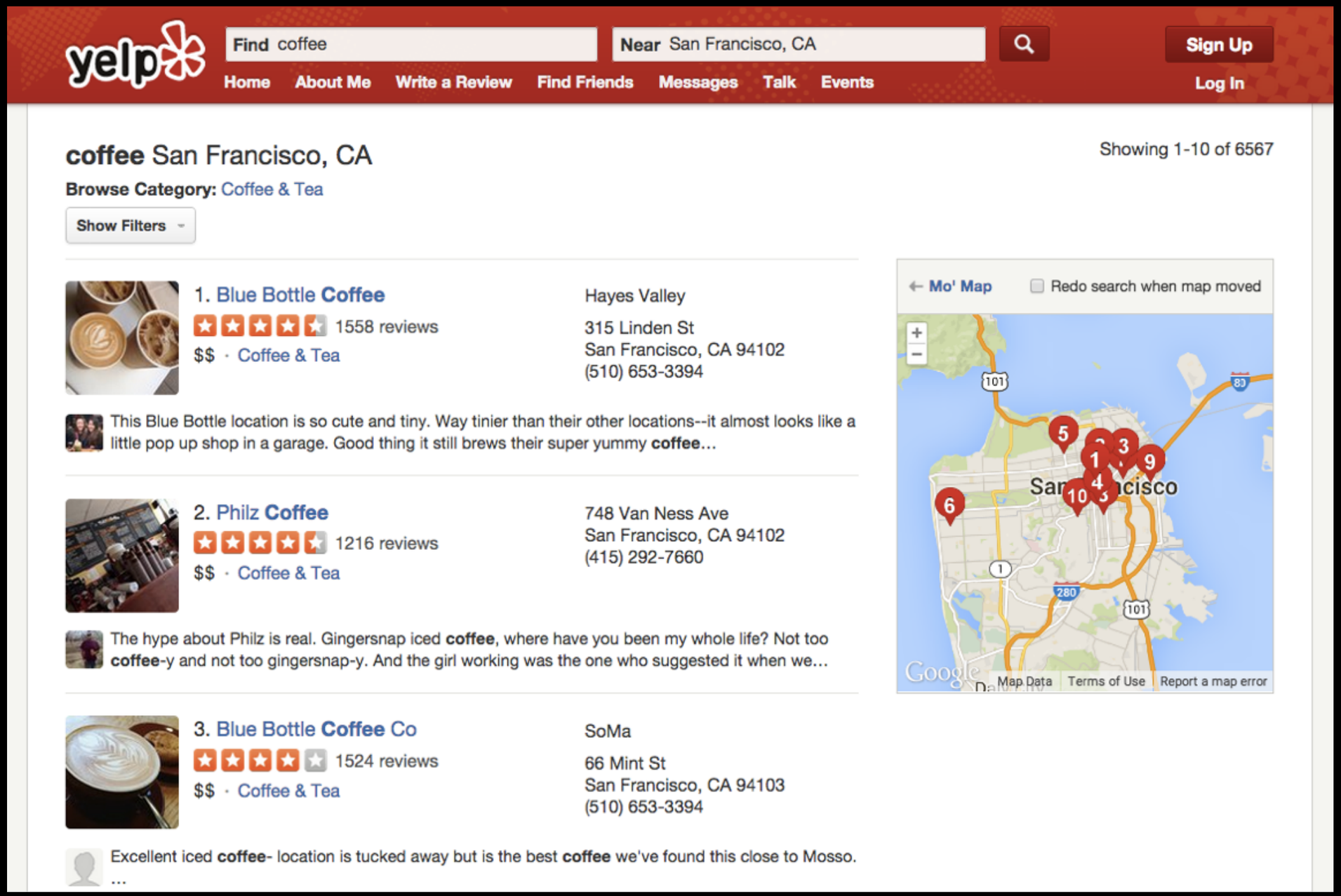
- We'd like to optimize F : $\mathbb{R}^d \rightarrow \mathbb{R}$, where d < 20.
- F's feasible set A is simple, e.g., box constraints.
- F is continuous but lacks special structure, e.g., concavity, that would make it easy to optimize.
- F is derivative-free: evaluations do not give gradient information.
- 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.

Optimization of expensive functions arises when fitting machine learning models

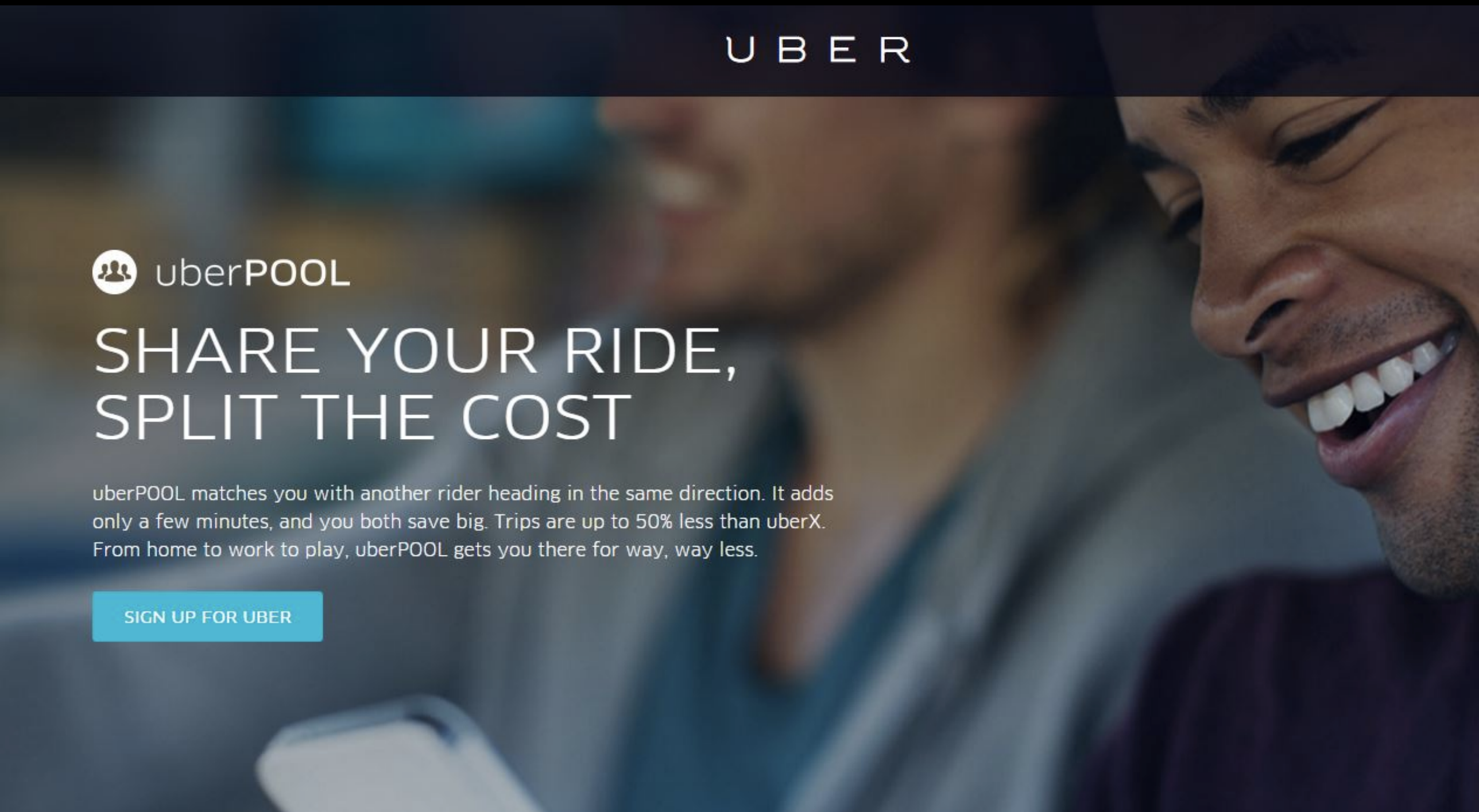


Optimization of expensive functions arises when tuning algorithms via backtesting

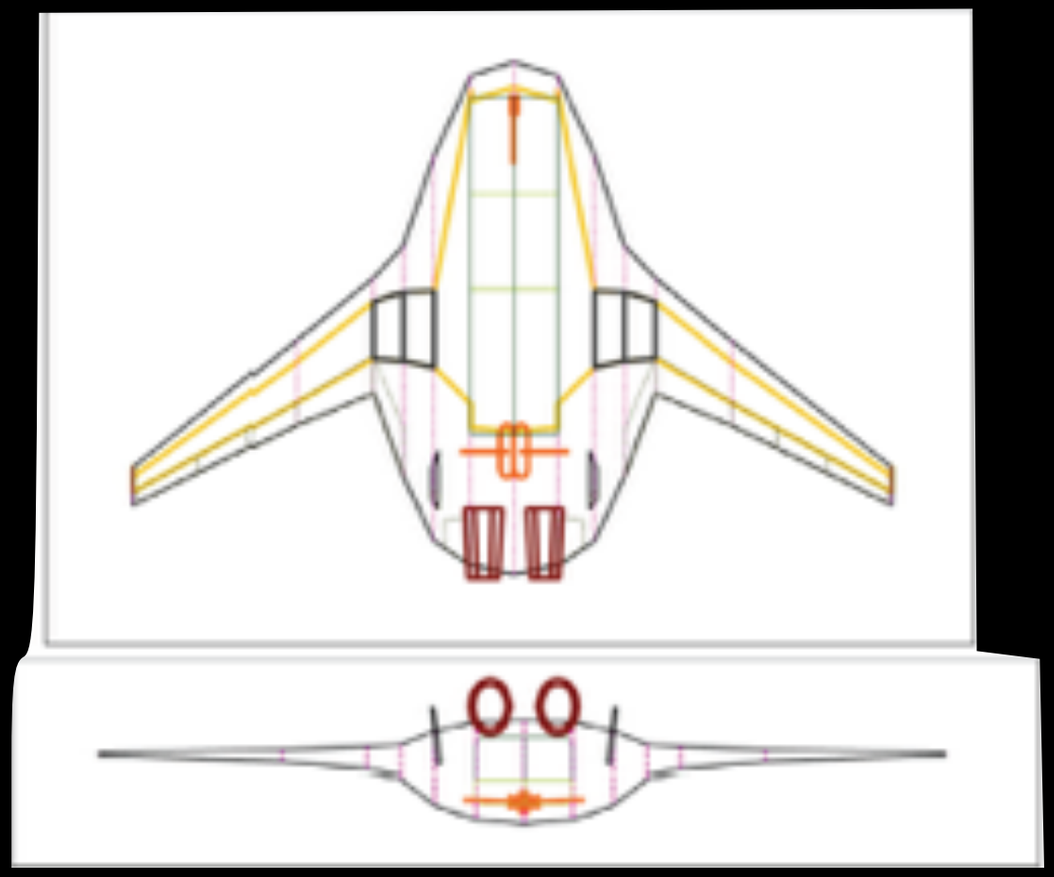
Optimization of expensive functions arises when tuning websites with A/B testing

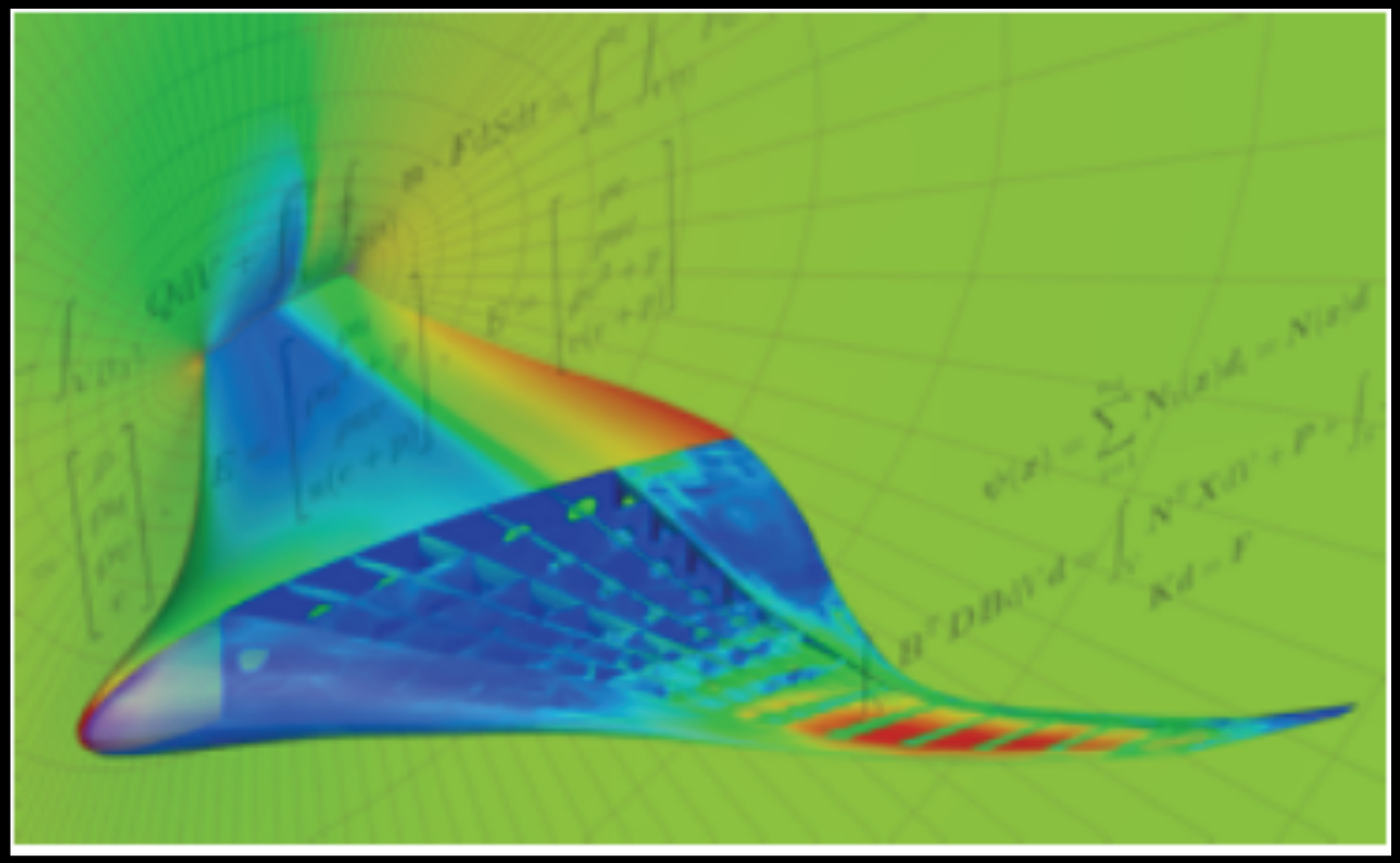


Optimization of expensive functions arises when tuning transportation markets



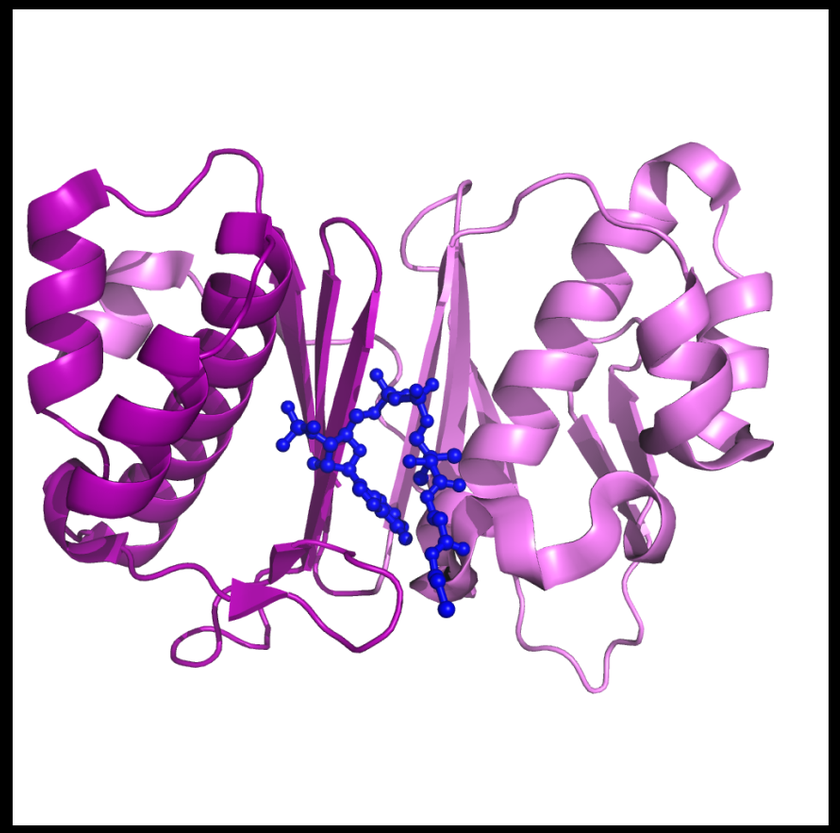
Optimization of expensive functions arises when optimizing physics-based models



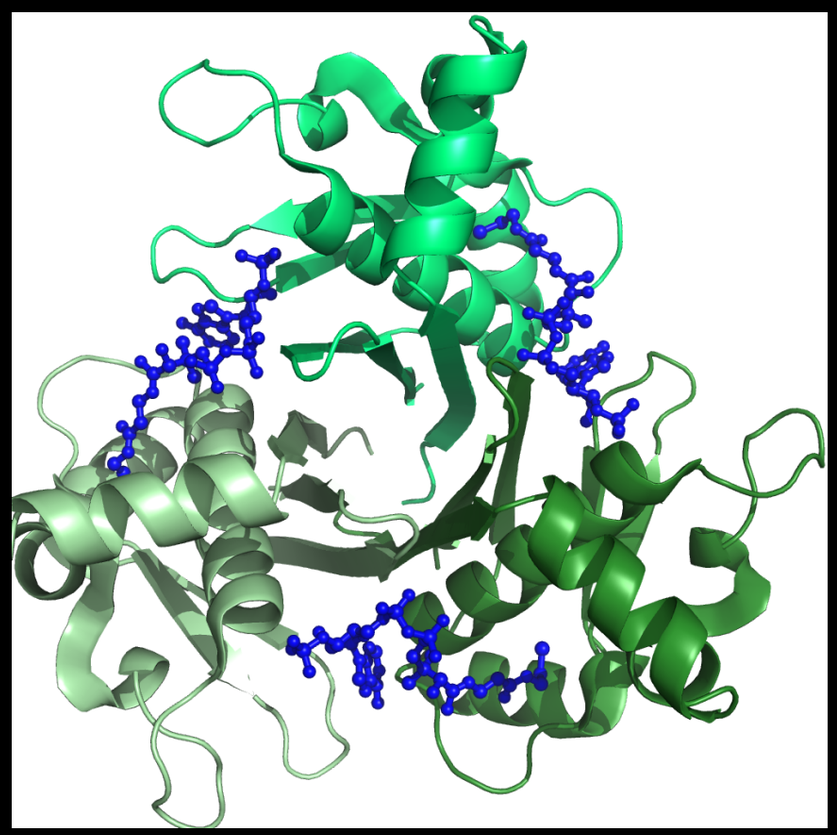


ongoing work with Karen Wilcox (MIT), Joaquim Martins (Michigan), Doug Allaire (Texas A&M), David Wolpert (Sante Fe Institute), Youssef Marzouk (MIT), and Marc Mignolet (Arizona State University),

Optimization of expensive functions arises in drug and materials discovery



Sfp
(a protein-modifying enzyme)



AcpS
(another protein-modifying enzyme)

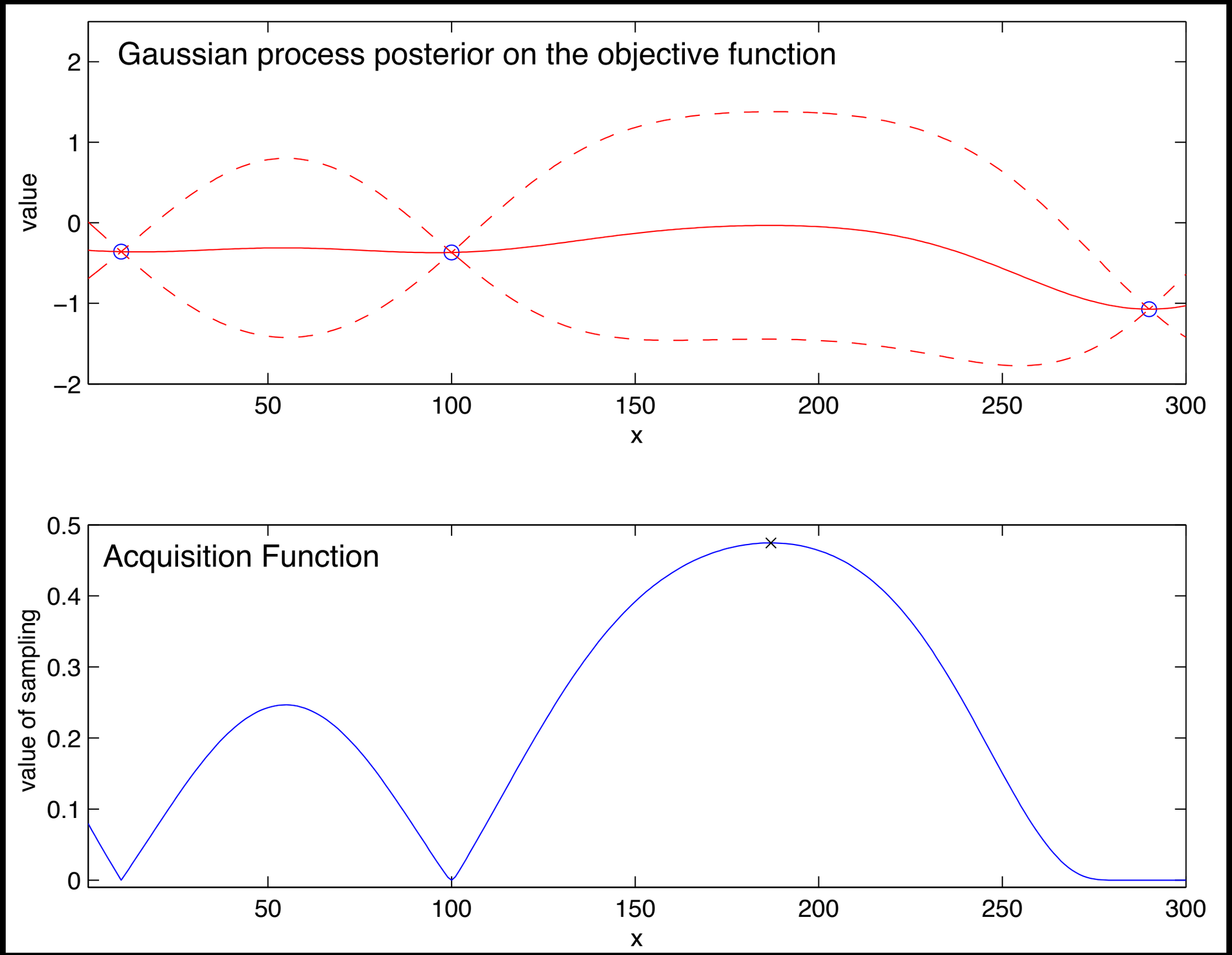
ongoing work with Mike Burkart and Nathan Gianneschi, UCSD

Bayesian Optimization is one way to optimize expensive functions

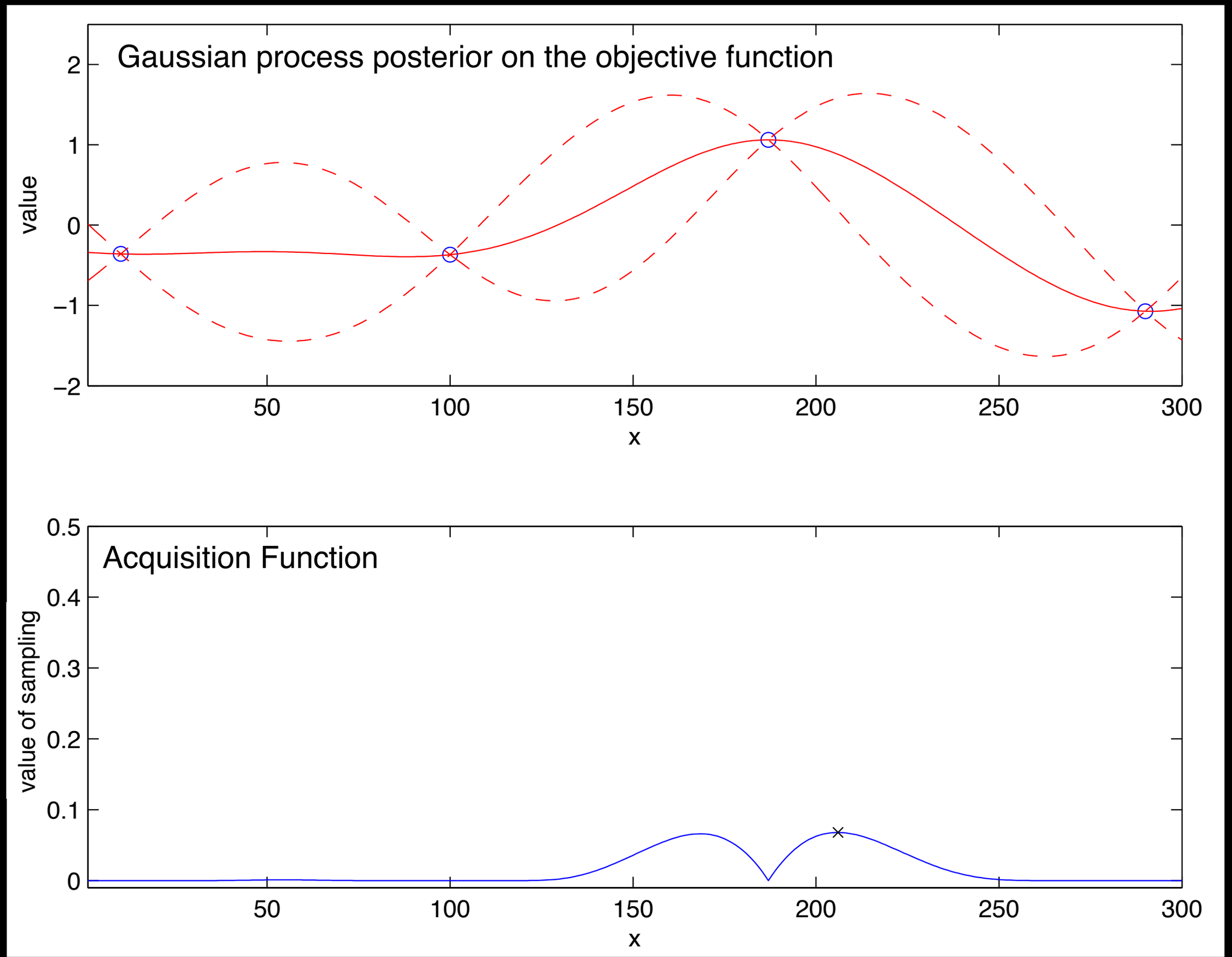
Assume a Bayesian prior on F
(usually a Gaussian process prior)

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

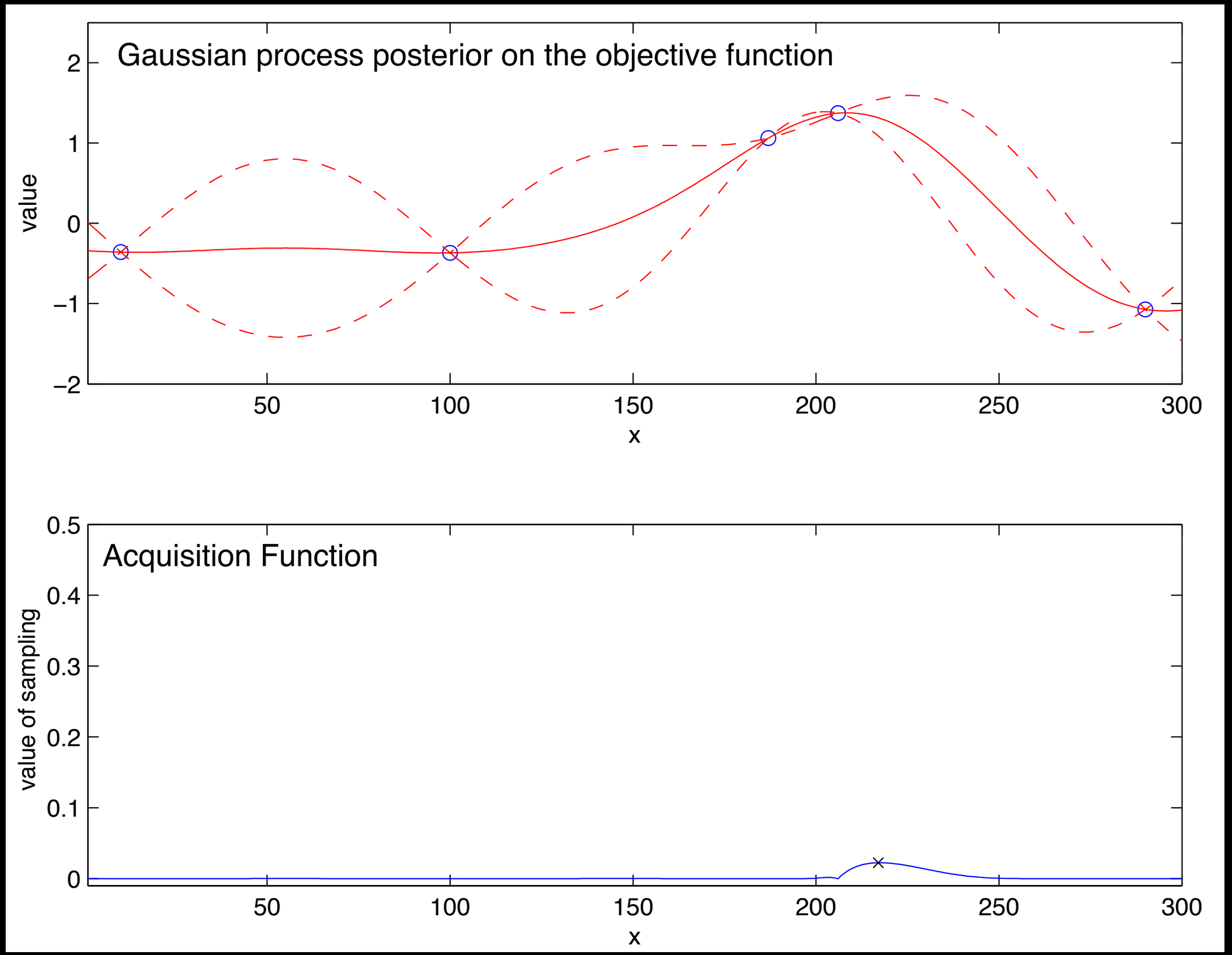
Here's Bayesian optimization, optimizing a 1-dim objective



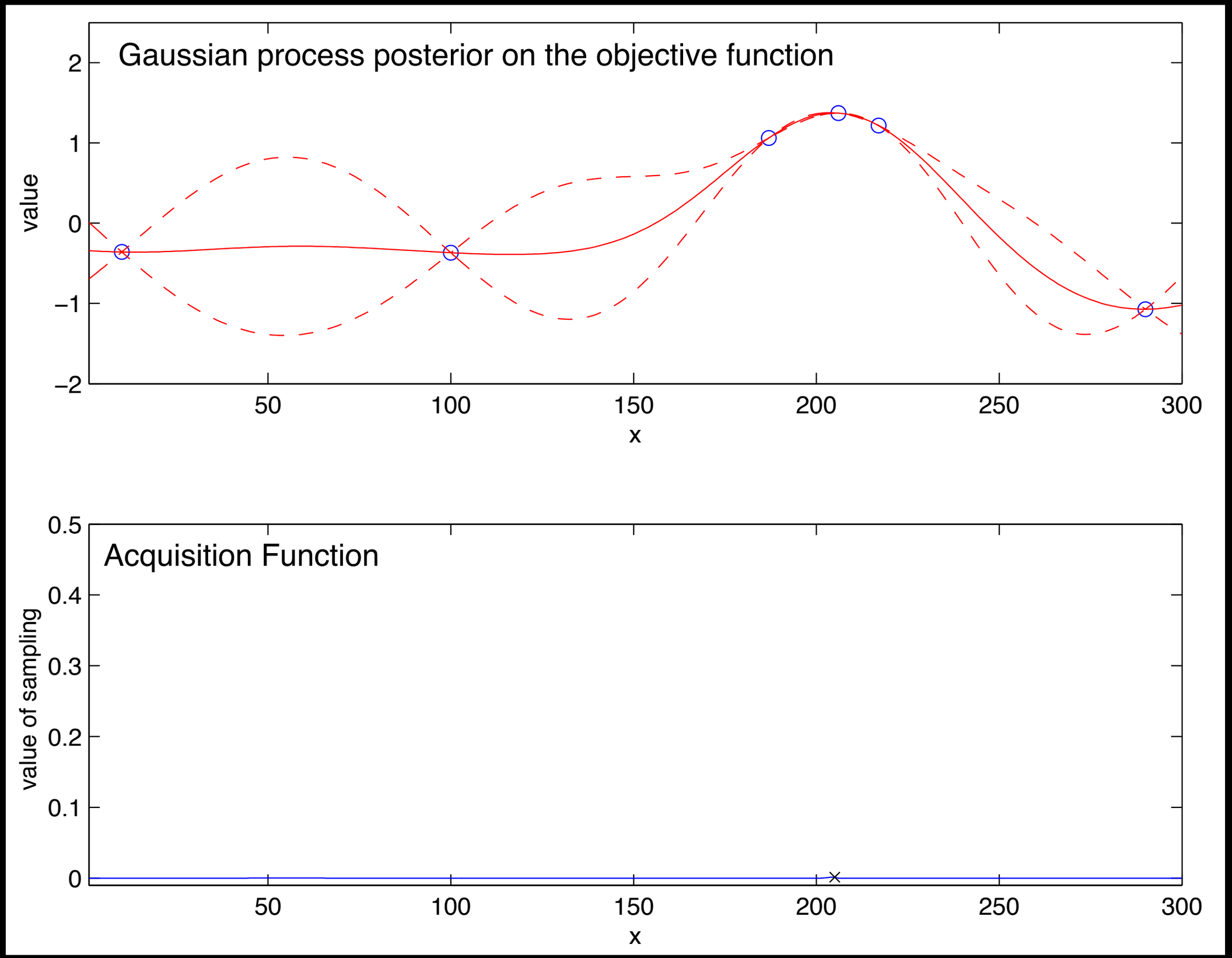
Here's Bayesian optimization, optimizing a 1-dim objective



Here's Bayesian optimization, optimizing a 1-dim objective



Here's Bayesian optimization, optimizing a 1-dim objective



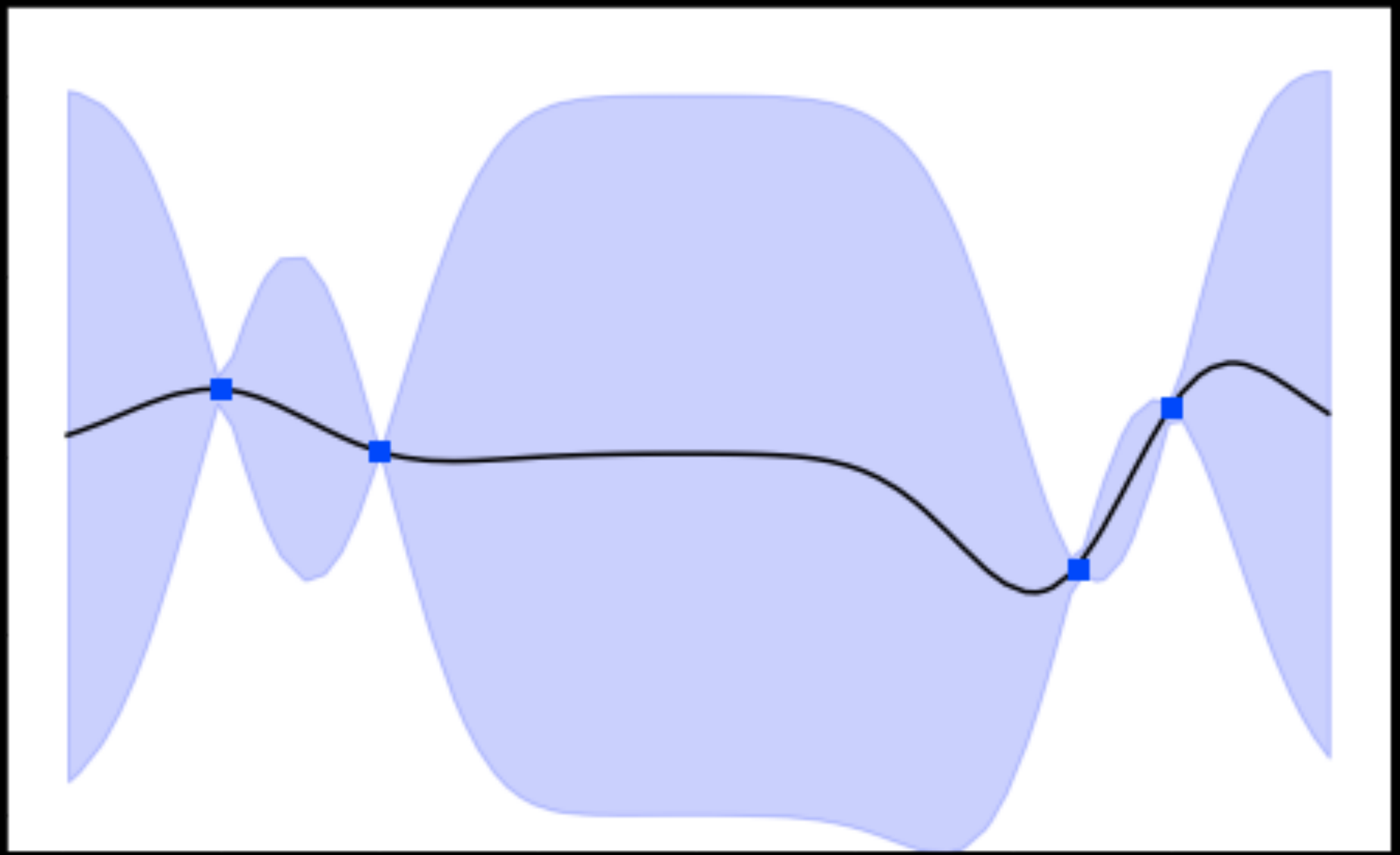
Agenda

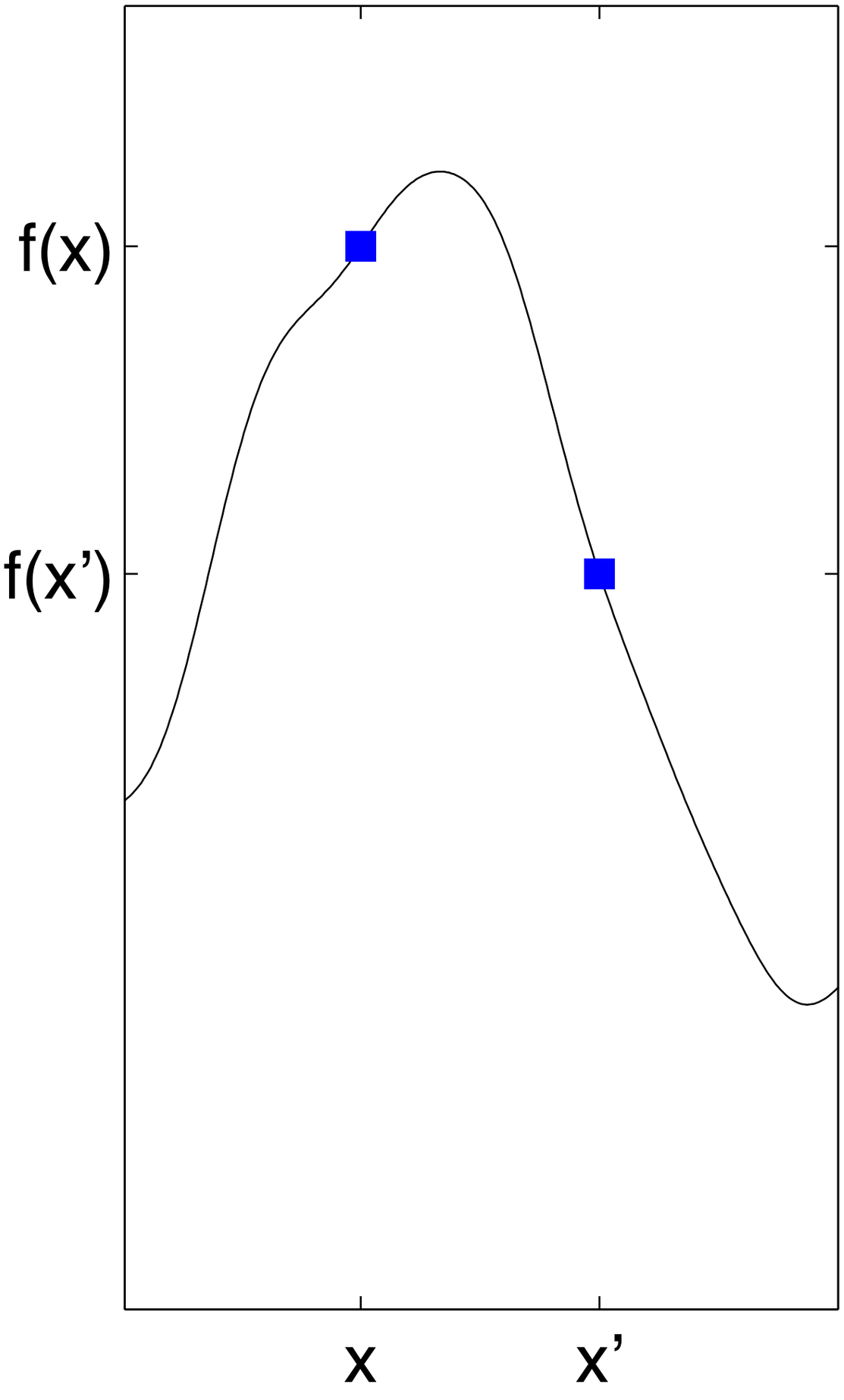
- Gaussian process regression
- Expected improvement
- Beyond the standard problem

Agenda

- **Gaussian process regression**
- Expected improvement (EI)
- Parallel BayesOpt (using EI)
- Parallel BayesOpt with gradients and/or noise (using KG)
- Code & Research directions

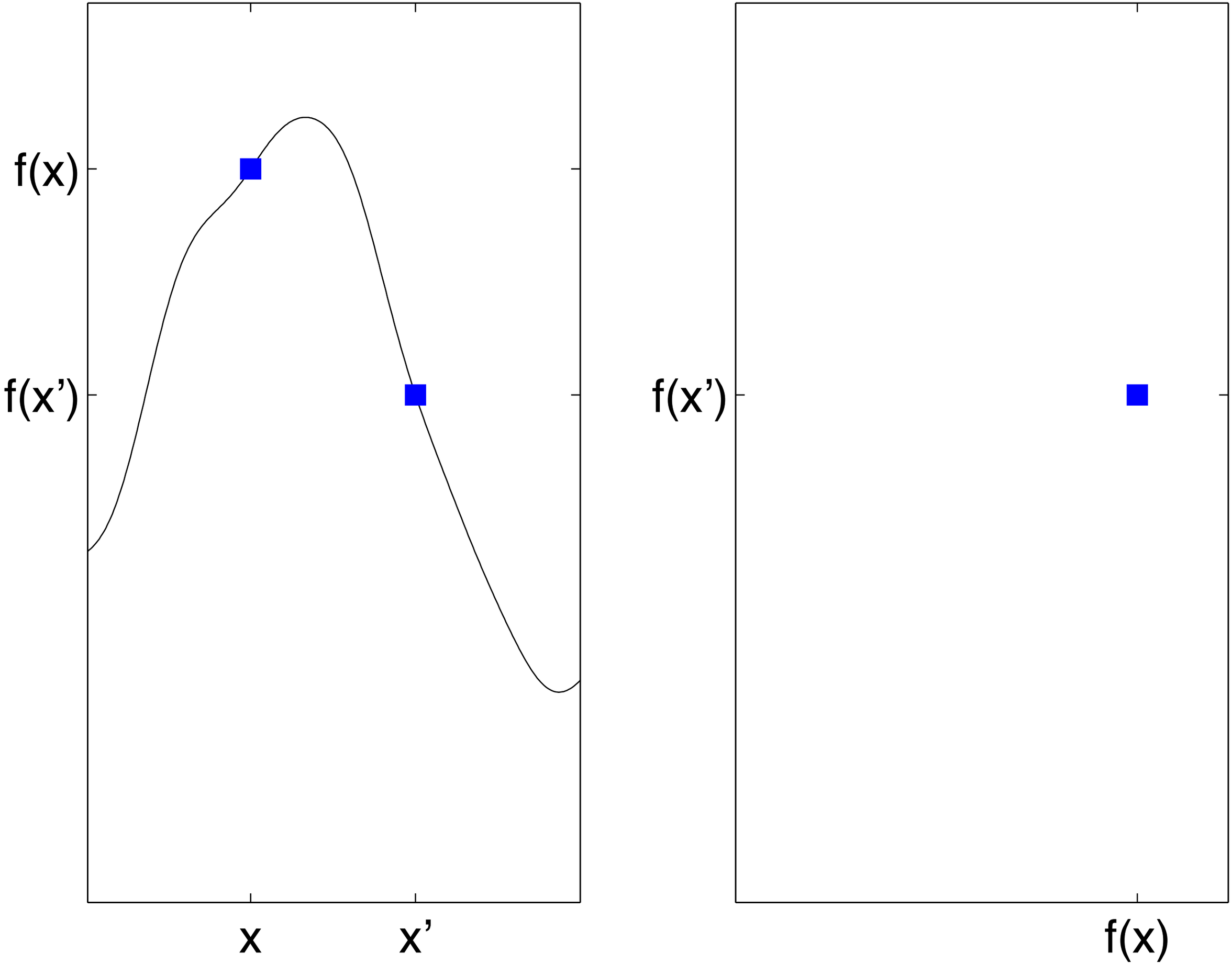
Bayesian optimization usually uses Gaussian process regression





Let's start simply

- Fix 2 points x and x'
- Consider the vector [f(x),f(x')]


f(x)
f(x')
x
x'
f(x')
f(x)

f(x)

f(x')

x

x'

f(x')

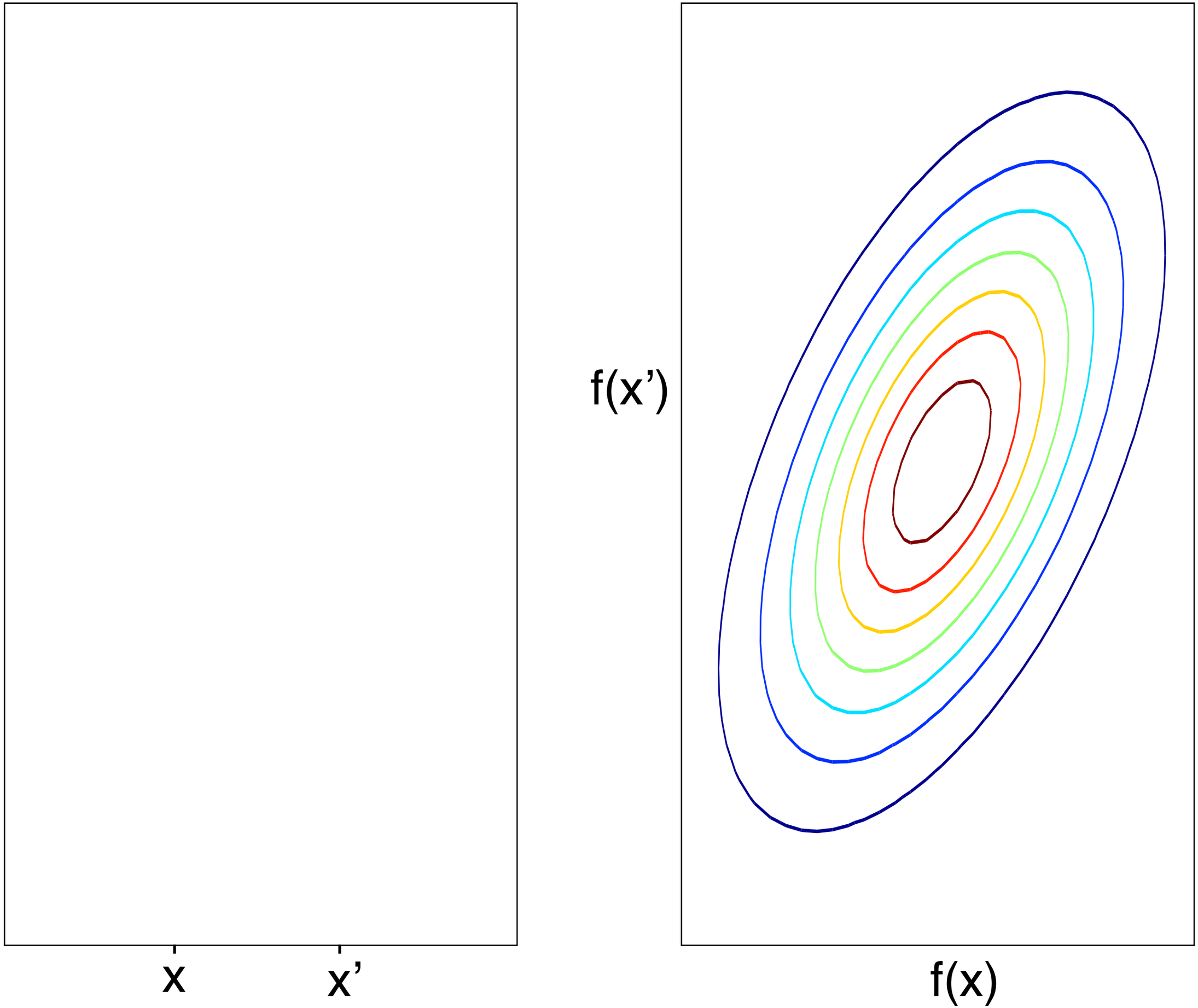
f(x)

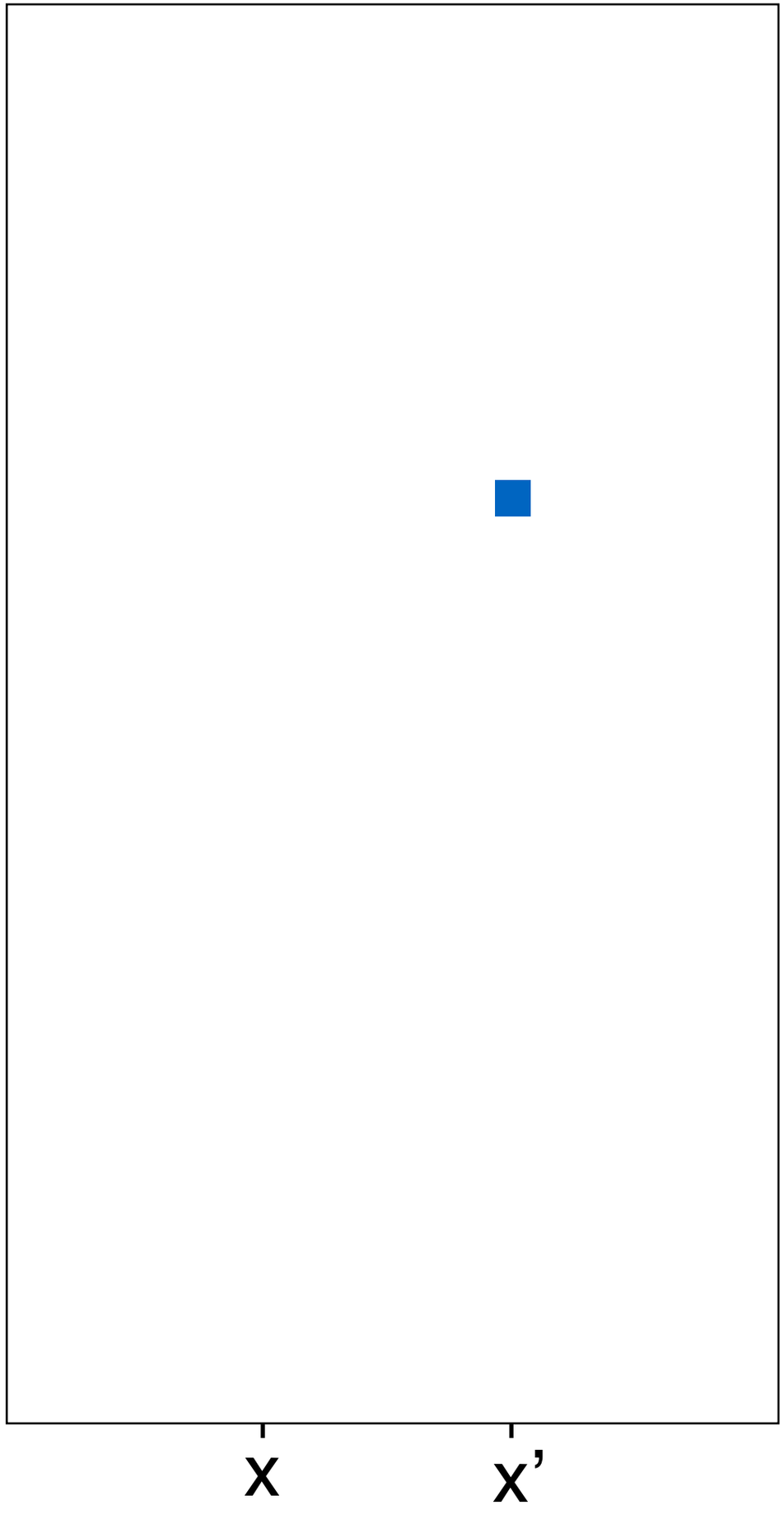
f(x')
f(x)
x
x'
f(x')
f(x)

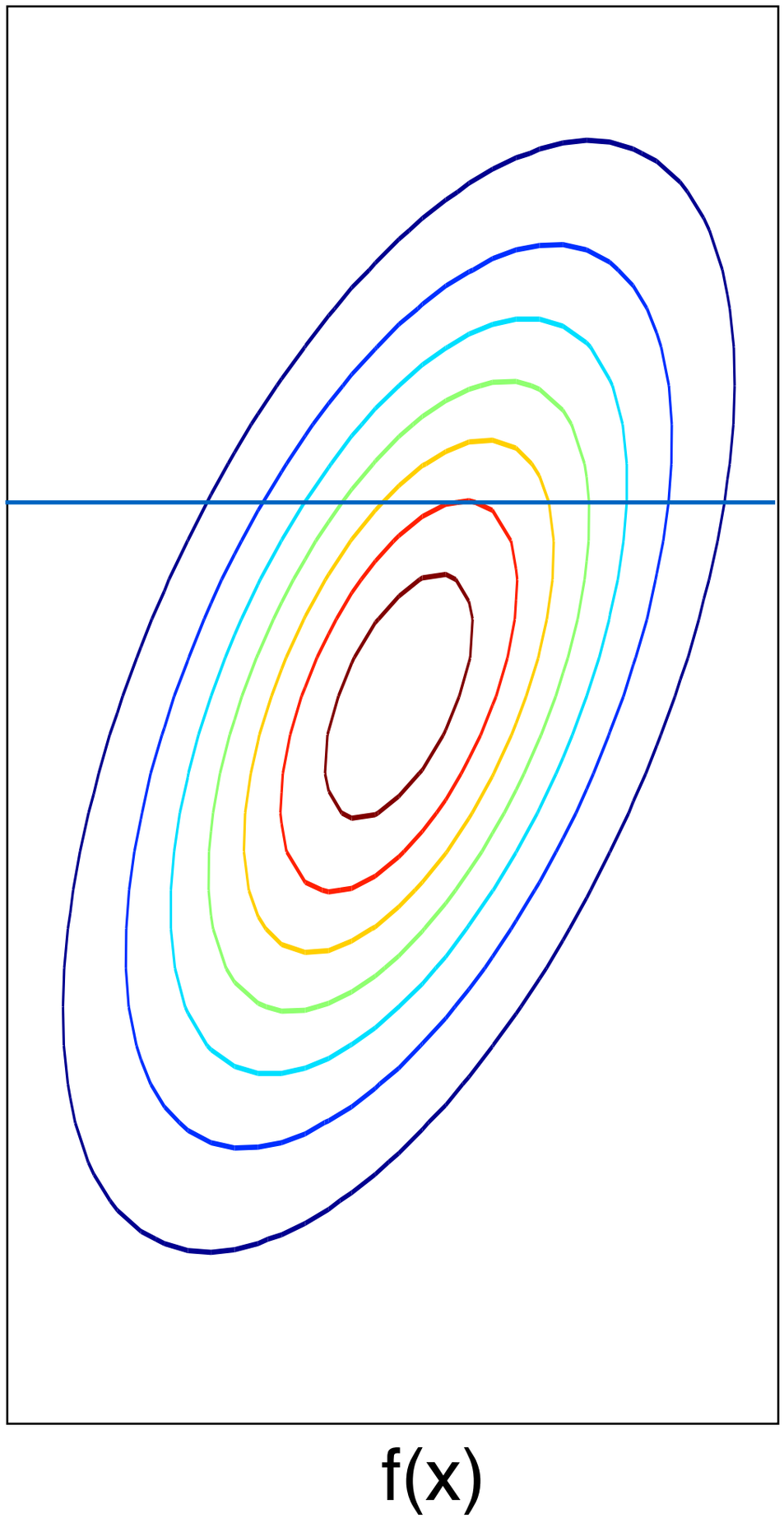
Let's place a multivariate normal prior on [f(x),f(x')]

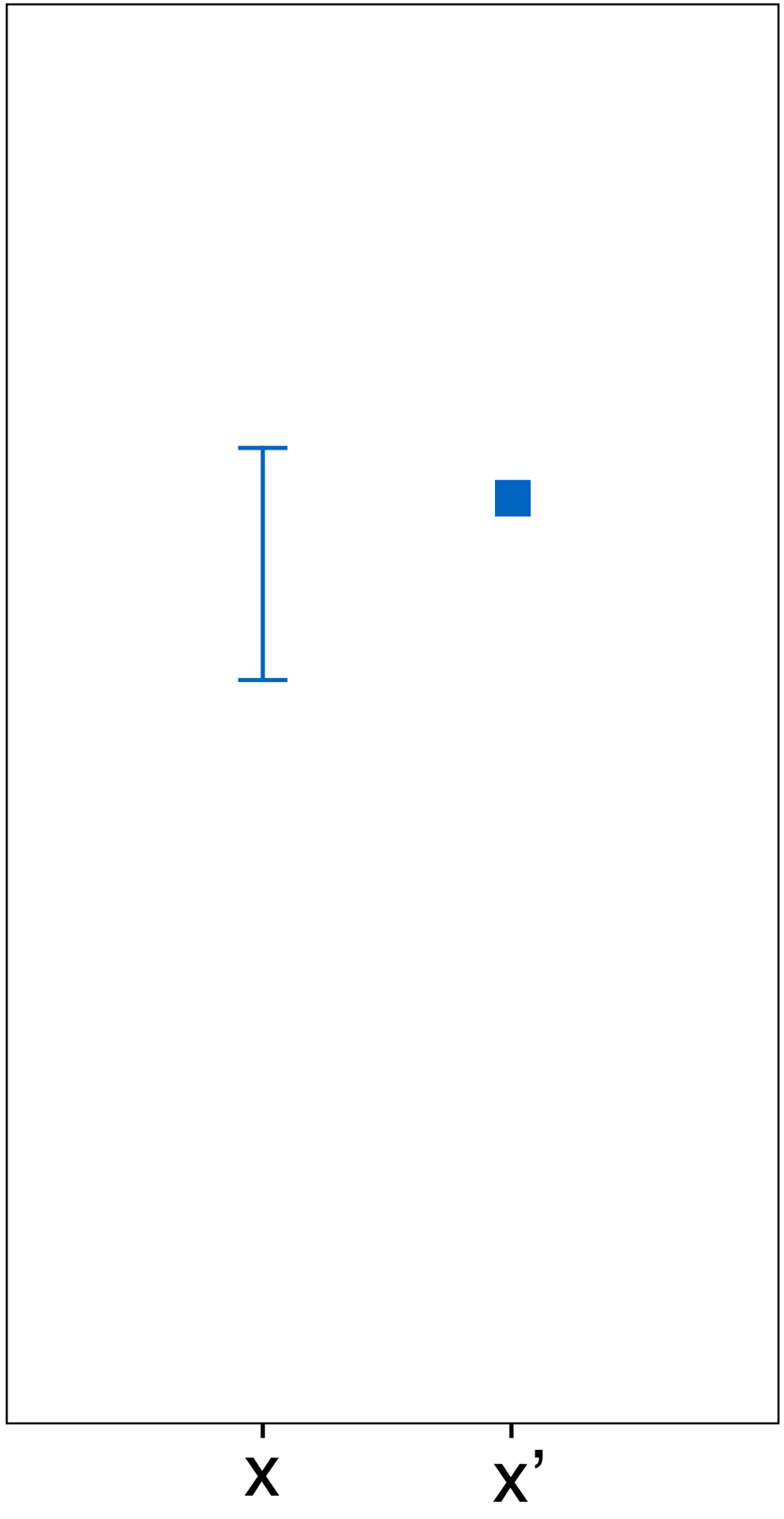
$$\begin{bmatrix} f(x) \\ f(x') \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_0(x) \\ \mu_0(x') \end{bmatrix}, \begin{bmatrix} \Sigma_0(x,x) & \Sigma_0(x,x') \\ \Sigma_0(x',x) & \Sigma_0(x',x') \end{bmatrix} \right)$$

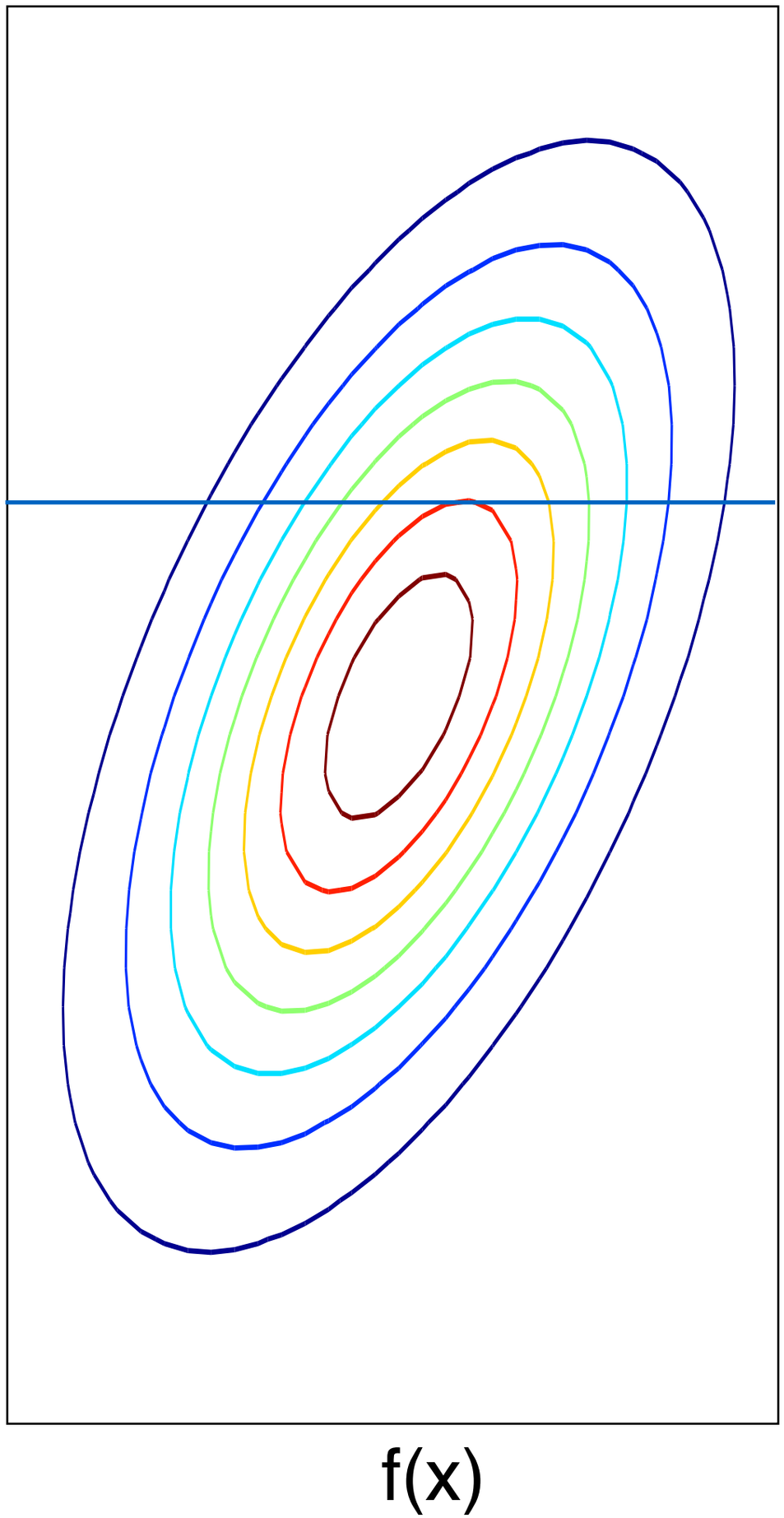
- Σ_0 is a function that decreases with ||x-x'||
- μ is another function, often set to a constant

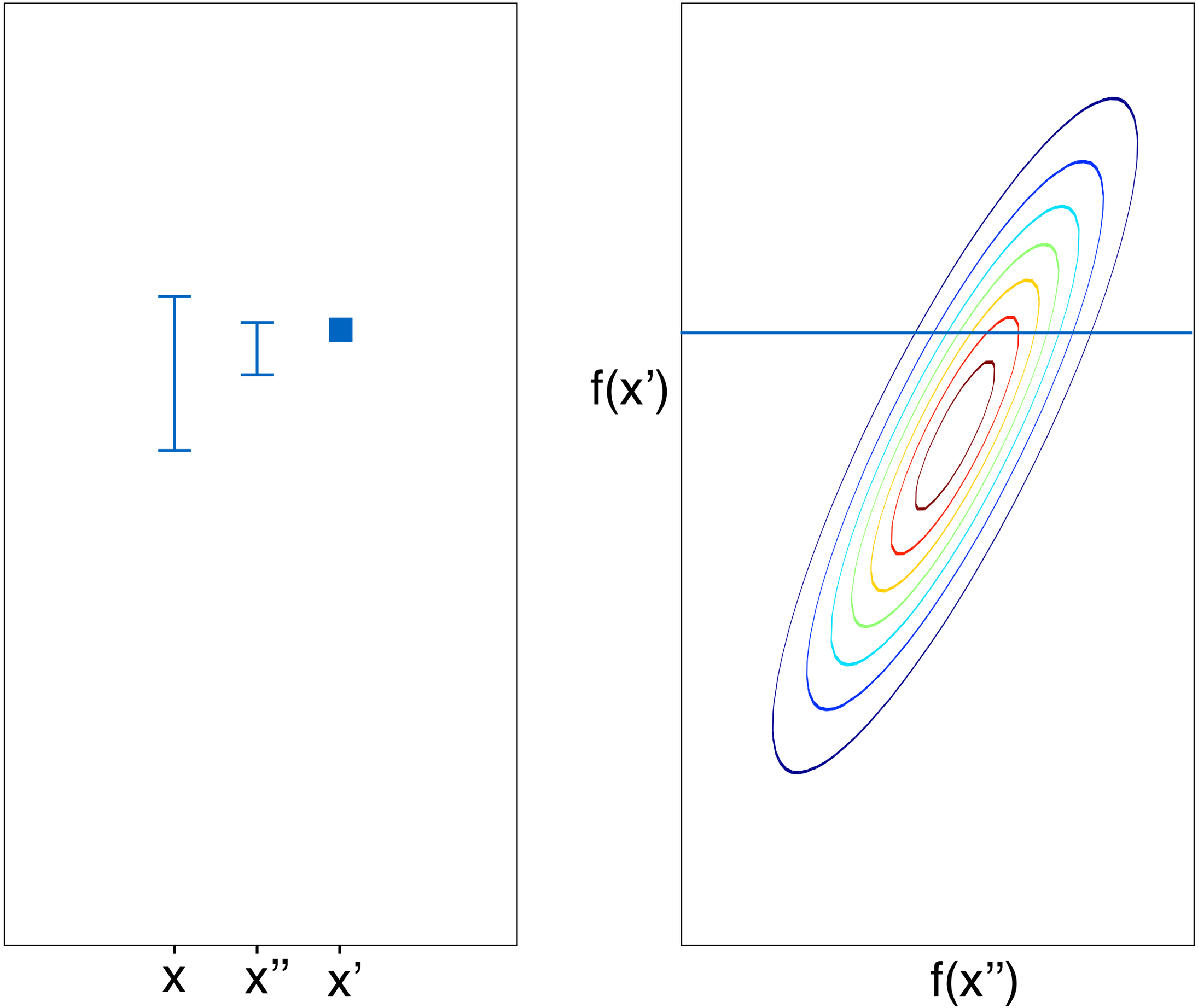

f(x')
x
x'
f(x)


x
x’


f(x’)
f(x)


x
x’


f(x’)
f(x)


f(x')
x
x''
x'
f(x'')

Gaussian Process Regression

- A prior on a function f is a **Gaussian process prior** if, for any collection of points $x_1, \ldots, x_k$,

$$\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_k) \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_0(x_1) \\ \vdots \\ \mu_0(x_k) \end{bmatrix}, \begin{bmatrix} \Sigma_0(x_1, x_1) & \ldots & \Sigma_0(x_1, x_k) \\ \vdots & \ddots & \vdots \\ \Sigma_0(x_k, x_1) & \ldots & \Sigma_0(x_k, x_k) \end{bmatrix} \right)$$

- we call Σ_0 the "kernel" or "covariance function"
- we call μ_0 the "mean function"

We can compute the posterior analytically

- The posterior on f(x') given $f(x_1)$, …, $f(x_n)$ is normal with mean $\mu_n(x')$ and variance $\sigma_n^2(x')$:

- $\mu_n(x') = \Sigma_0(x', x_{1:n})\ \Sigma_0(x_{1:n}, x_{1:n})^{-1}\ f(x_{1:n})$

- $\sigma_n^2(x') = \Sigma_0(x', x') - \Sigma_0(x', x_{1:n}) \Sigma_0(x_{1:n}, x_{1:n})^{-1} \Sigma_0(x_{1:n}, x')$

- Formula assumes $\mu_0 = 0$.
  There is a similar formula for general μ_0.

How should we choose the kernel & mean functions?

- 1. Designate a parametric family.
- 2. Take an initial stage of samples from points sampled uniformly at random
- 3. Choose hyperparameters using either:
  - 2a. Maximum likelihood estimation
  - 2b. Maximum *a posteriori* estimation, with a prior on the hyperparameters
  - 2c. Sample the hyperparameters from their posterior

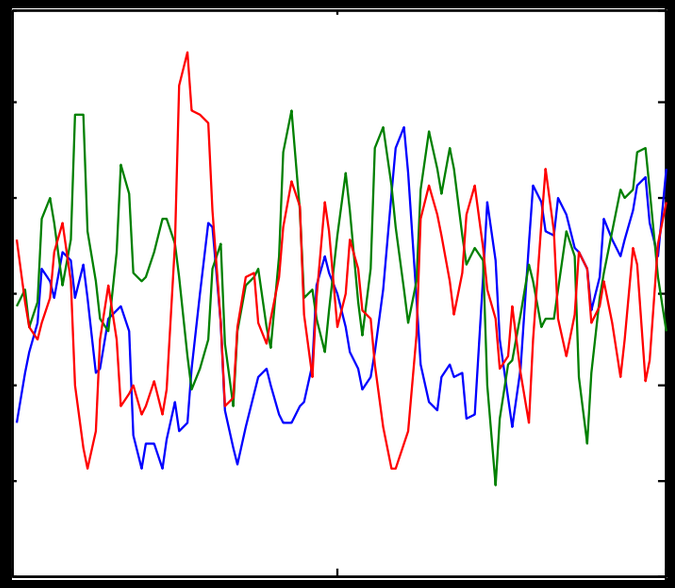
How should we choose the mean function?

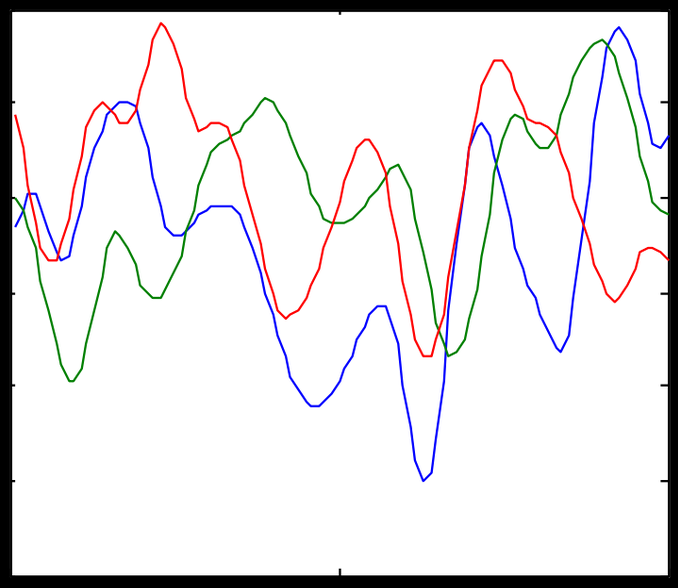
- Parametric Family:
  - $\mu_0(x) = \beta$
  - $\mu_0(x) = \beta_0 + \Sigma_j \beta_j \Psi_j(x)$
- Prior:
  - Flat over R (non-informative)
  - Uniform
  - Normal

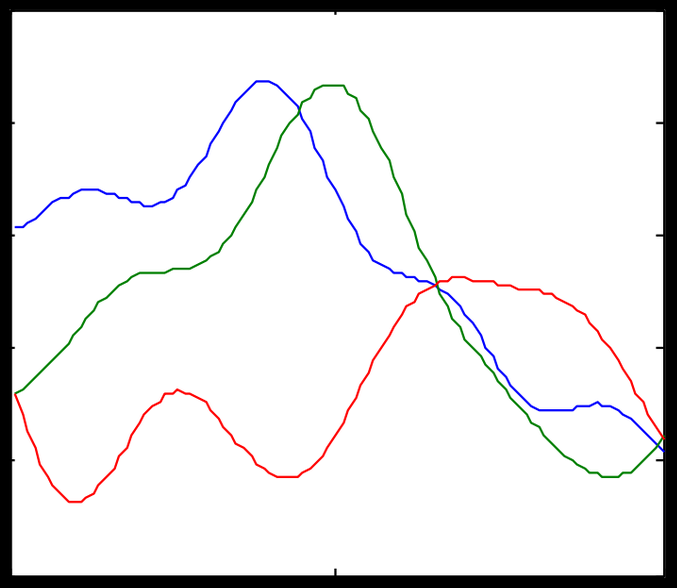
How should we choose the kernel?

- Parametric Family:
  - $\Sigma_0(x,x') = \alpha_0 \exp(-\|x-x'\|^p)$
  - $\Sigma_0(x,x') = \alpha_0 \, 2^{1-\nu} z^\nu K_\nu(z) / \Gamma(\nu)$
    where $z=(2\nu)^{1/2}\|x-x'\|/\rho$,
    $K_\nu(.)$ is the modified Bessel function
  - The above with x_d rescaled by α_d
- Prior: flat, uniform, or normal

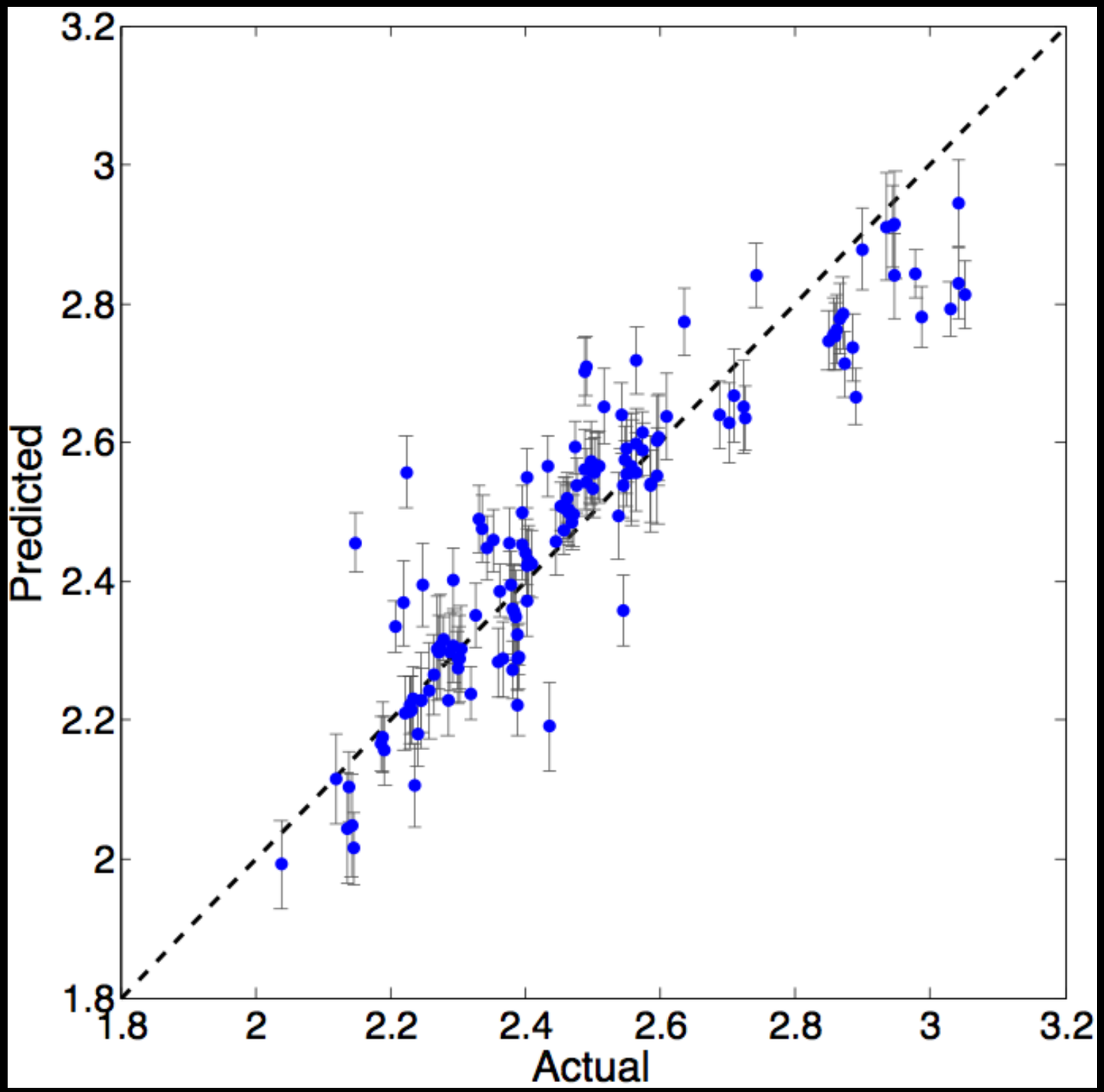
How should we choose the kernel?







Leave one-out cross-validation is worth doing



Noise can be incorporated

$$\mu_n(x') = \Sigma_0(x', x_{1:n}) \, [\Sigma_0(x_{1:n}, x_{1:n}) + \lambda^2 I]^{-1} \, f(x_{1:n})$$

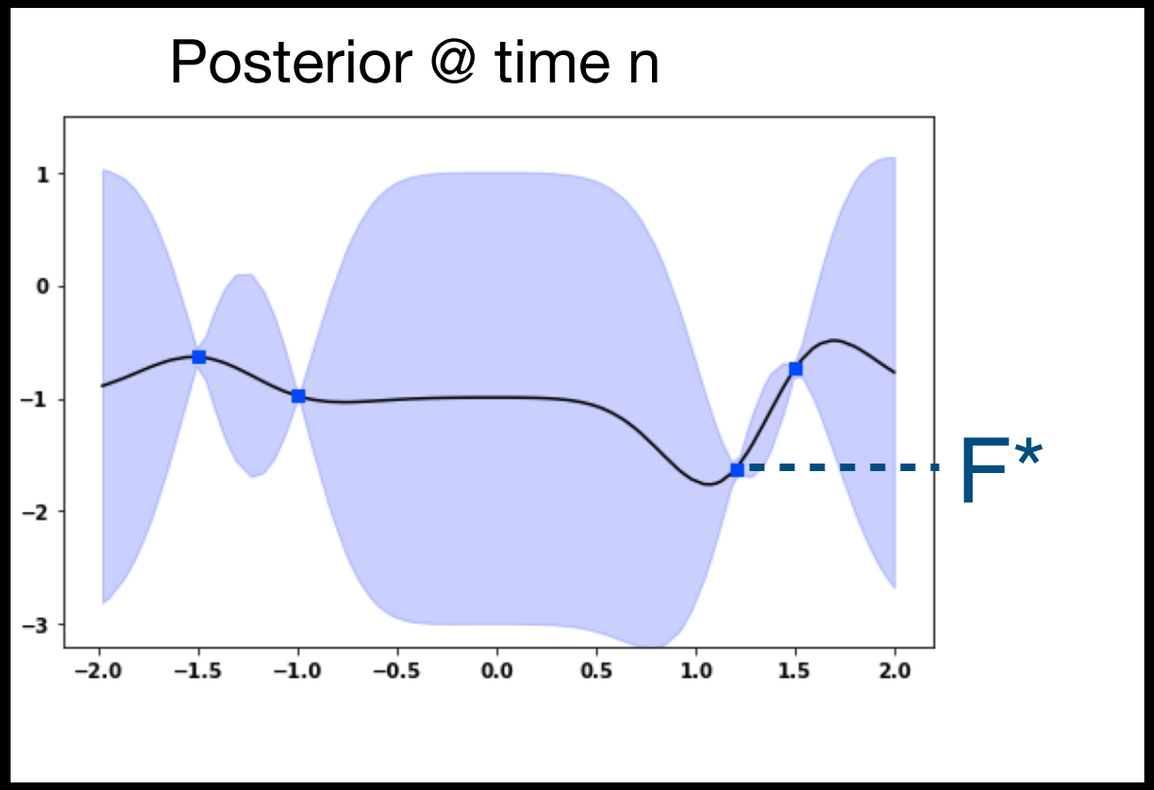
$$\sigma_n^2(x') = \Sigma_0(x', x') - \Sigma_0(x', x_{1:n}) [\Sigma_0(x_{1:n}, x_{1:n}) + \lambda^2 I]^{-1} \Sigma_0(x_{1:n}, x')$$

For heteroscedastic noise:
Kersting et al. "Most likely heteroscedastic gaussian process regression" ICML 2007

Agenda

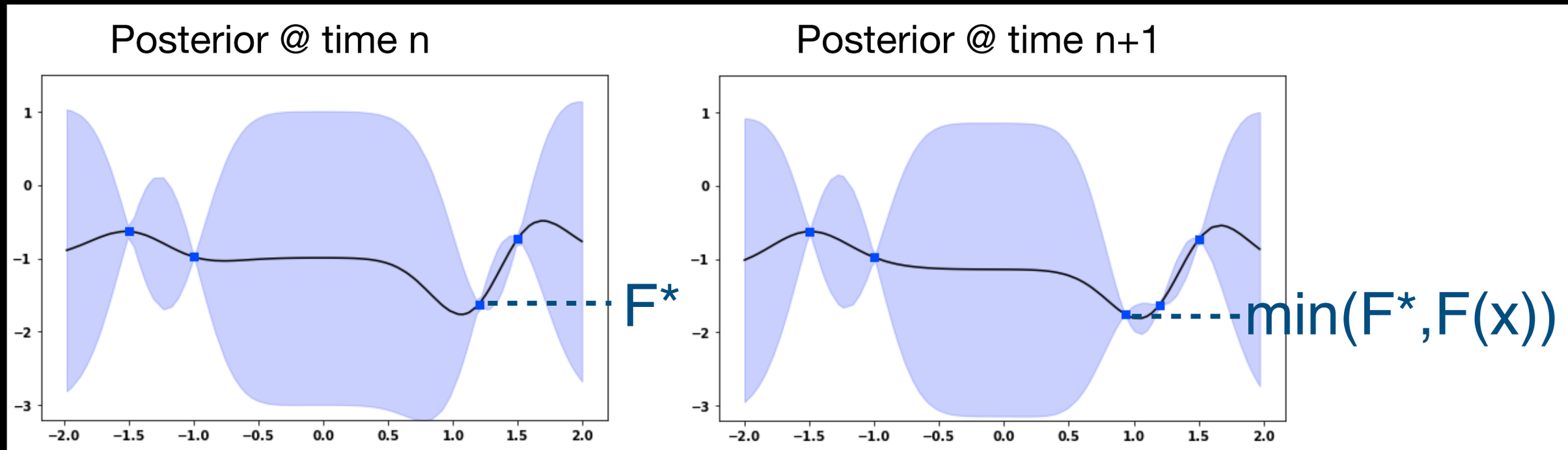
- Gaussian process regression
- **Expected improvement (EI)**
- Parallel BayesOpt (using EI)
- Parallel BayesOpt with gradients and/or noise (using KG)
- Code & Research directions

This is the Expected Improvement (EI) acquisition function [Mockus 1989; Jones, Schonlau & Welch 1998]



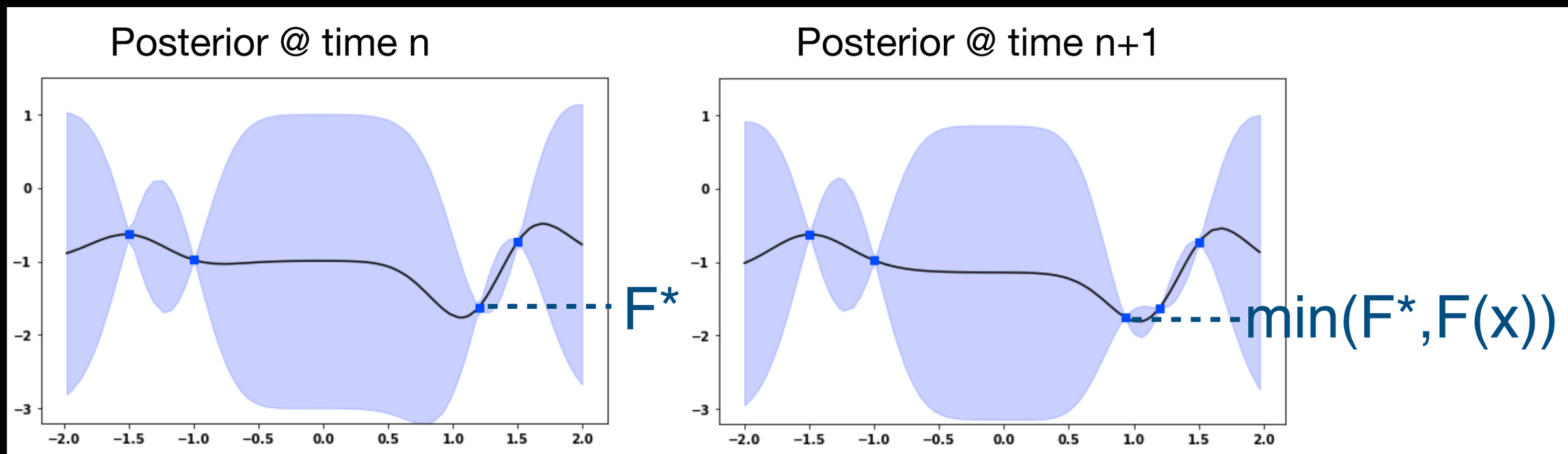
- Loss if we stop now: F*

This is the Expected Improvement (EI) acquisition function [Mockus 1989; Jones, Schonlau & Welch 1998]



- Loss if we stop now:
  F*
- Loss if we stop after sampling F(x):
  min(F*,F(x))

This is the Expected Improvement (EI) acquisition function [Mockus 1989; Jones, Schonlau & Welch 1998]



- Loss if we stop now:
  F*
- Loss if we stop after sampling F(x):
  min(F*,F(x))
- Reduction in loss due to sampling:
  $E_n[F^* - \min(F^*, F(x))] = E_n[(F^* - F(x))^+] = EI(x)$

Expected improvement is Bayes-optimal (in the noise-free standard BO problem) under some assumptions:

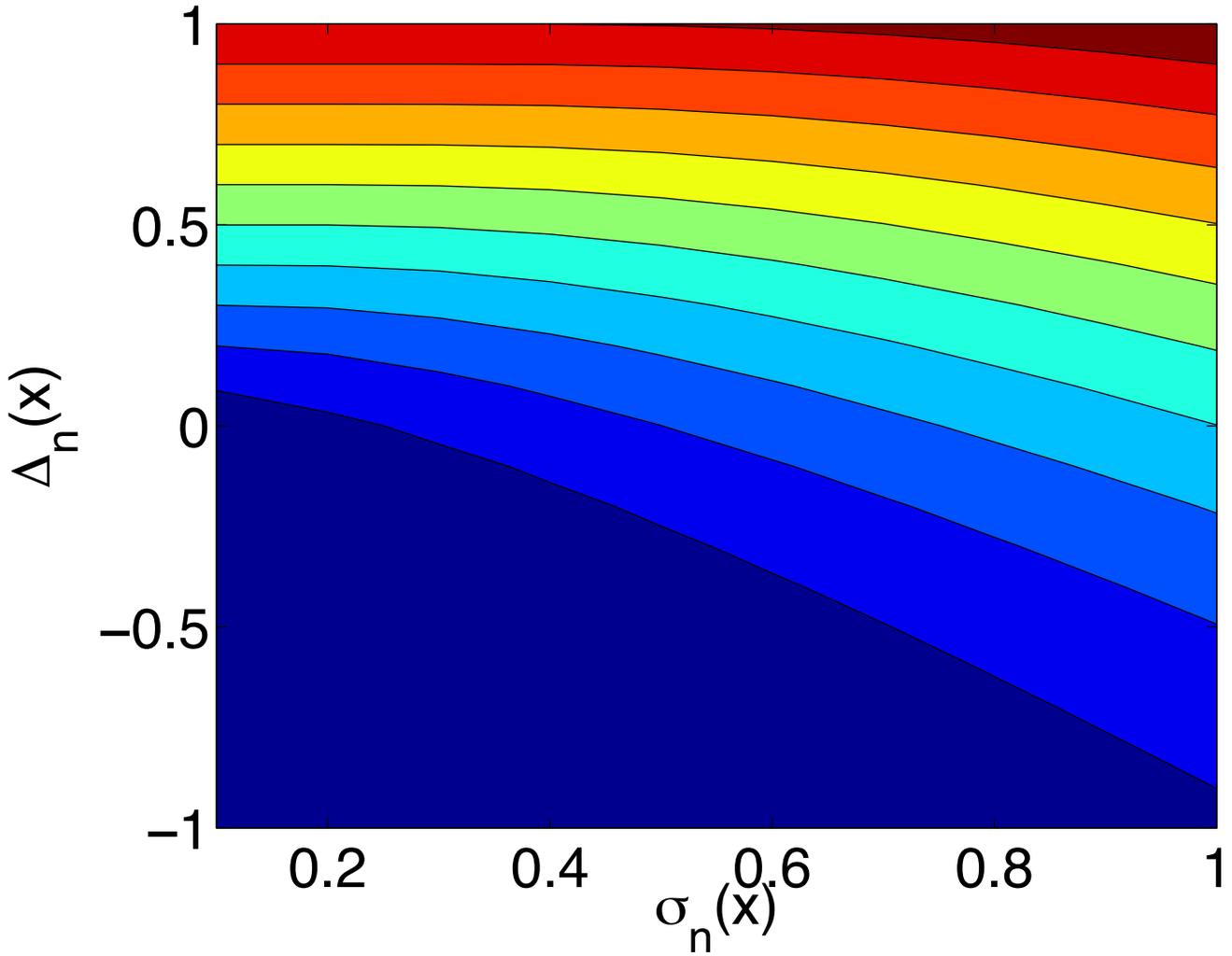
- this is our last evaluation
- we are risk neutral
- we are only willing to select a previously evaluated point as a final solution

You can compute expected improvement in closed form

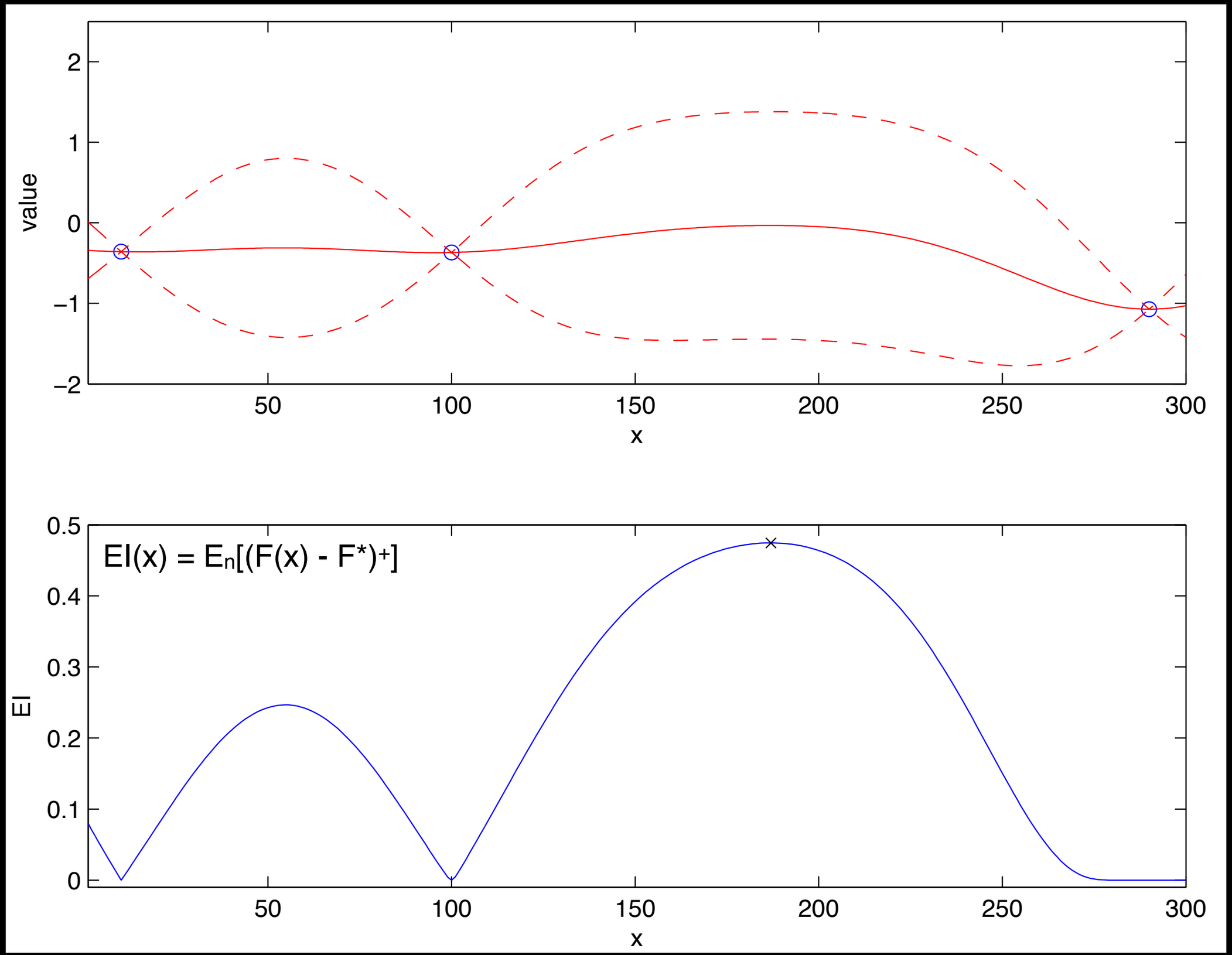
$$\mathrm{EI}_n(x) = [\Delta_n(x)]^+ + \sigma_n(x)\varphi\left(\frac{\Delta_n(x)}{\sigma_n(x)}\right) - |\Delta_n(x)|\Phi\left(-\frac{|\Delta_n(x)|}{\sigma_n(x)}\right)$$

where $\Delta_n(x) = F_n^* - \mu_n(x)$

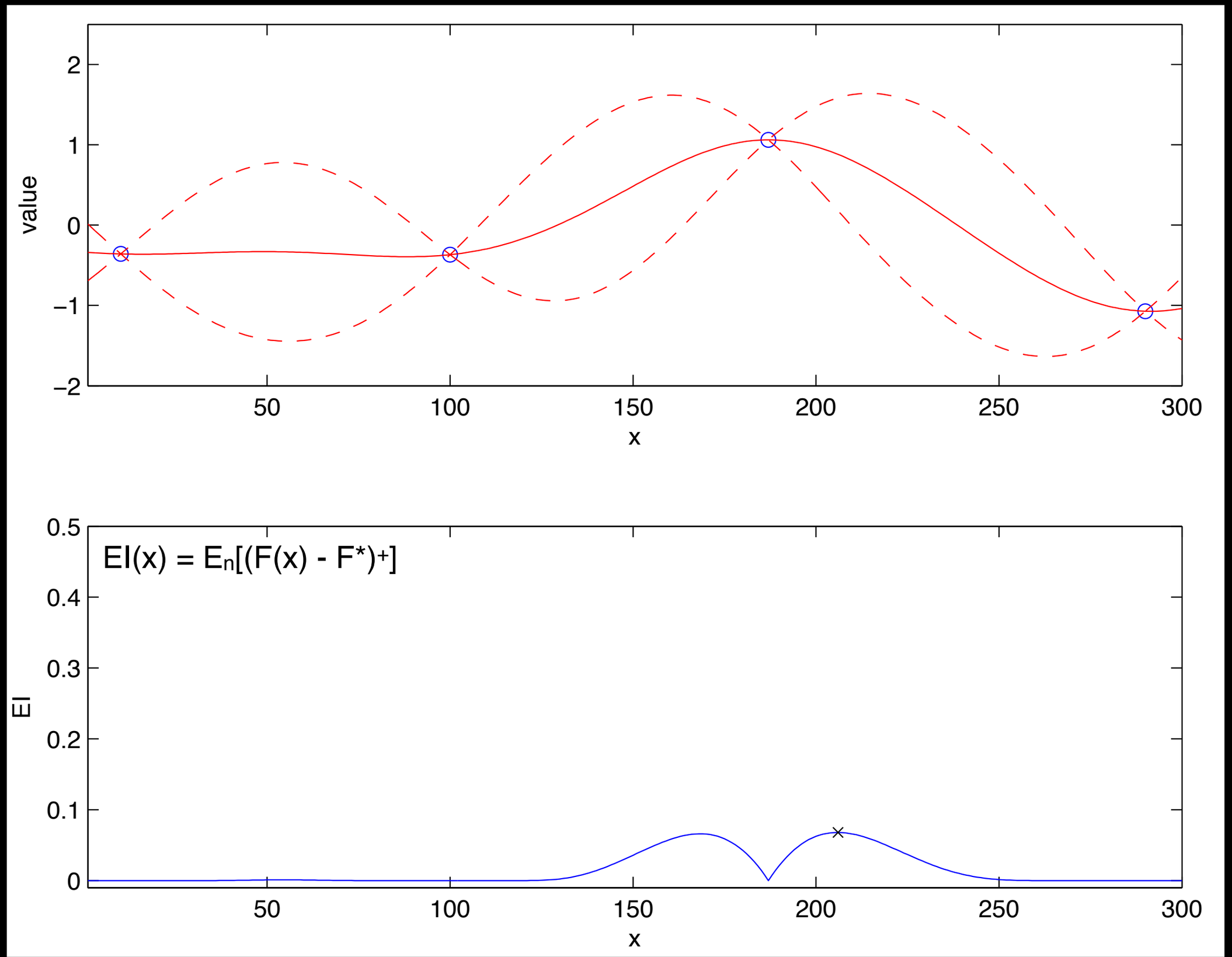
Expected improvement trades exploration ($\sigma_n(x)$) vs. exploitation ($\Delta_n(x)$)



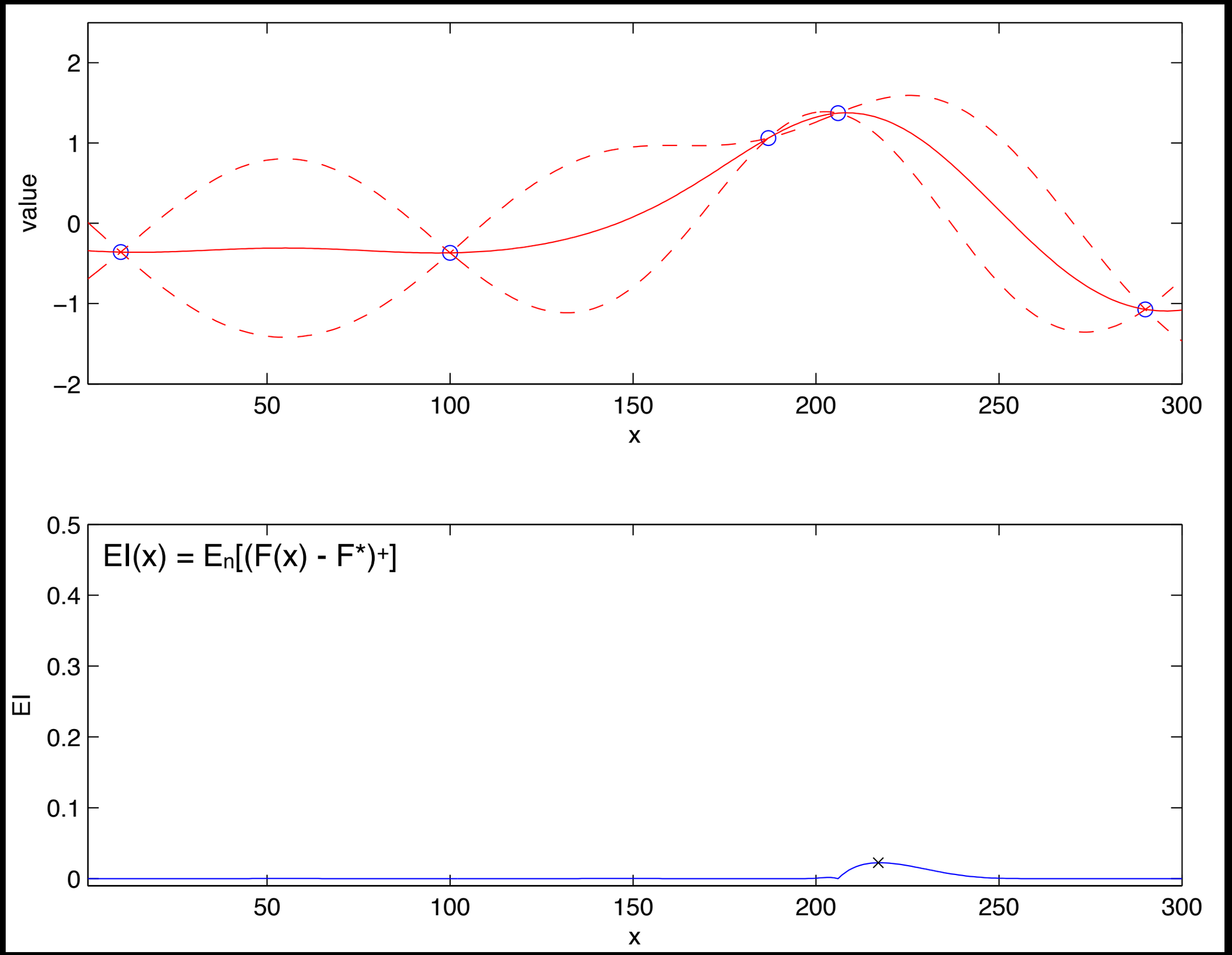
Here is expected improvement maximizing a 1-dim objective



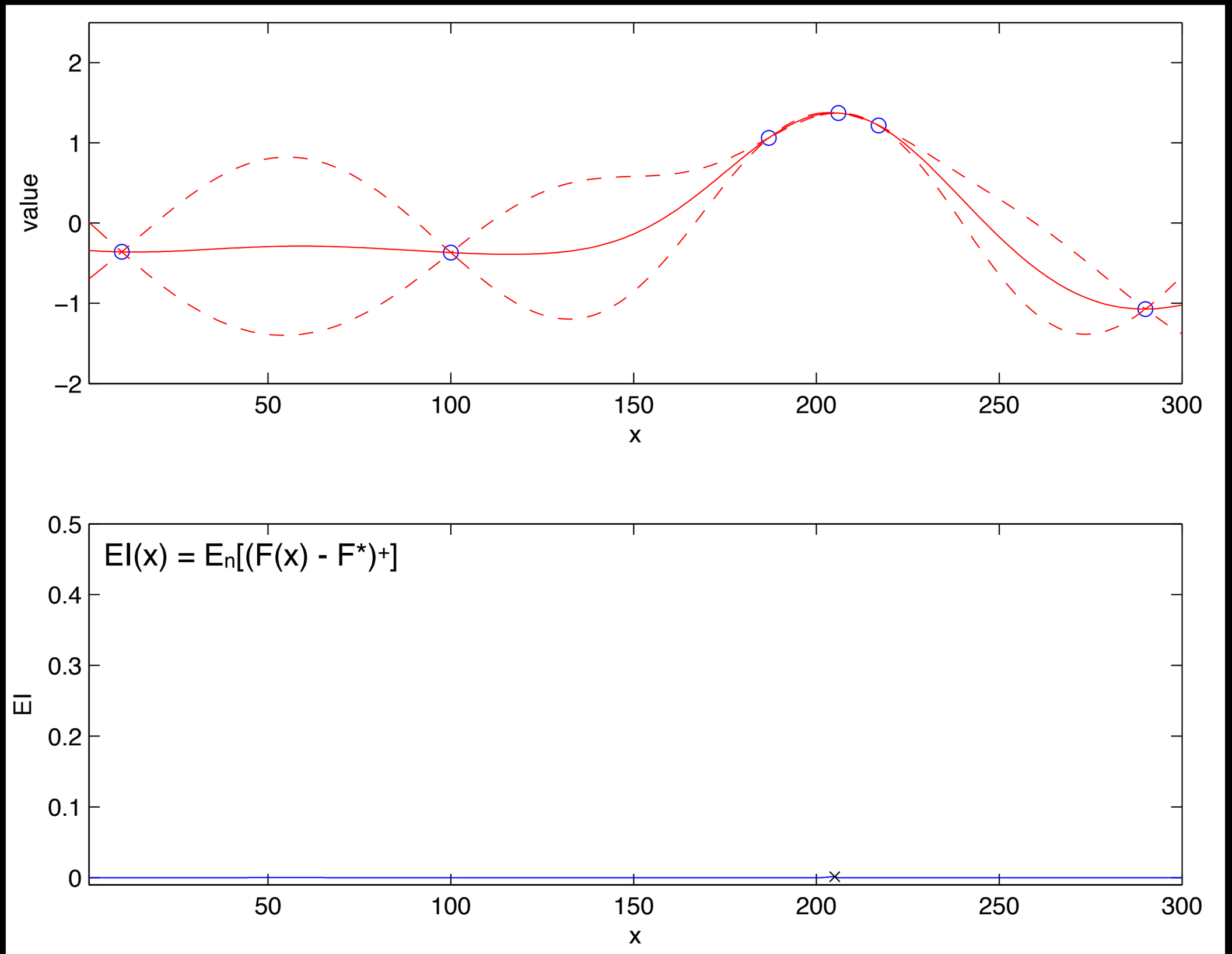
Here is expected improvement maximizing a 1-dim objective



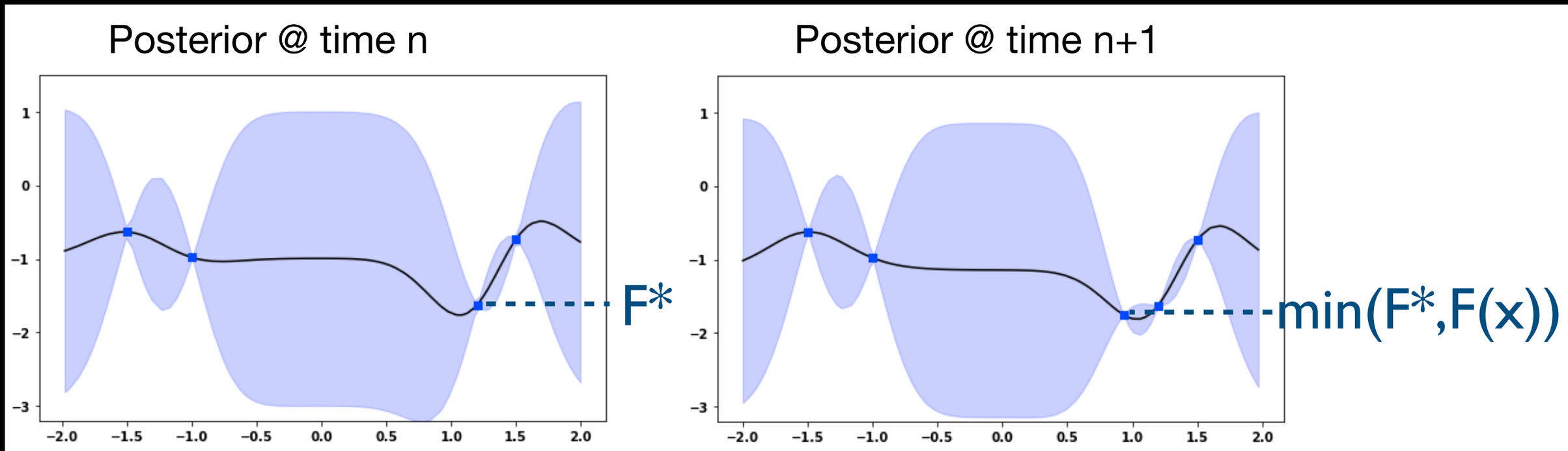
Here is expected improvement maximizing a 1-dim objective



Here is expected improvement maximizing a 1-dim objective



Expected Improvement is one-step Bases-Optimal under assumptions



Expected improvement is Bayes-Optimal
(in the noise-free standard BayesOpt problem) if:

- this is our last evaluation
- we are risk neutral
- we are only willing to select a previously evaluated point as a final solution

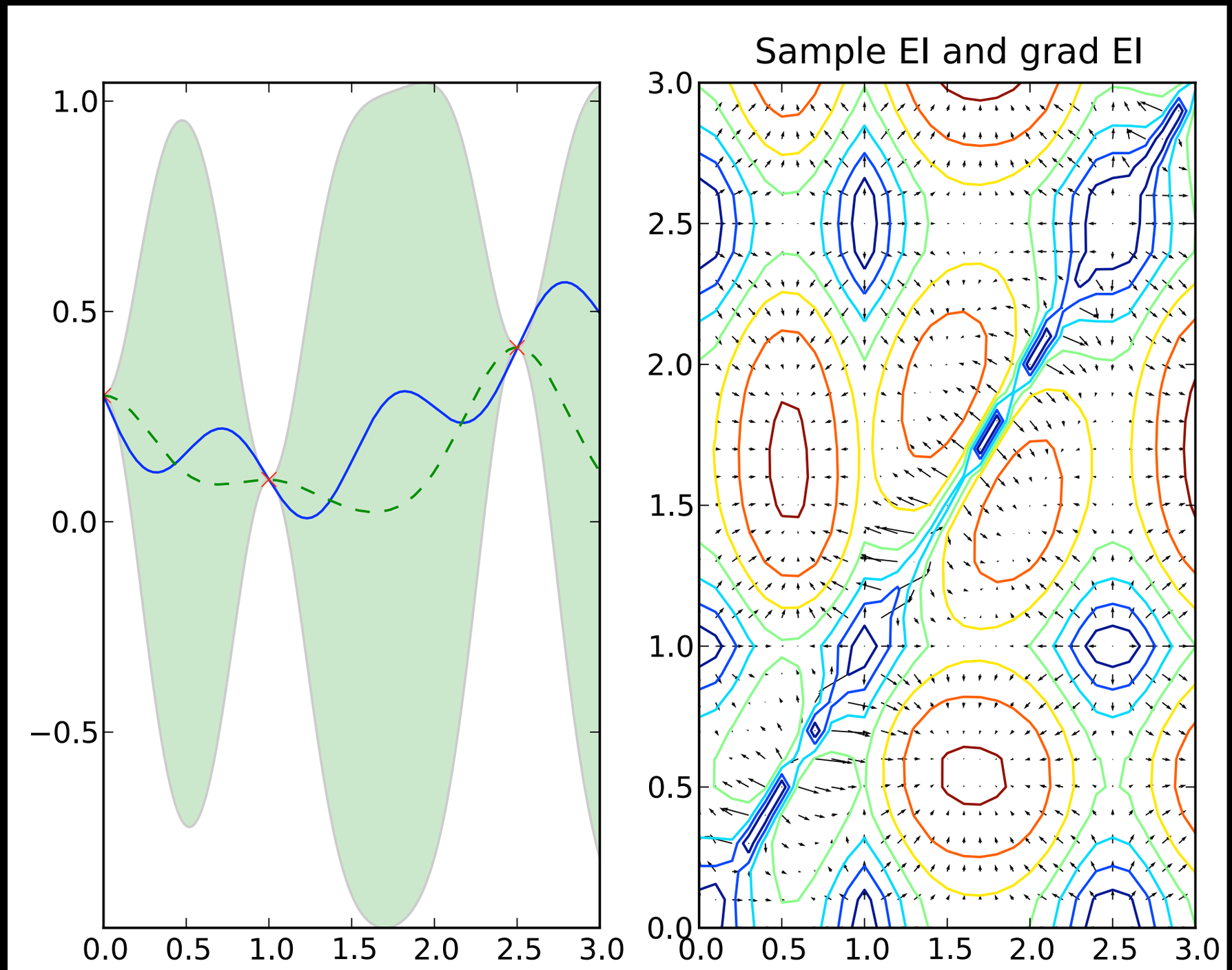
Agenda

- Gaussian process regression
- Expected improvement (EI)
- **Parallel BayesOpt (using EI)**
- Parallel BayesOpt with gradients and/or noise (using KG)
- Code & Research directions

We can parallelize EI

$$EI(x_{1:q}) = E_n[(F^* - \min\{F(x_1),\ldots,F(x_q)\})^+]$$

We can parallelize EI



$$EI(x_{1:q}) = E_n[(F^* - \min\{F(x_1),\ldots,F(x_q)\})^+]$$

Here's how to maximize parallel EI

1. Construct an unbiased estimator
   of the gradient of $\mathrm{EI}(x_{1:q})$
   using infinitesimal perturbation analysis (IPA).

2. Use multistart stochastic gradient ascent
   to approximately maximize $\mathrm{EI}(x_{1:q})$

Here's how we estimate ∇EI

- $Y=[f(x_1),\ldots,f(x_q)]'$ is multivariate normal
- Let m = E[Y] and C = Chol(Cov[Y])
- $Y=m+CZ$, where Z is a vector of independent standard normals
- $EI(x_{1:q}) = E[h(Y)]$ where $h(Y)= (F^*- \min\{Y\})^+]$
- If our problem is well-behaved,
  we can switch derivative and expectation:
  $\nabla EI(x_{1:q}) = E[\nabla h(m+CZ)]$

Here's how we estimate ∇EI

1. Simulate a vector Z of independent standard normals

2. Calculate m = E[Y], C = Chol(Cov[Y])

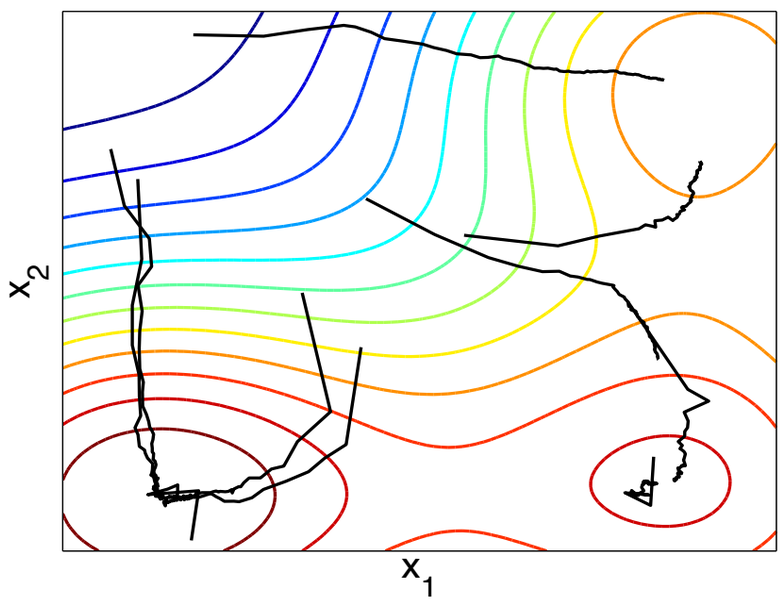
3. Our estimator of $\nabla EI(x_1,...,x_q)$ is $\nabla h(m+CZ)$
   where $h(Y)= (F^*- \min\{Y\})^+$

We use this estimator of ∇EI in multistart stochastic gradient ascent

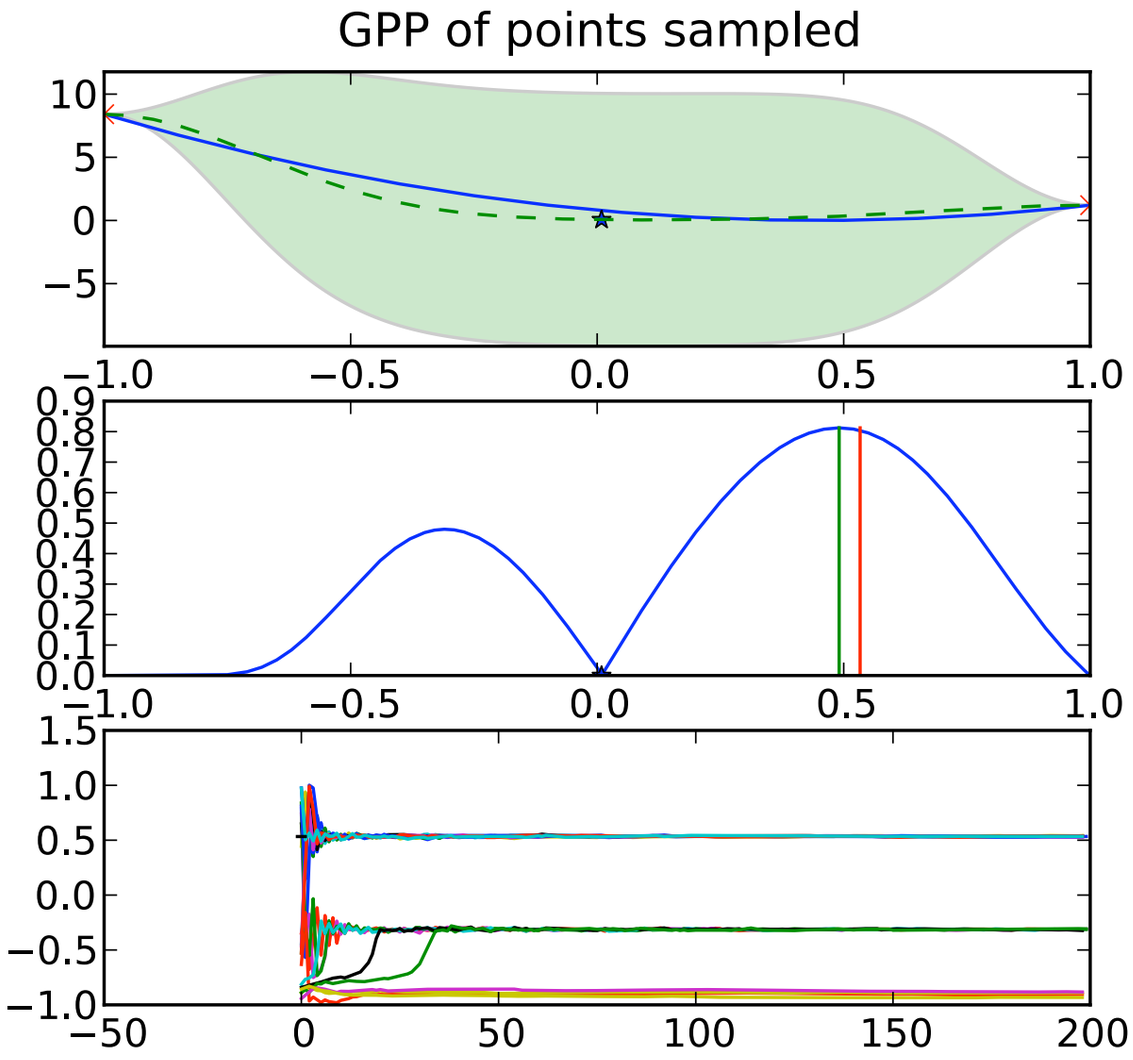
1. Select several starting points, uniformly at random.
2. From each starting point, iterate using the stochastic gradient method until convergence.

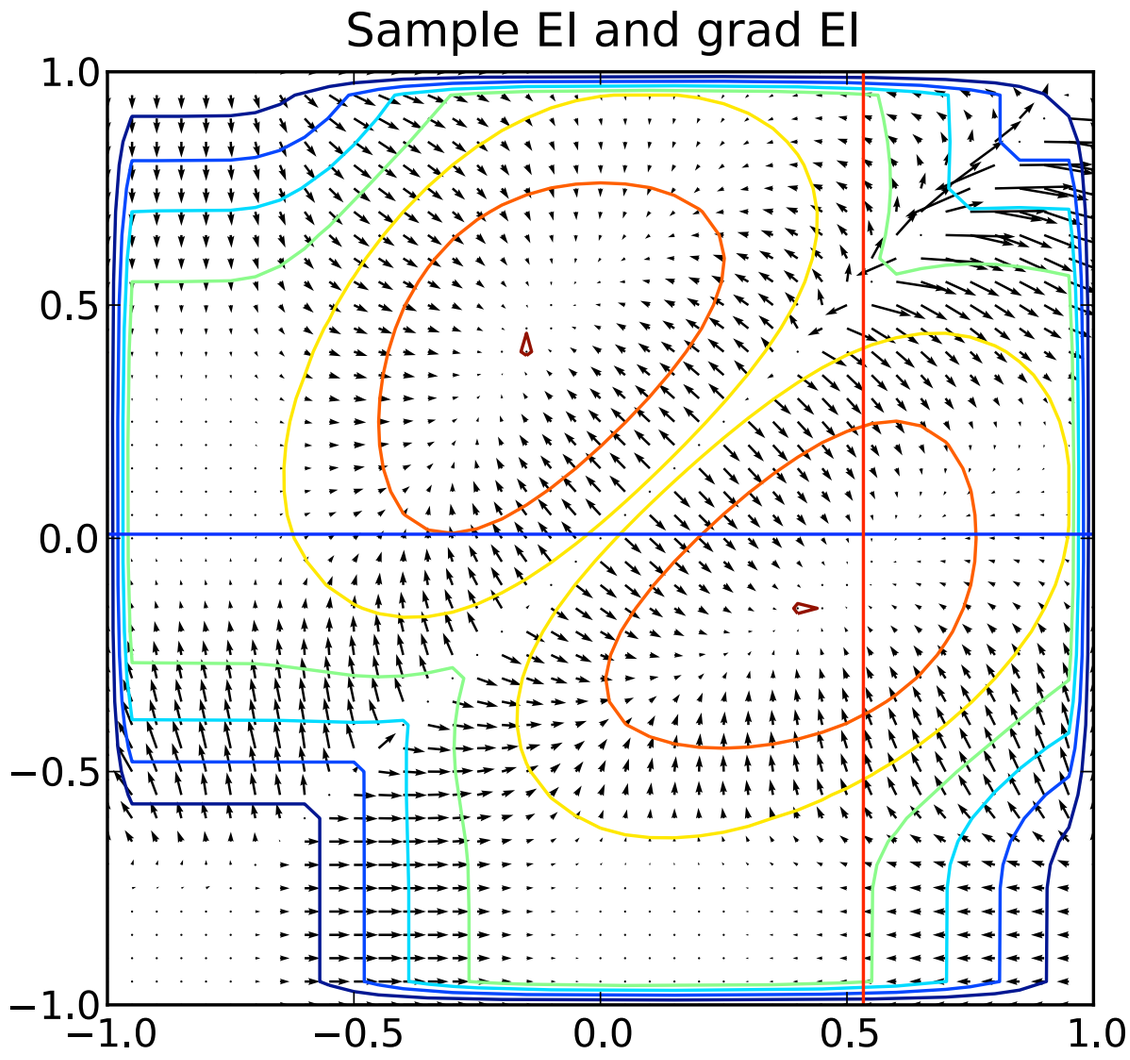
$$(\vec{x}_1, \ldots, \vec{x}_q) \leftarrow (\vec{x}_1, \ldots, \vec{x}_q) + \alpha_n g(\vec{x}_1, \ldots, \vec{x}_q, \omega),$$

   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.

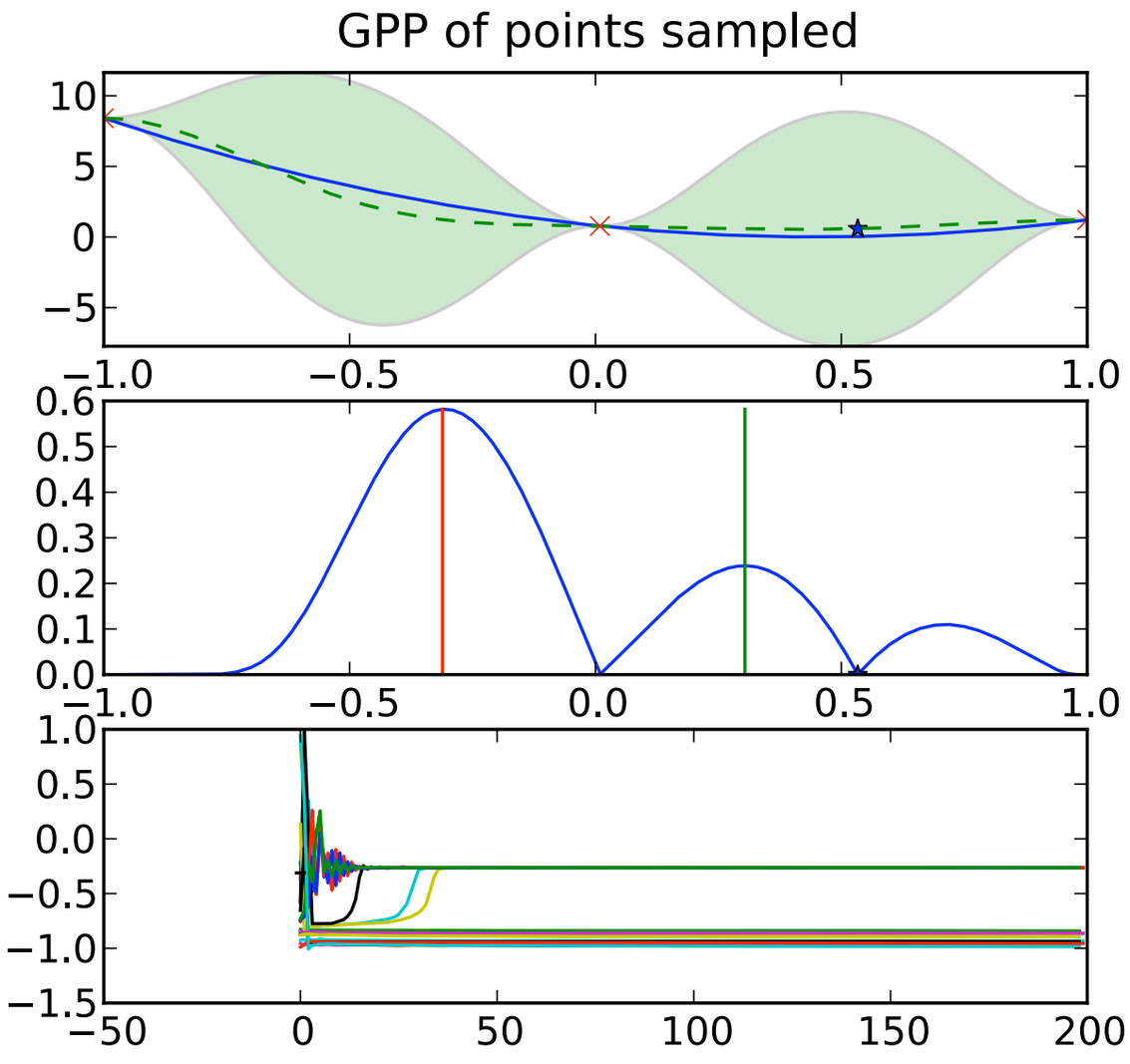


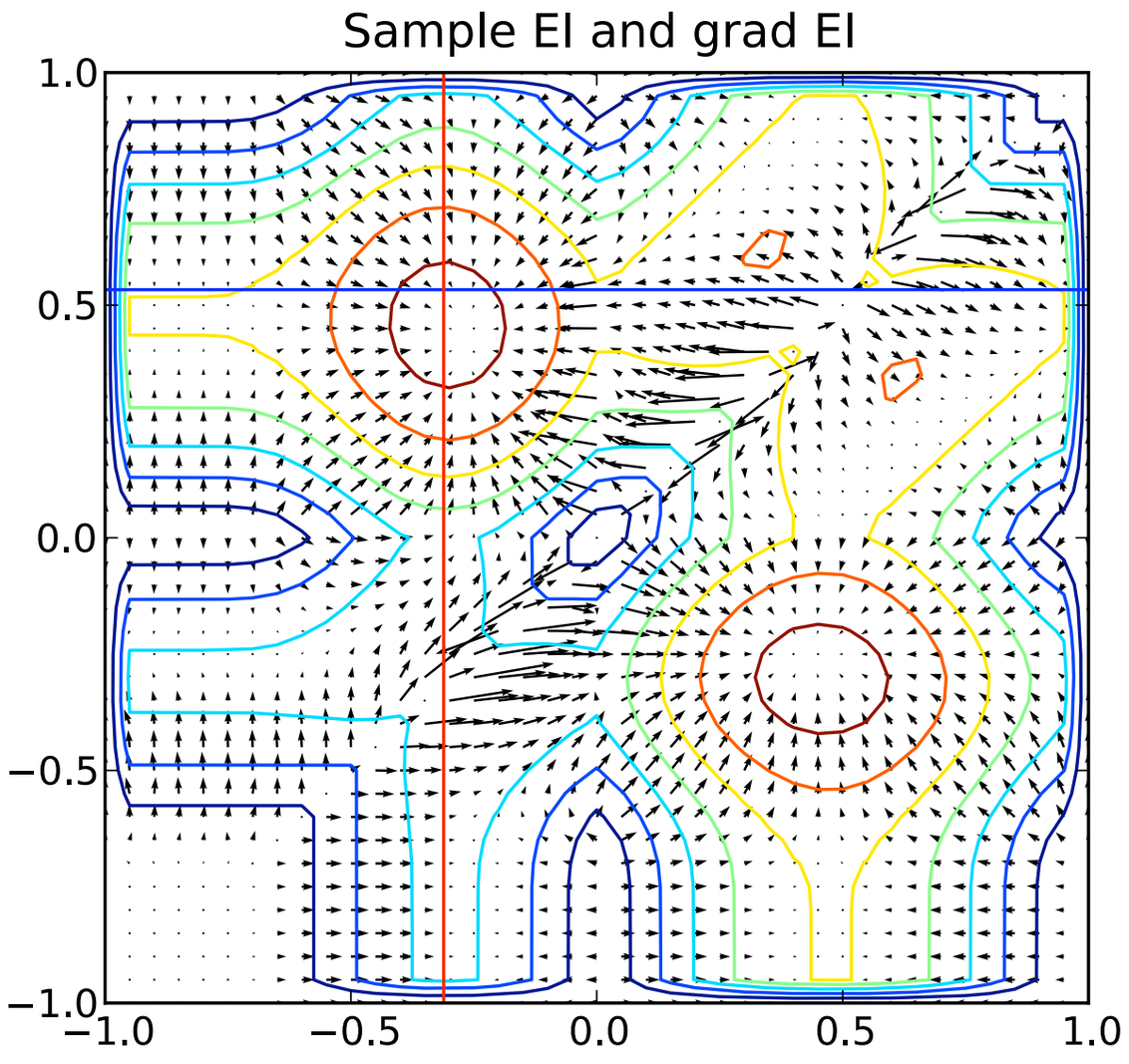
Animation


GPP of points sampled
10
5
0
−5
−1.0
−0.5
0.0
0.5
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−50
0
50
100
150
200

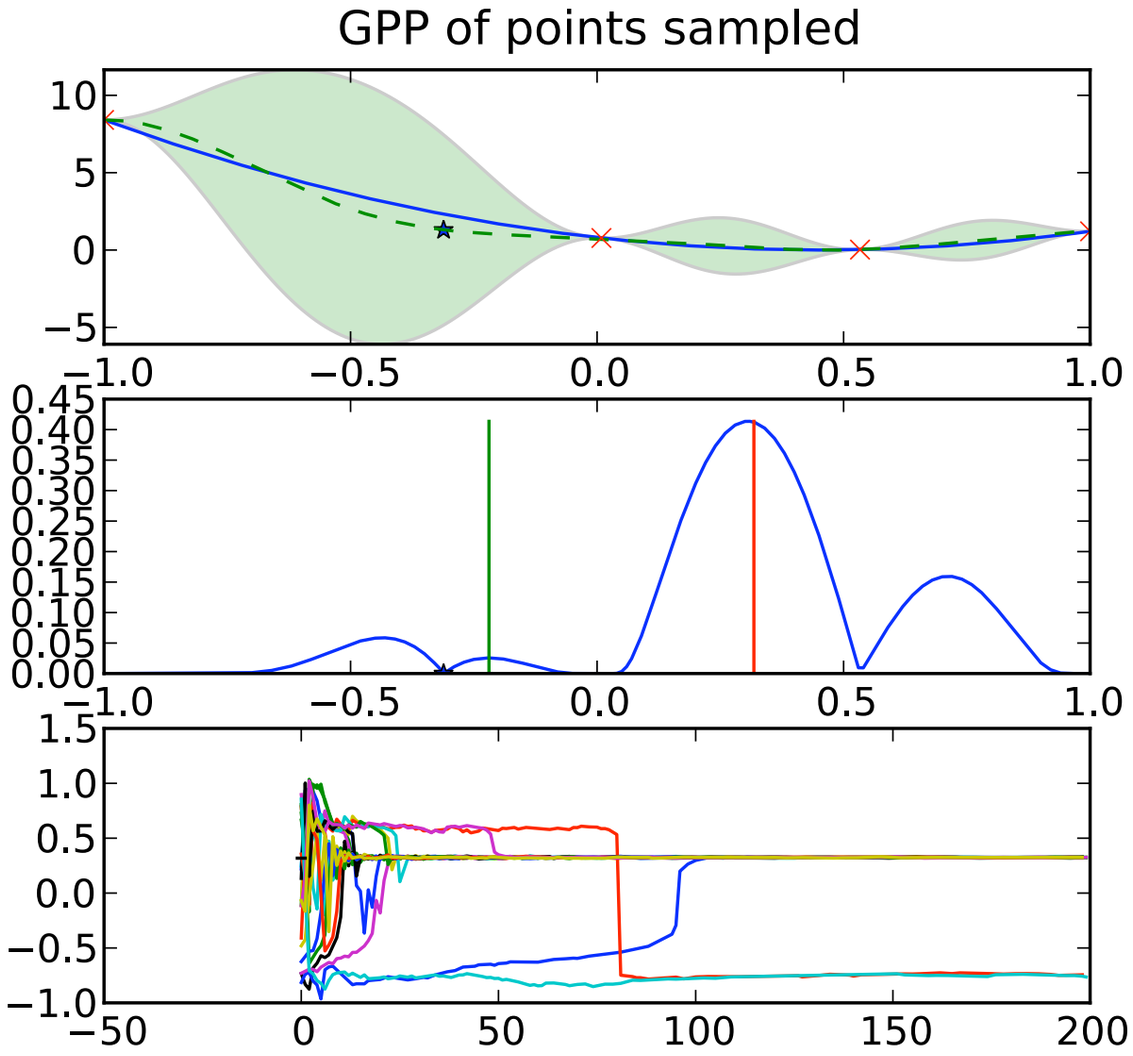

Sample EI and grad EI
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0

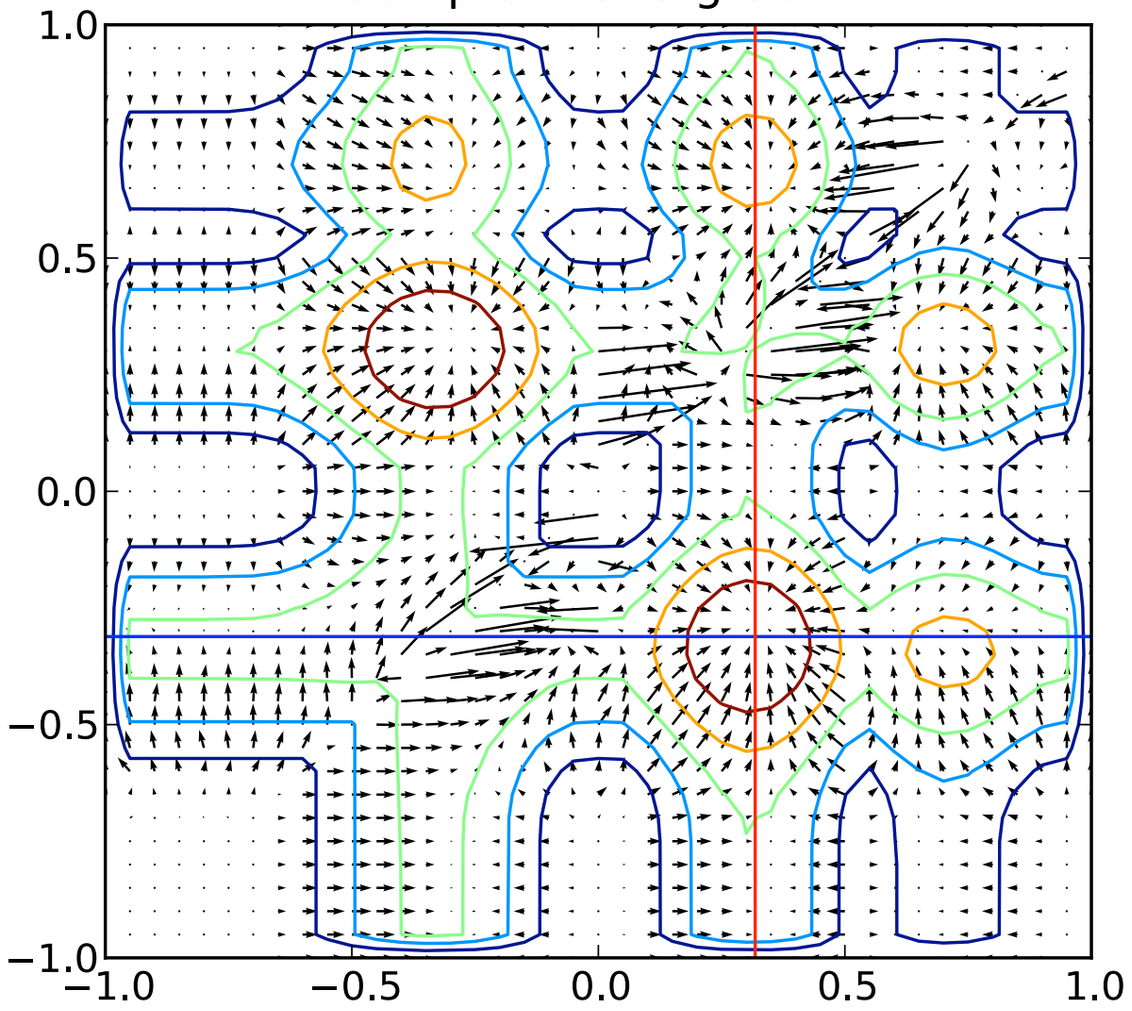
Animation


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

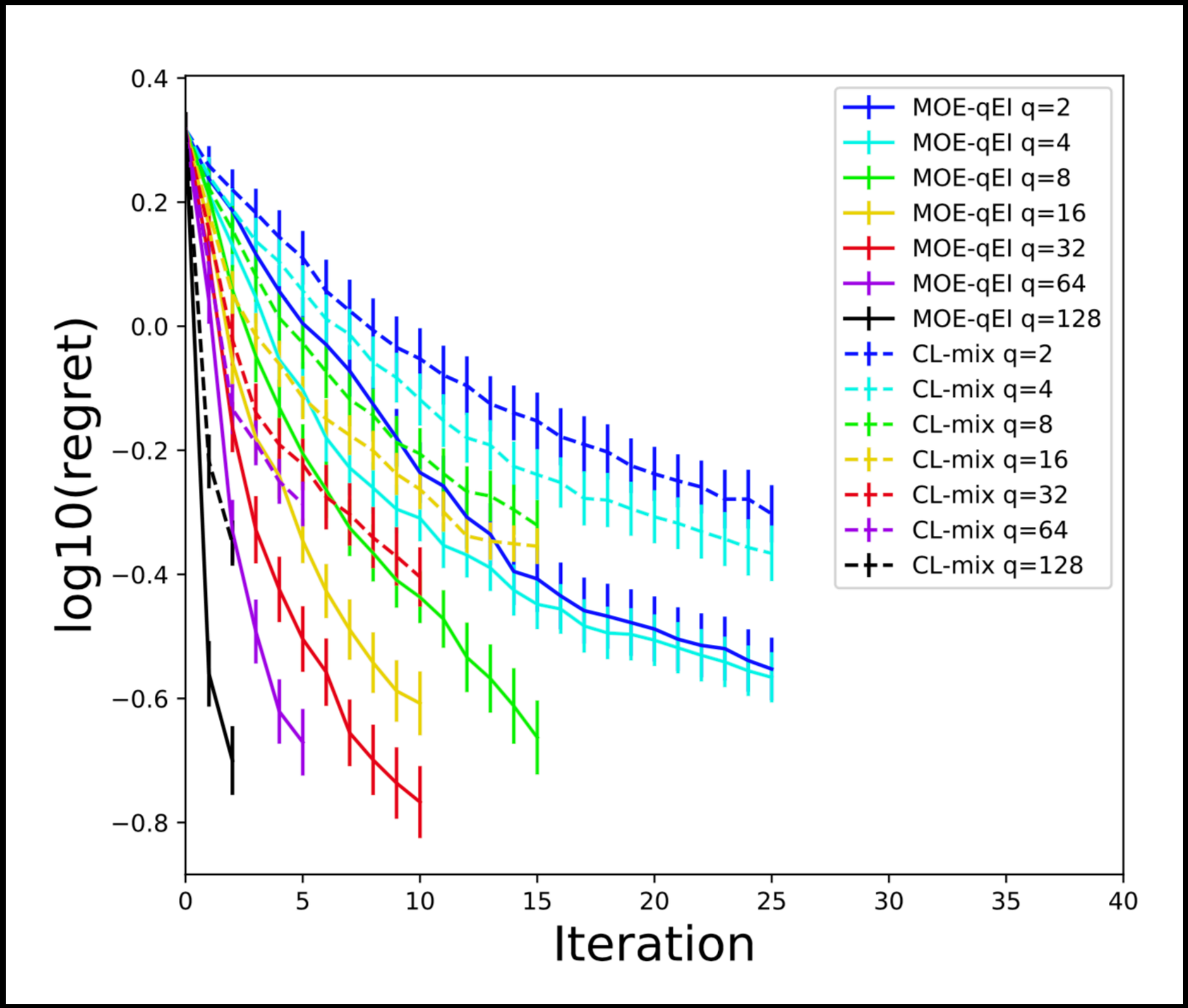

Sample EI and grad EI
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0

Animation


GPP of points sampled
10
5
0
−5
−1.0
−0.5
0.0
0.5
1.0
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
1.5
1.0
0.5
0.0
−0.5
−1.0
−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

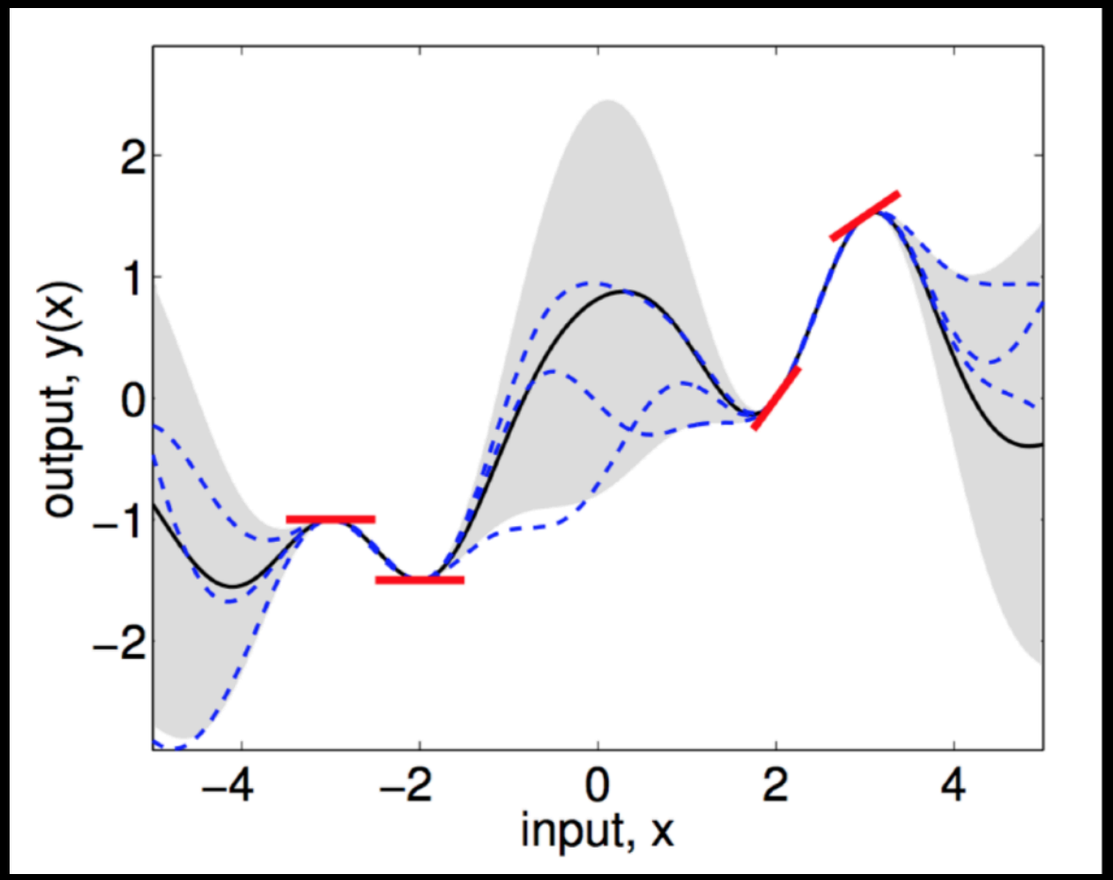
Here's a demonstration on a 6-dimensional BayesOpt problem with up to 128 parallel evaluations



Agenda

- Gaussian process regression
- Expected improvement (EI)
- Parallel BayesOpt (using EI)
- **Parallel BayesOpt with gradients/noise (using KG)**
- Code & Research directions

Bayesian (global) optimization with gradients



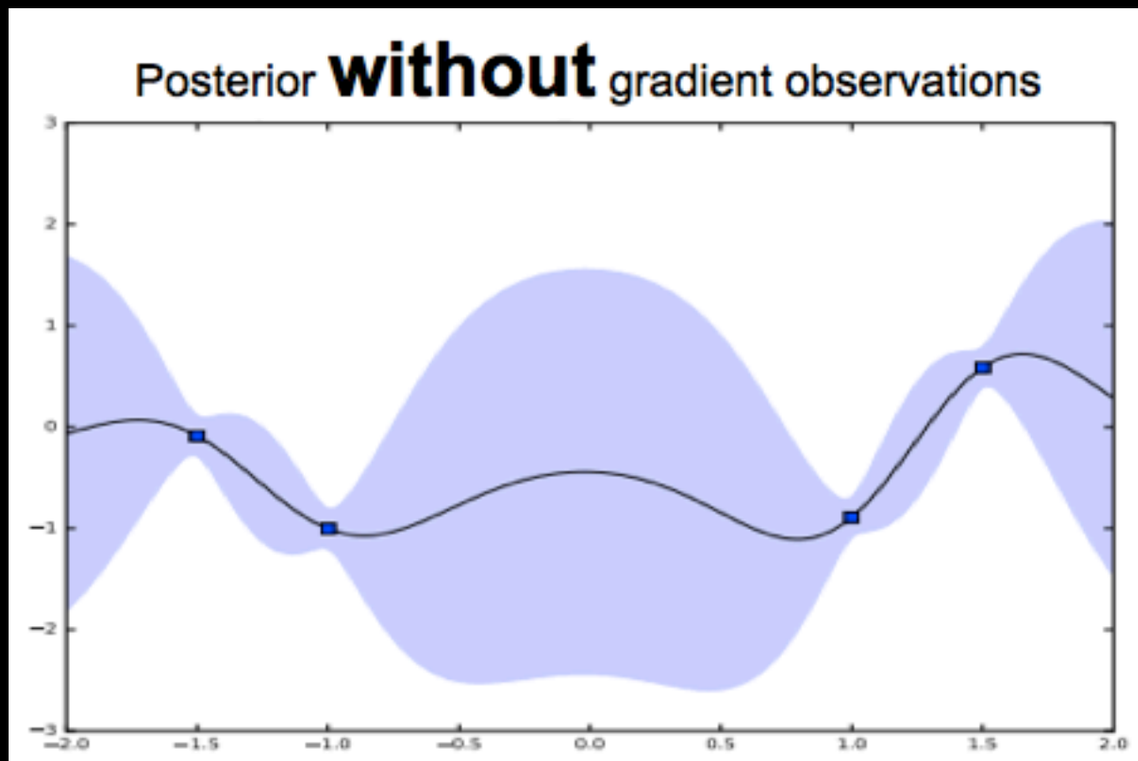
Evaluating at x provides

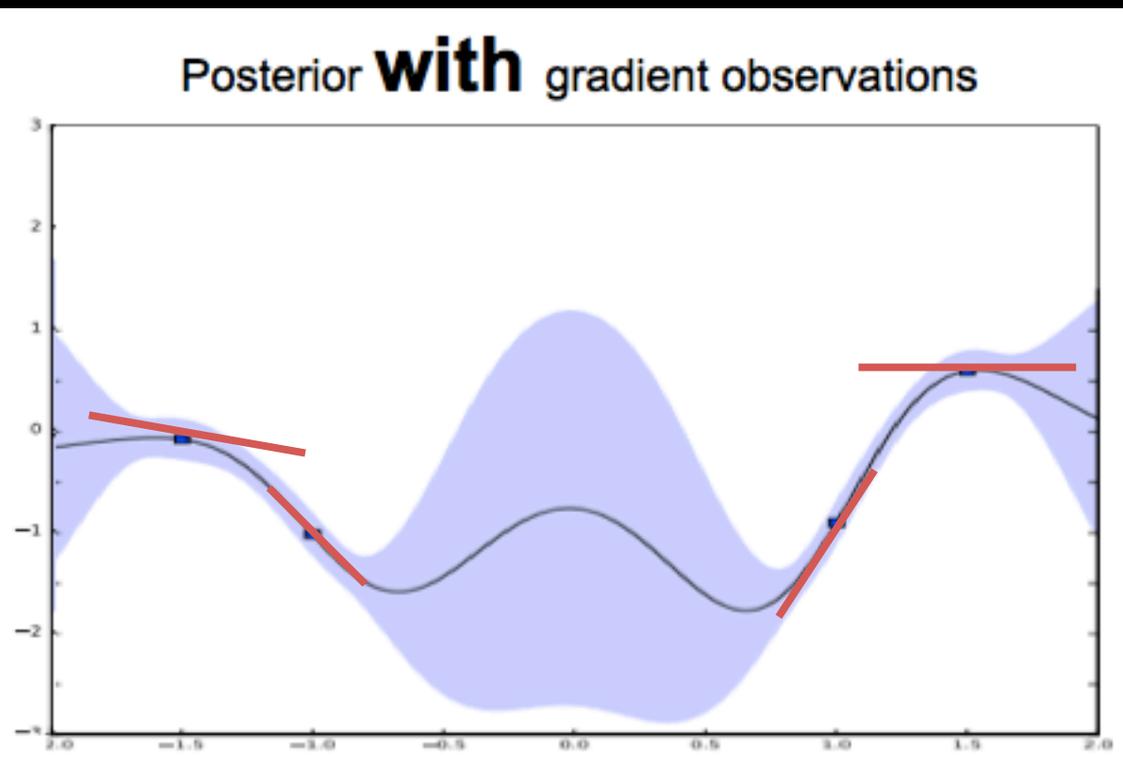
- F(x), optionally with noise
- optionally incomplete/ noisy/ biased observations of $\nabla F(x)$

Evaluations can be sequential or in batches

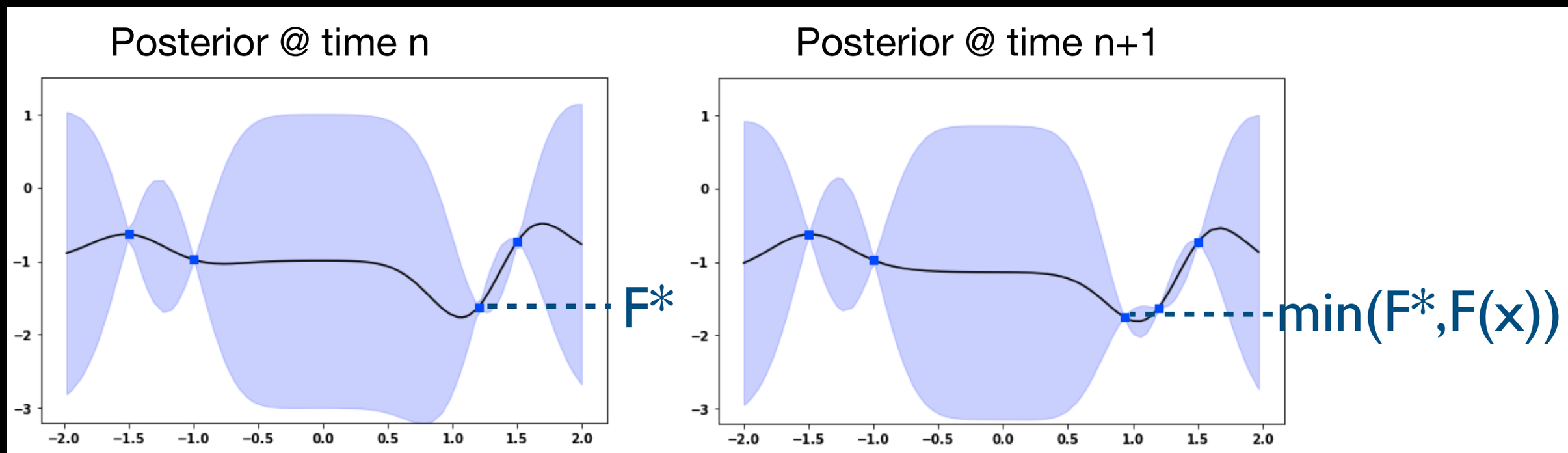
[Wu, Poloczek, Wilson, & F., NIPS 2017]

GP regression handles gradients naturally [Rasmussen & Williams 2006]



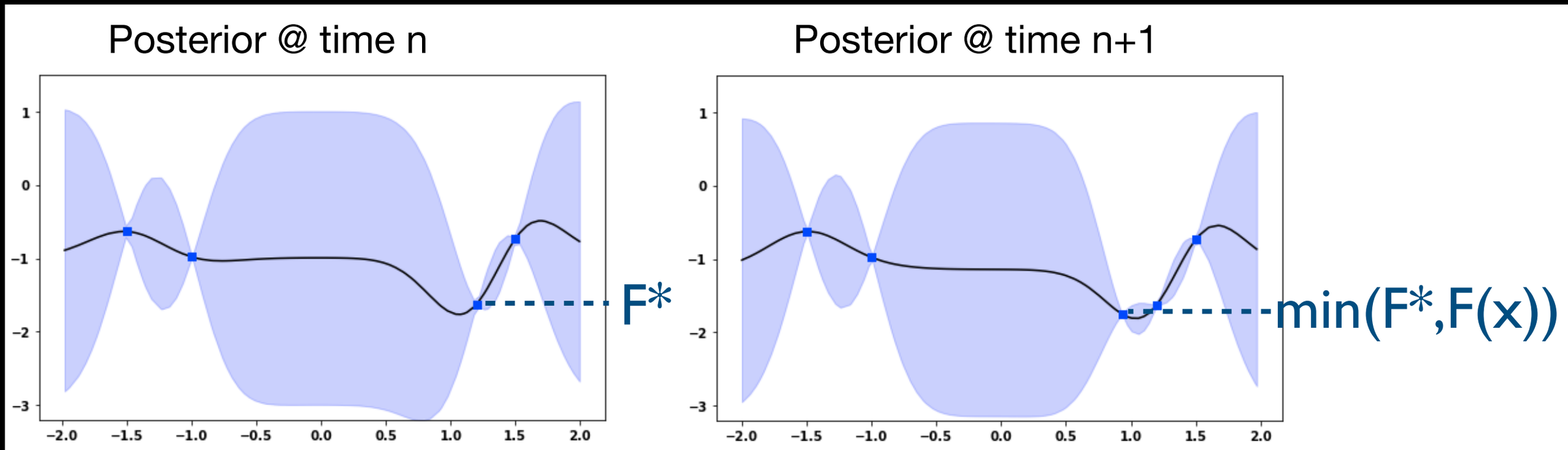


Here is EI in a standard BO problem

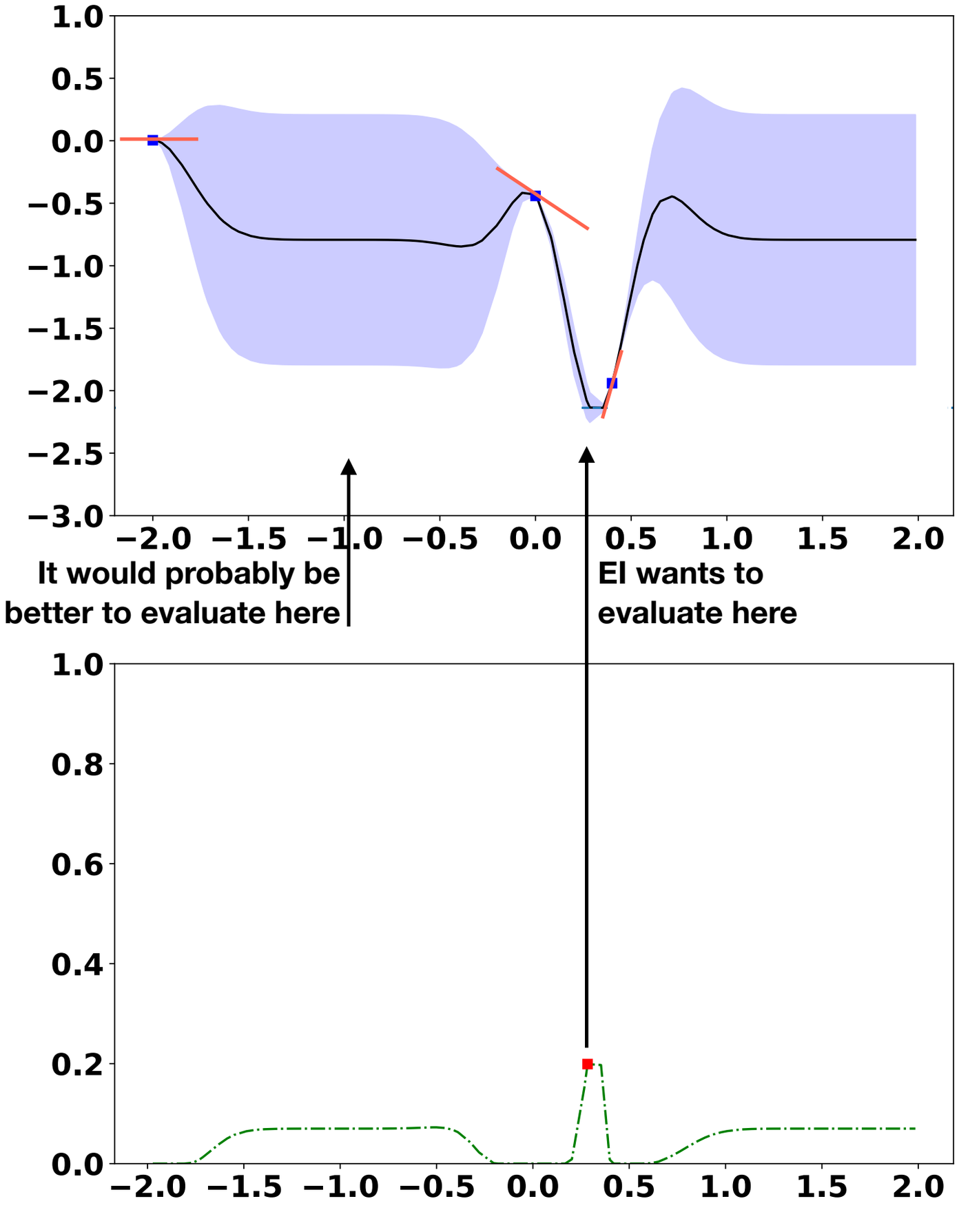


- Loss if we stop now:
  F^*
- Loss if we stop after sampling F(x):
  $\min(F^*,F(x))$
- Reduction in loss due to sampling:
  $E_n[F^* - \min(F^*,F(x))] = E_n[(F^* - F(x))^+] = EI(x)$

EI places no value on gradient information



- Loss if we stop now:
  F*
- Loss if we stop after sampling F(x) & its gradient:
  min(F*,F(x))
- Reduction in loss due to sampling:
  $E_n[F^* - \min(F^*,F(x))] = E_n[(F^* - F(x))^+] = EI(x)$



EI can make poor decisions

Recall: Expected improvement is Bayes-Optimal (in the noise-free standard BO problem) if:

- this is our last evaluation
- we are risk neutral
- we are only willing to select a previously evaluated point as a final solution

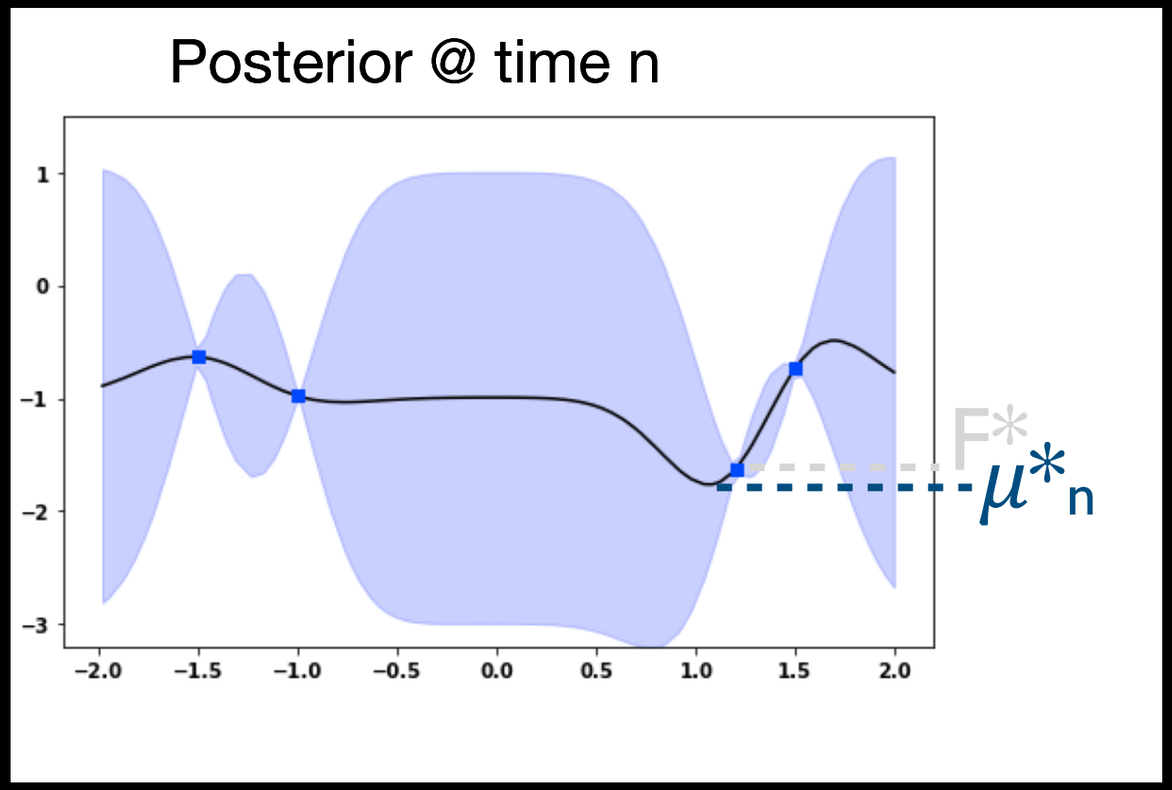
Let's drop one of these assumptions

- this is our last evaluation
- we are risk neutral
- ~~we are only willing to select a previously evaluated point as a final solution~~

This gives the Knowledge-Gradient acquisition function

- this is our last evaluation
- we are risk neutral
- ~~we are only willing to select a previously evaluated point as a final solution~~

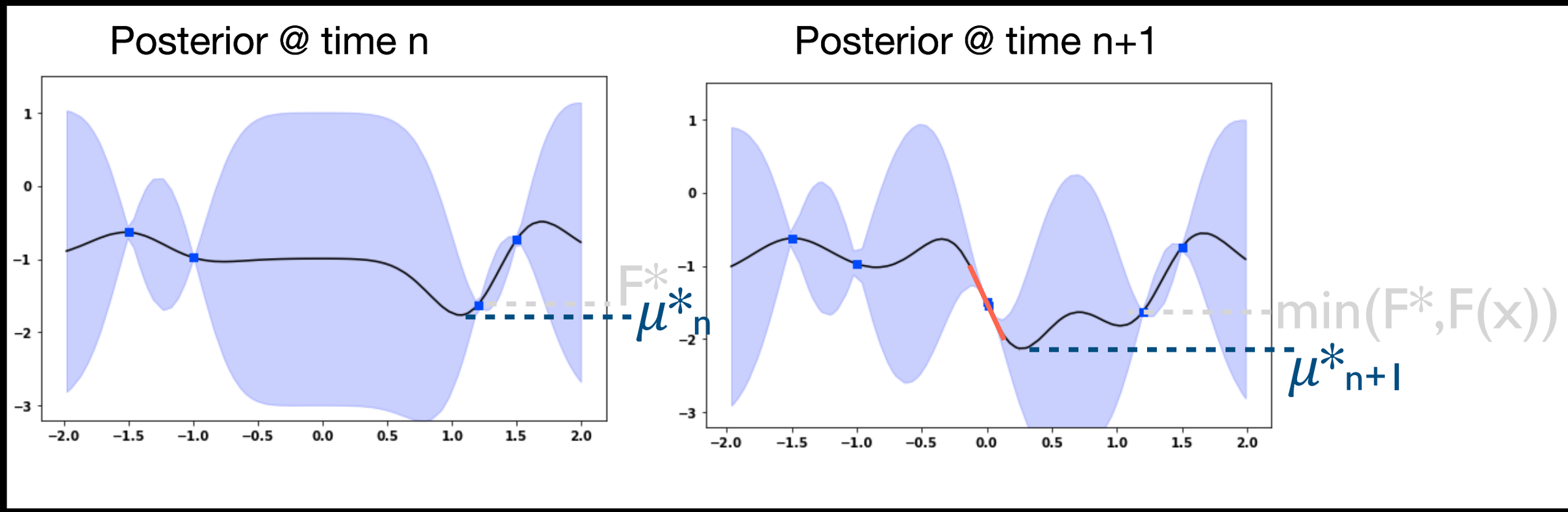
The KG acquisition function, KG(x)



- Loss if we stop now:
  $\mu^*_n = \min_x \mu_n(x)$

$\mu_n(x) := E_n[F(x)]$ is the expected value of our objective under the posterior @ time n

The KG acquisition function, KG(x)

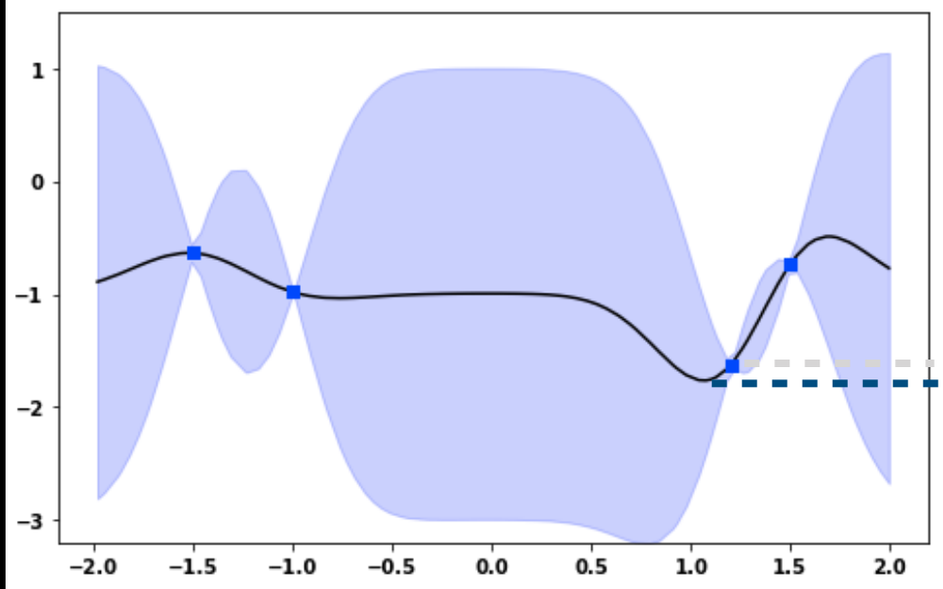


- Loss if we stop now:
  $\mu^*_n = \min_x \mu_n(x)$
- Loss if we stop after sampling F(x) and its gradient:
  $\mu^*_{n+1} = \min_x \mu_{n+1}(x)$

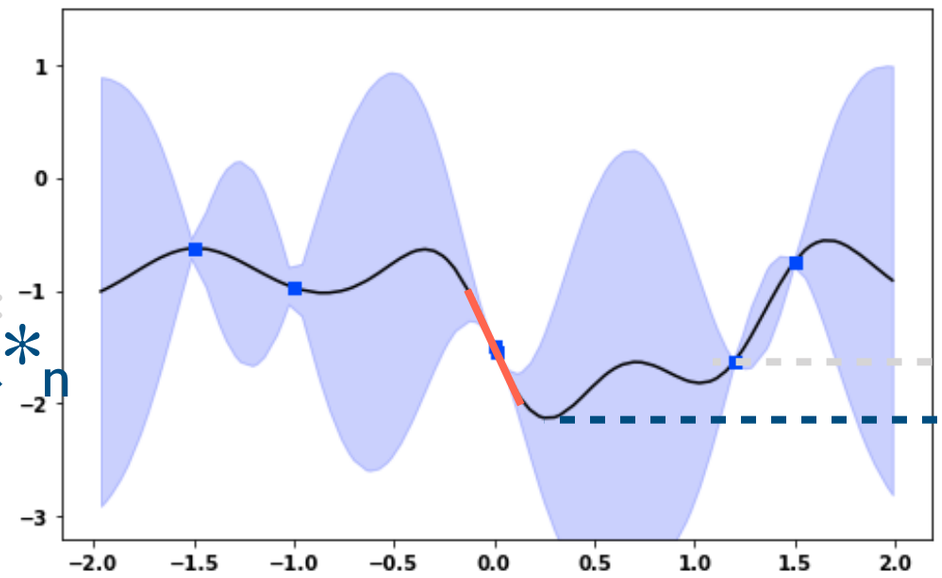
$\mu_n(x) := E_n[F(x)]$ is the expected value of our objective under the posterior @ time n

The KG acquisition function, KG(x)

Posterior @ time n



Posterior @ time n+1

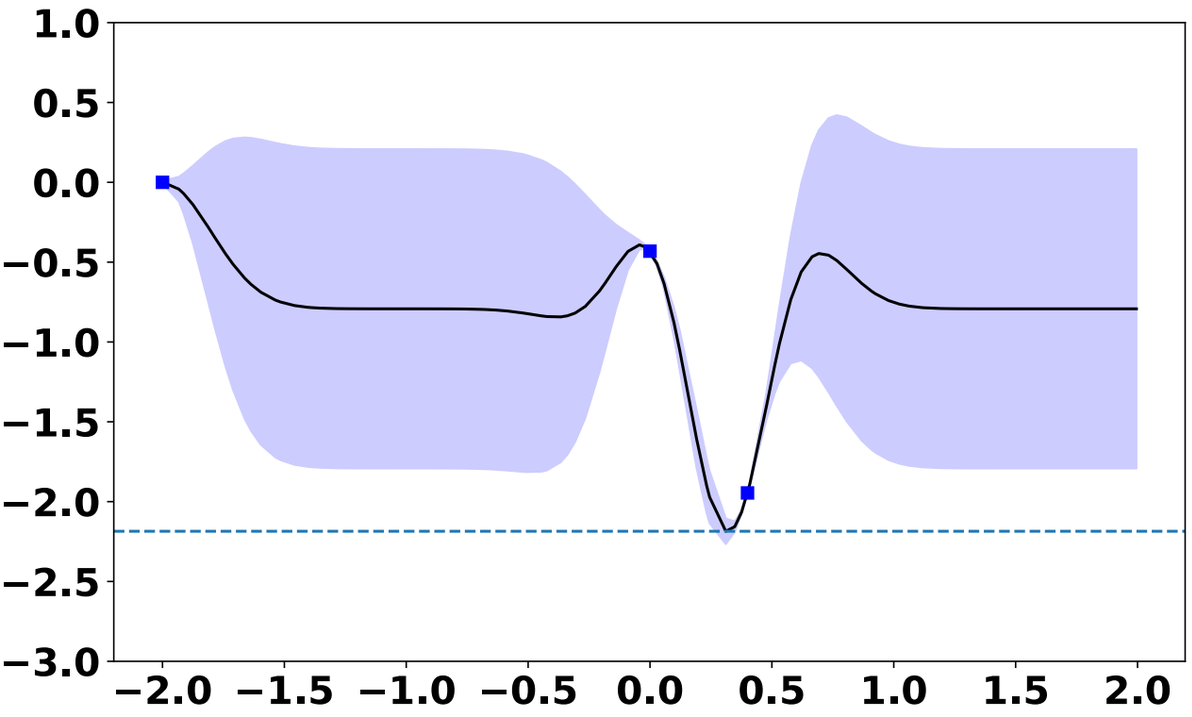


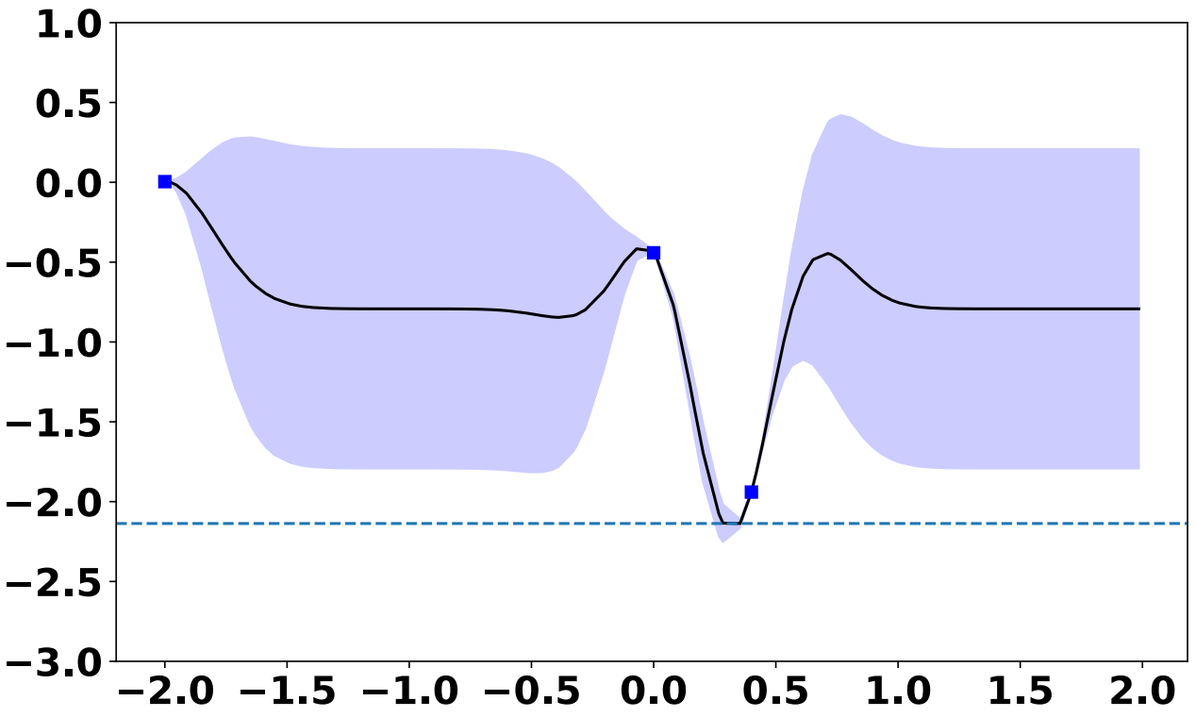
KG values this outcome; EI doesn't

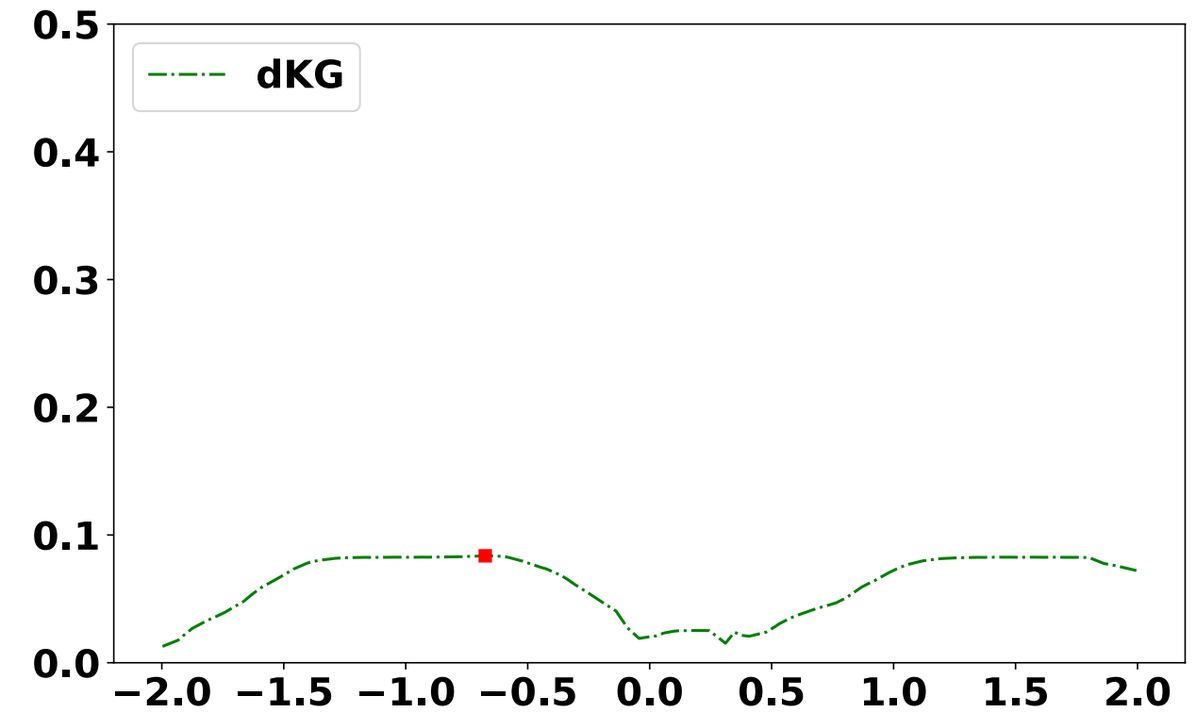
- Loss if we stop now:
  $\mu^*_n = \min_x \mu_n(x)$
- Loss if we stop after sampling F(x) and its gradient:
  $\mu^*_{n+1} = \min_x \mu_{n+1}(x)$
- Reduction in loss due to sampling:
  $KG(x) = E_n[\mu^*_n - \mu^*_{n+1} \mid \text{sample } x]$

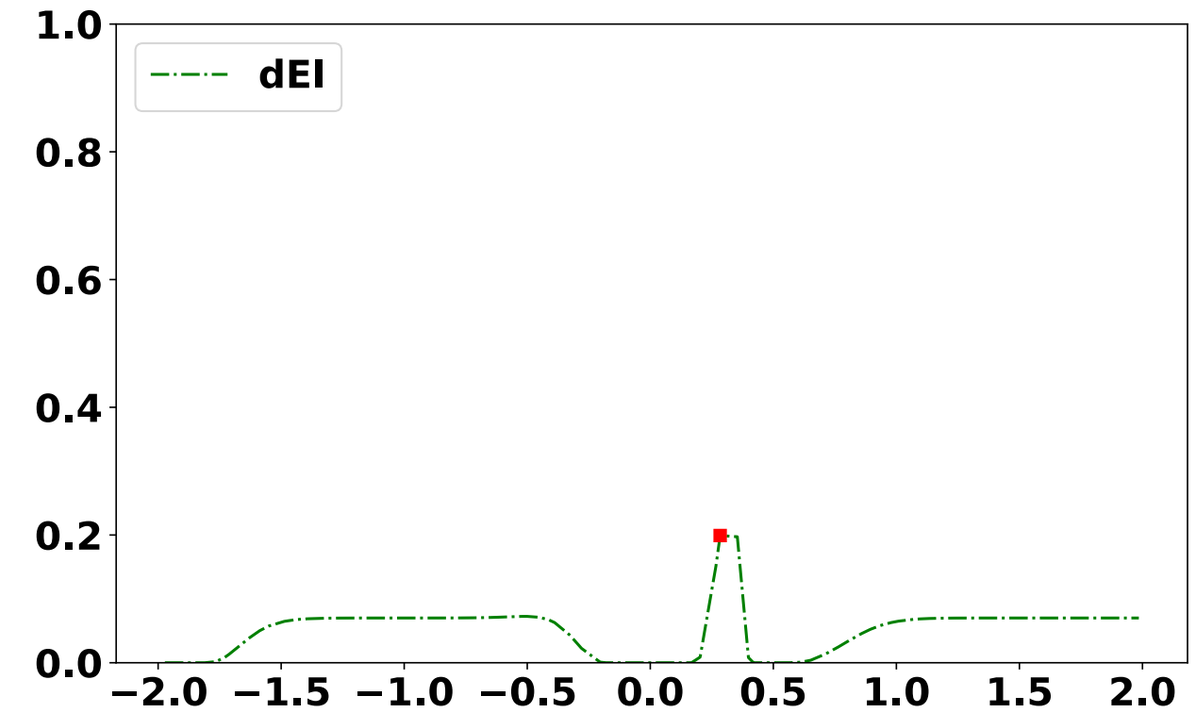
$\mu_n(x) := E_n[F(x)]$ is the expected value of our objective under the posterior @ time n

KG explores more effectively than EI when we have gradients

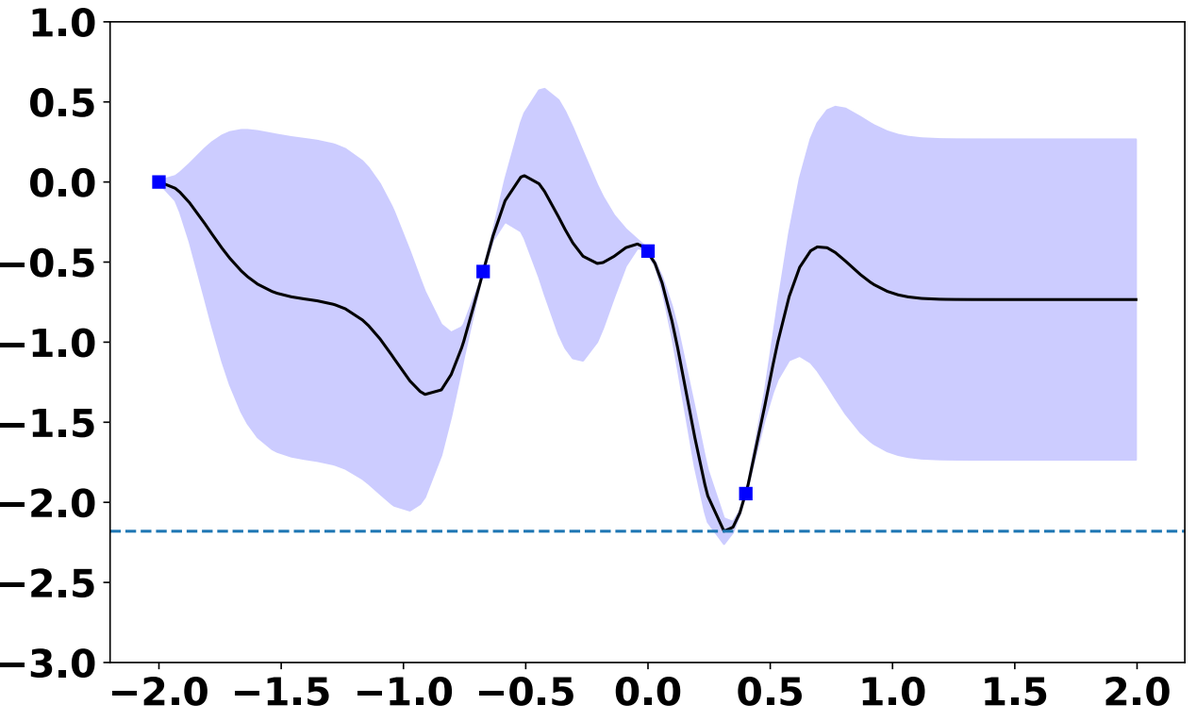


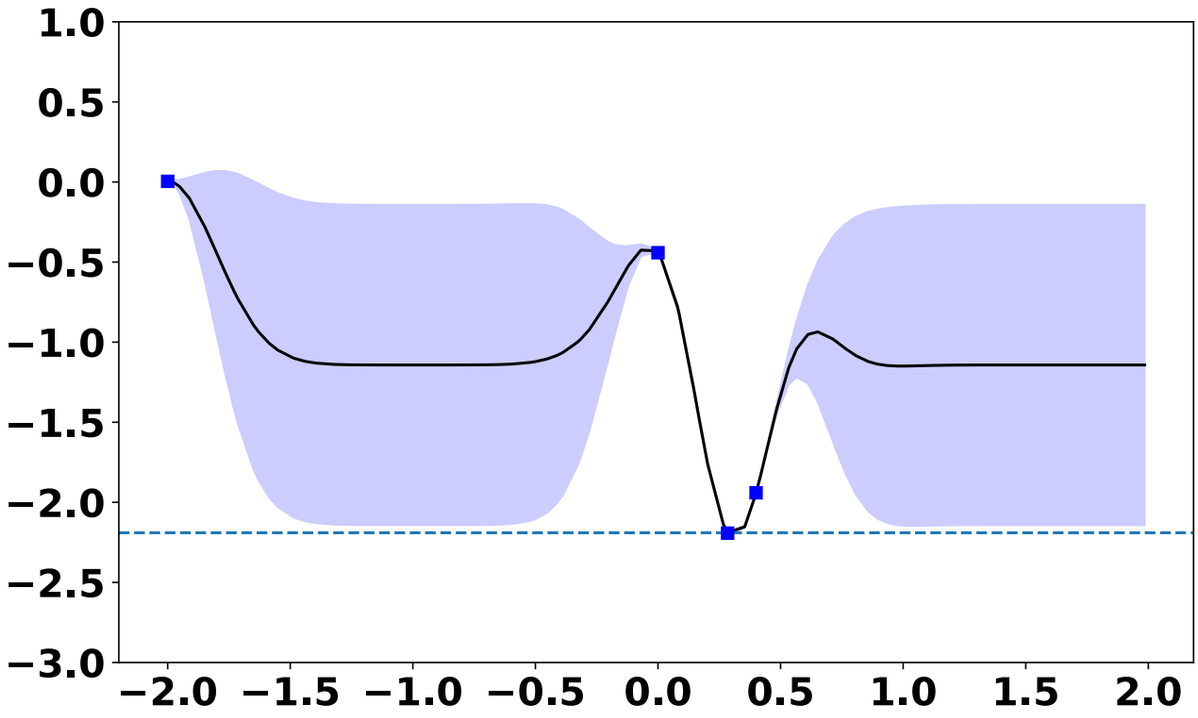


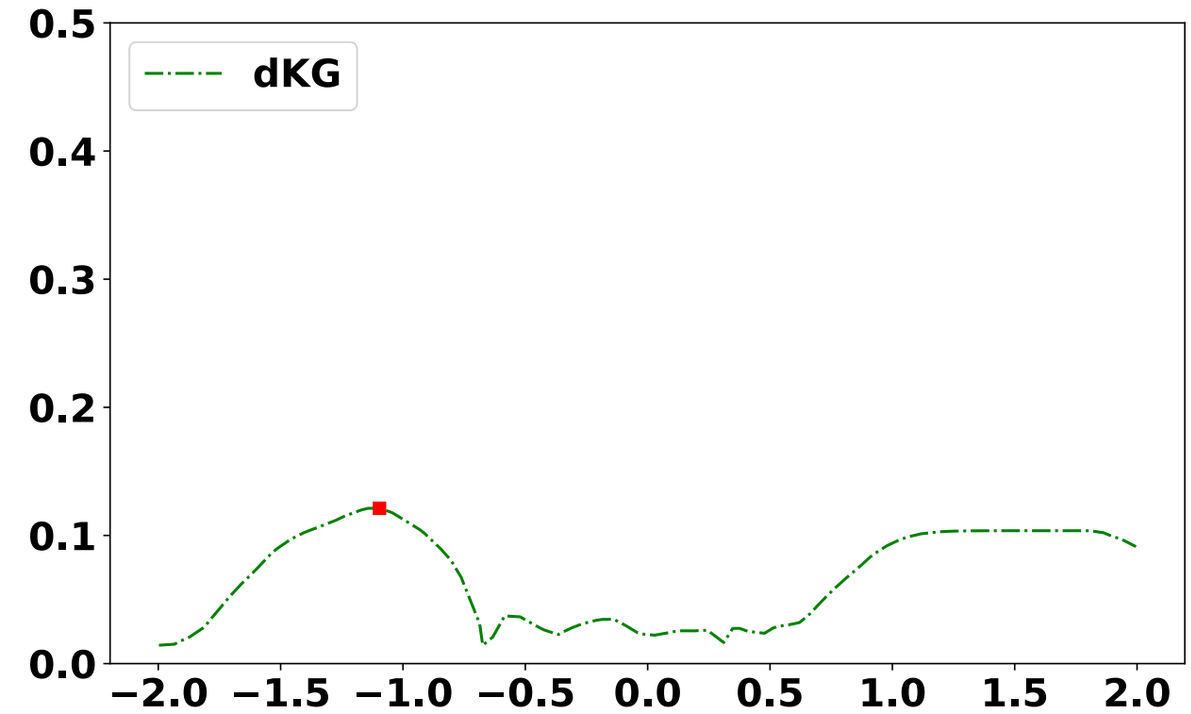


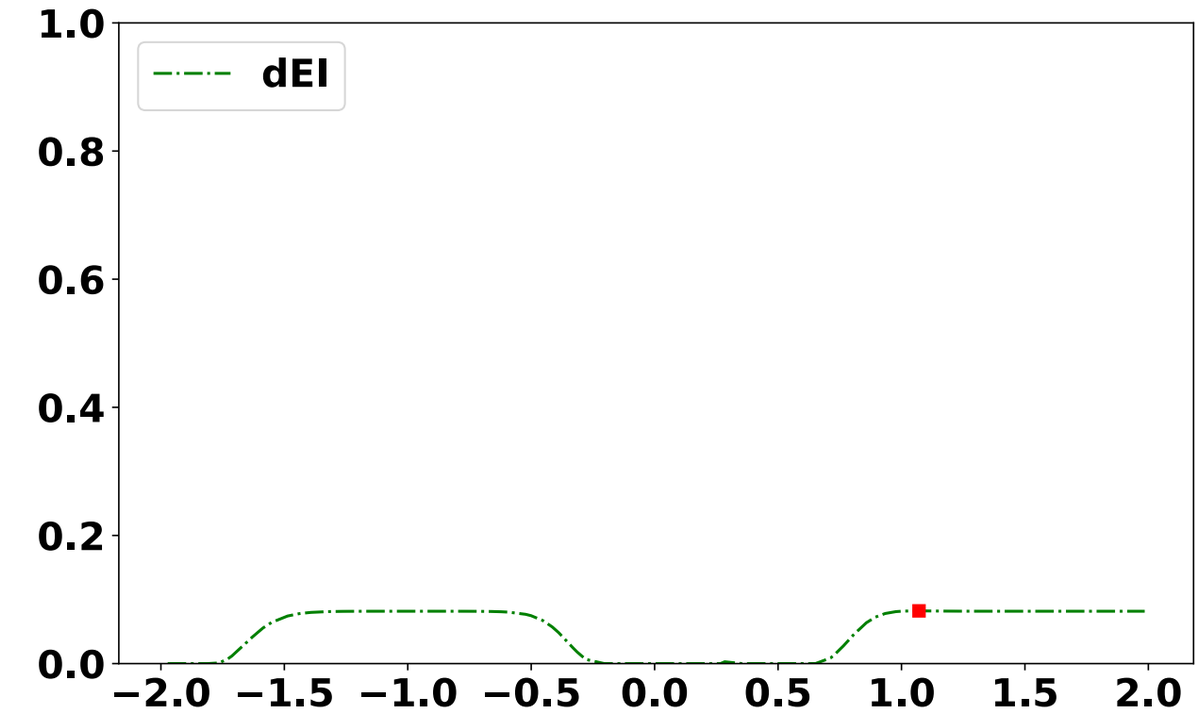


KG explores more effectively than EI when we have gradients

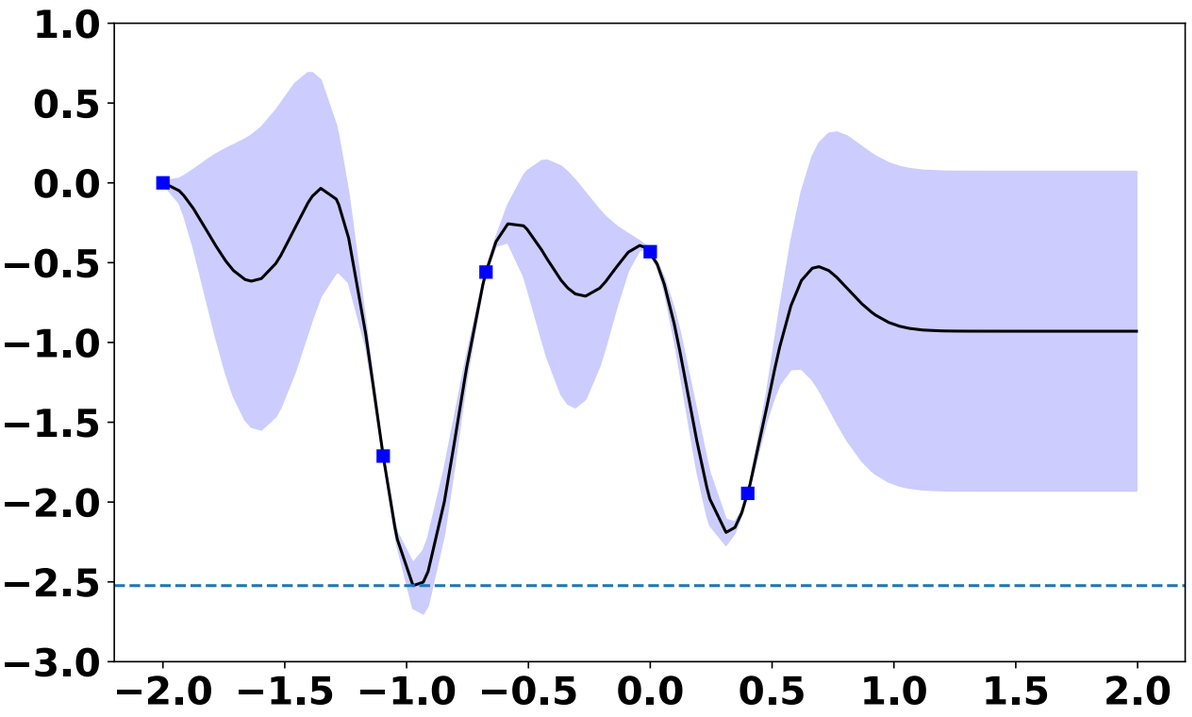


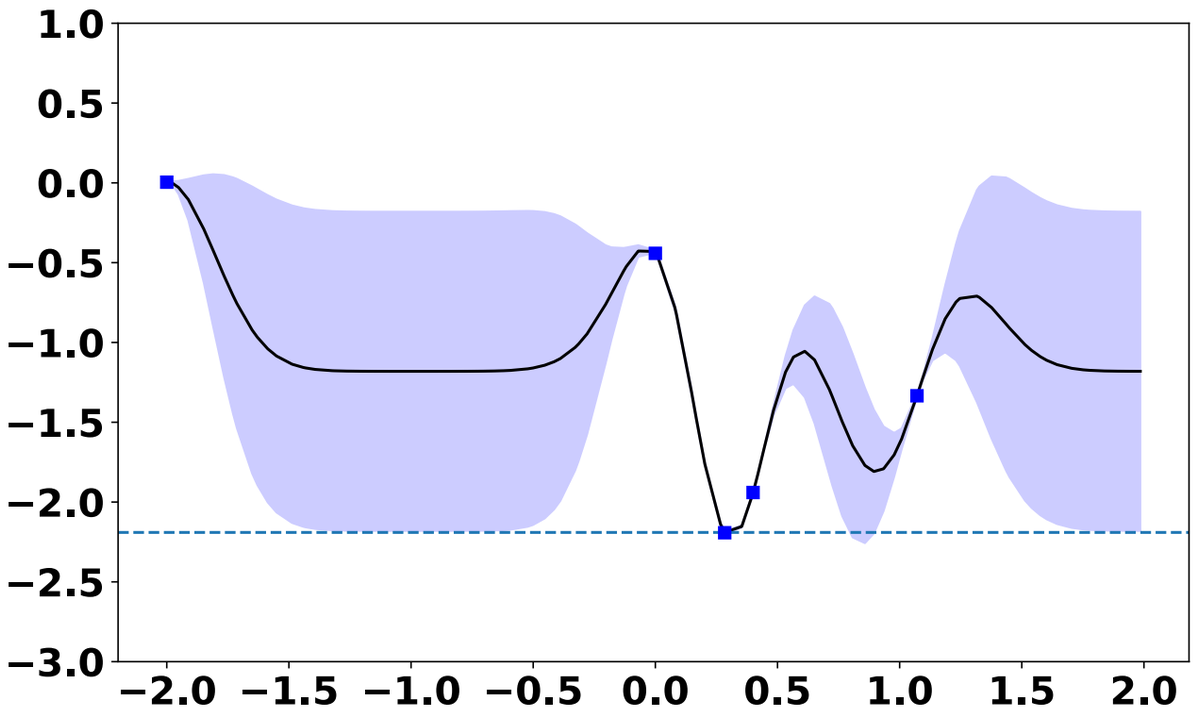


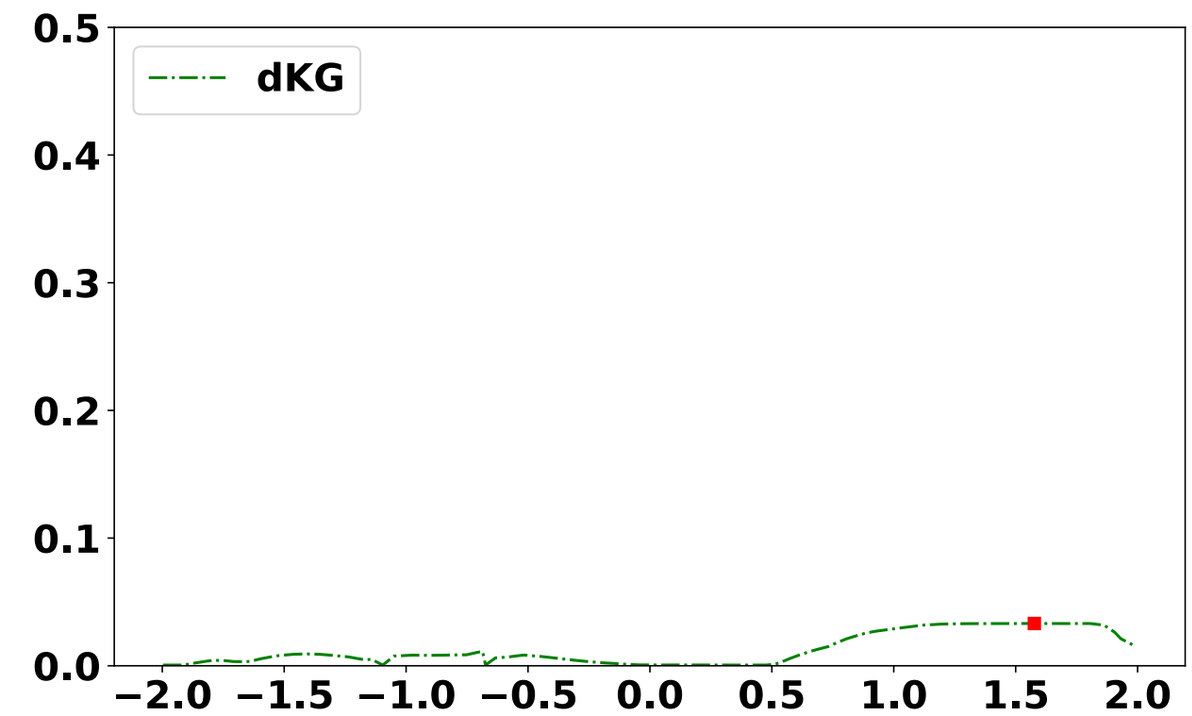


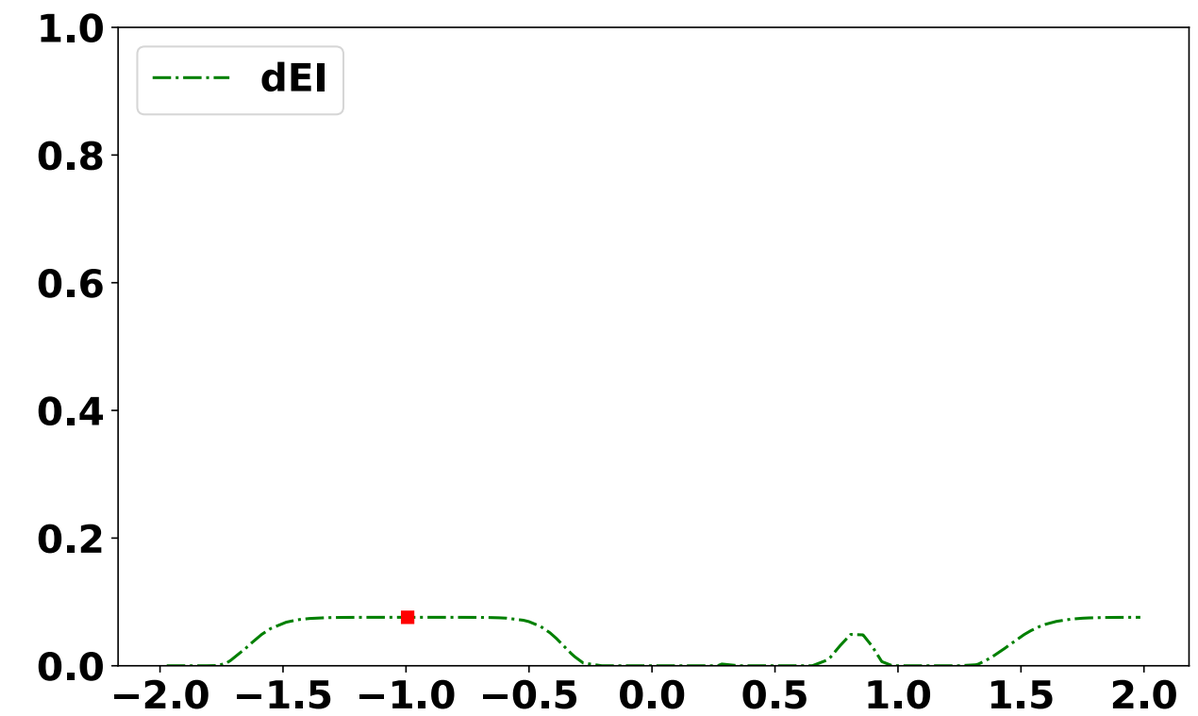


KG explores more effectively than EI when we have gradients

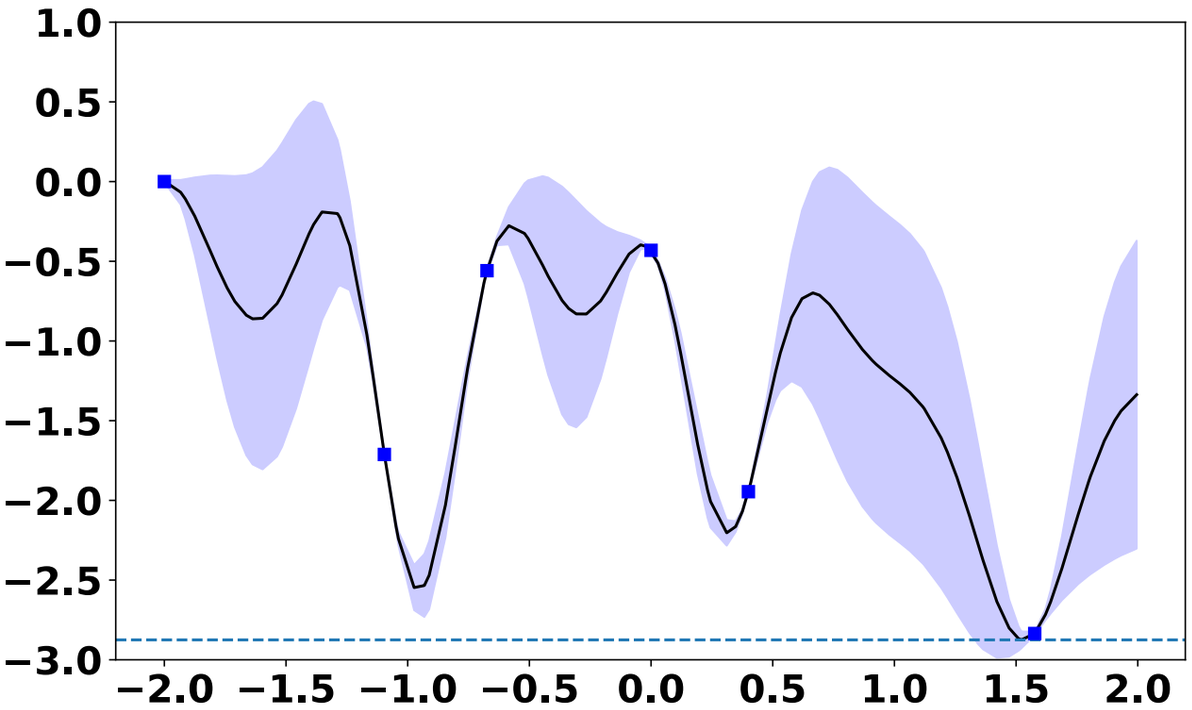


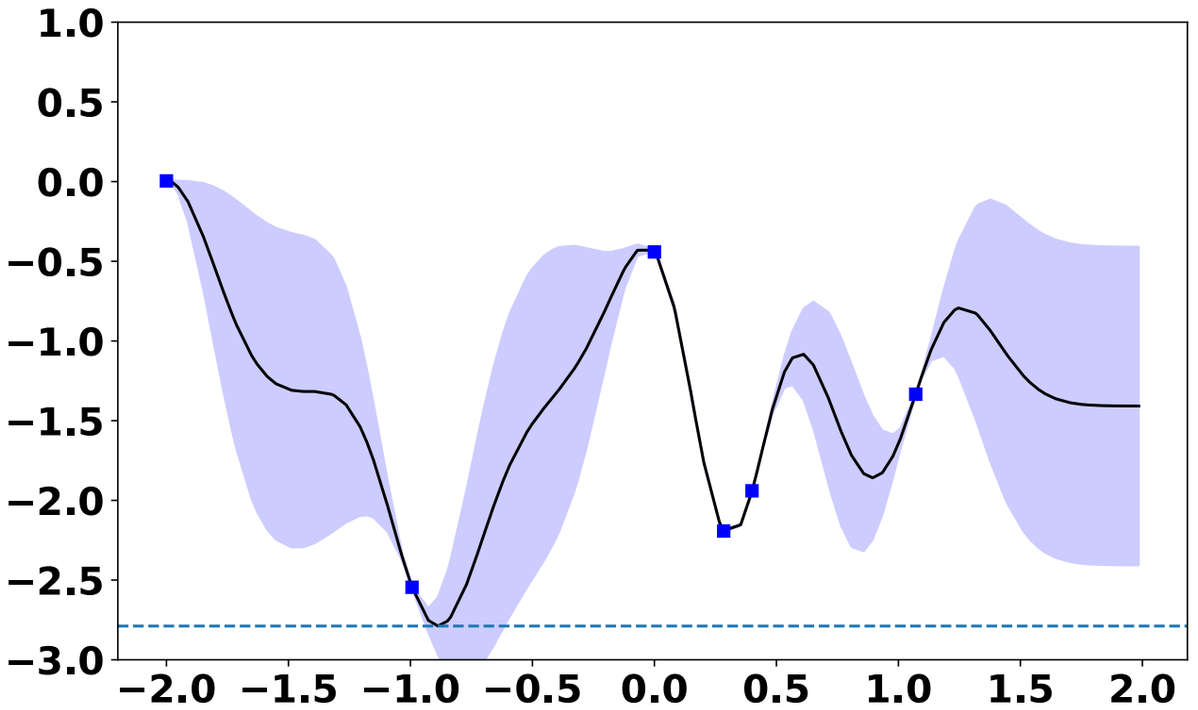


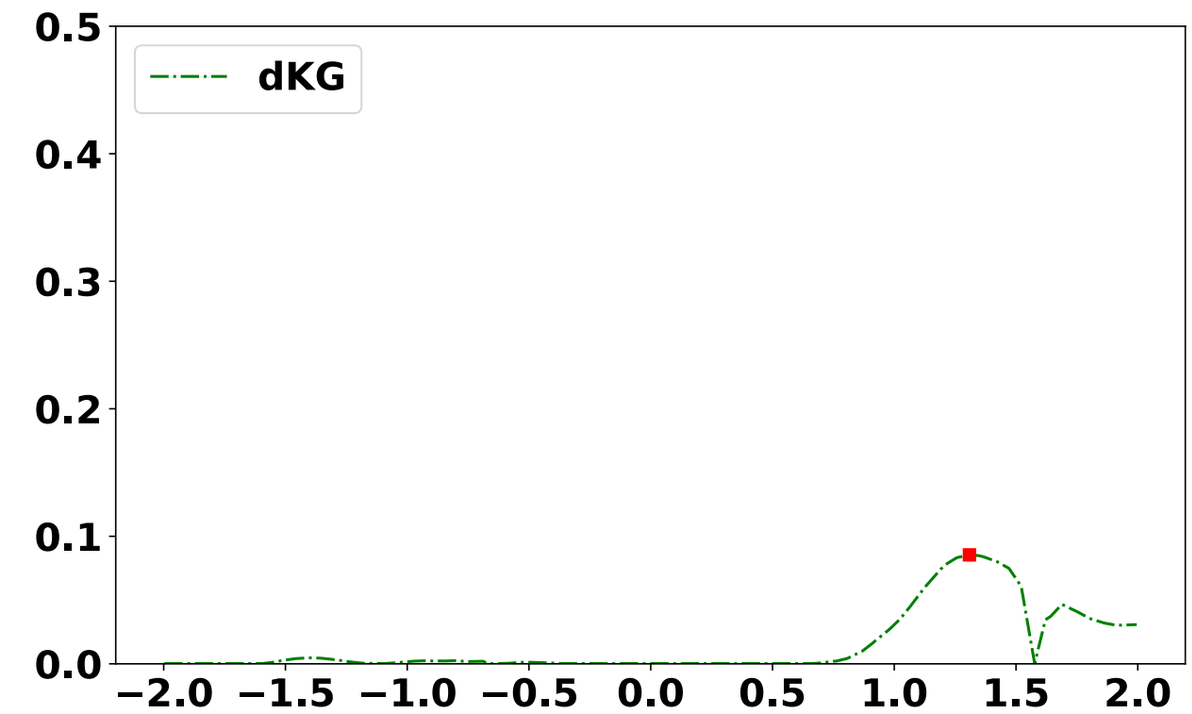


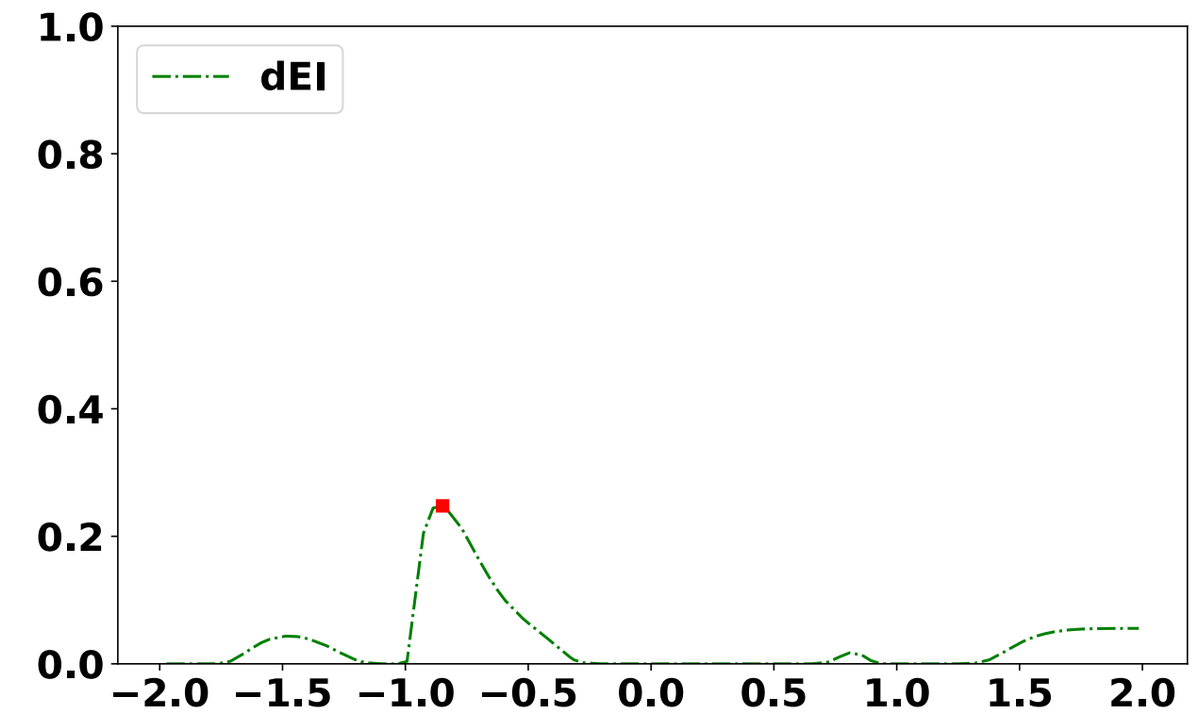


KG explores more effectively than EI when we have gradients

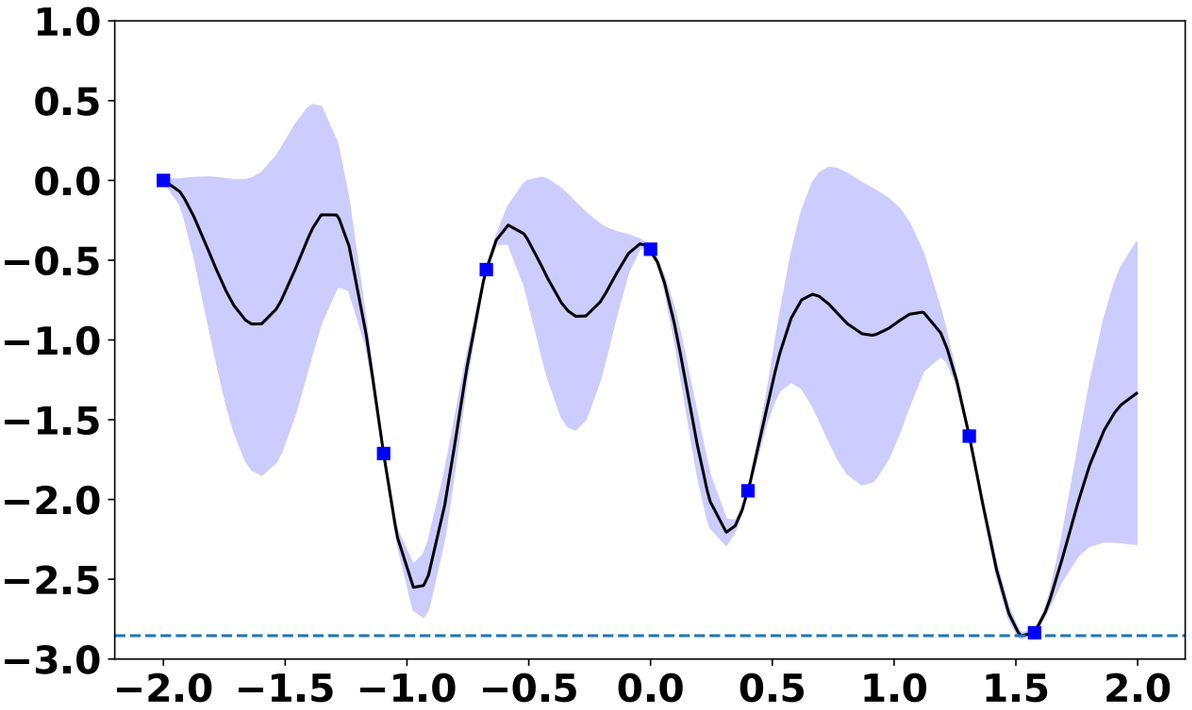


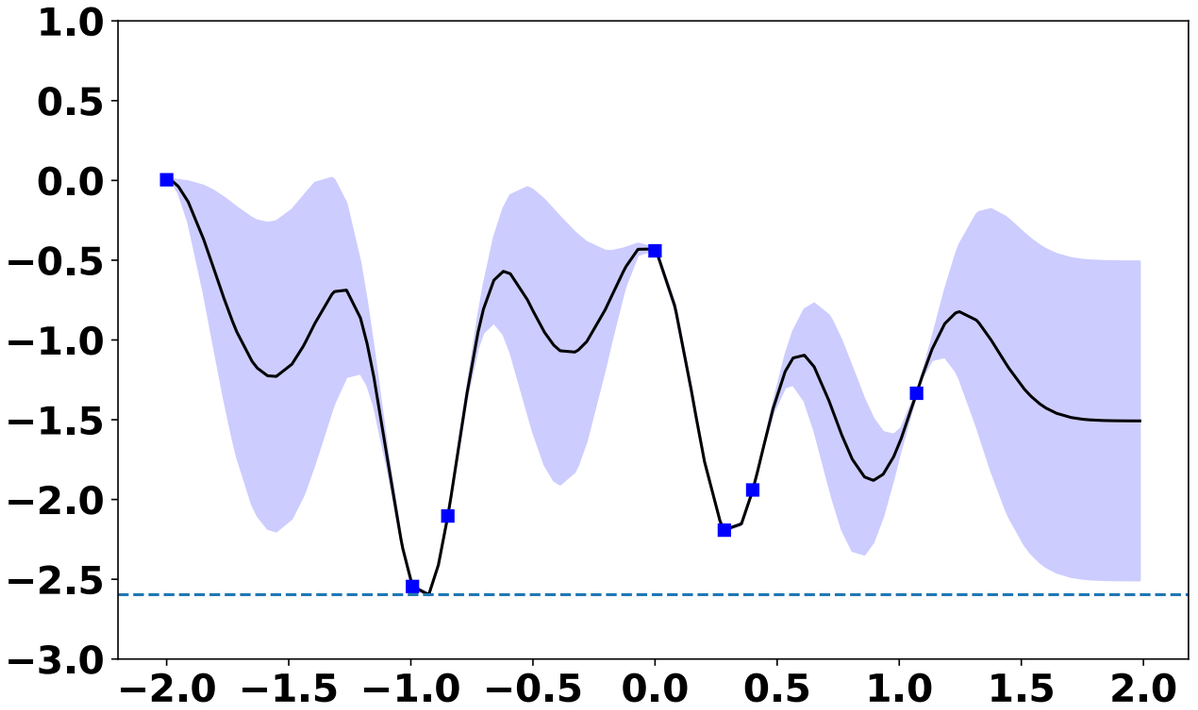


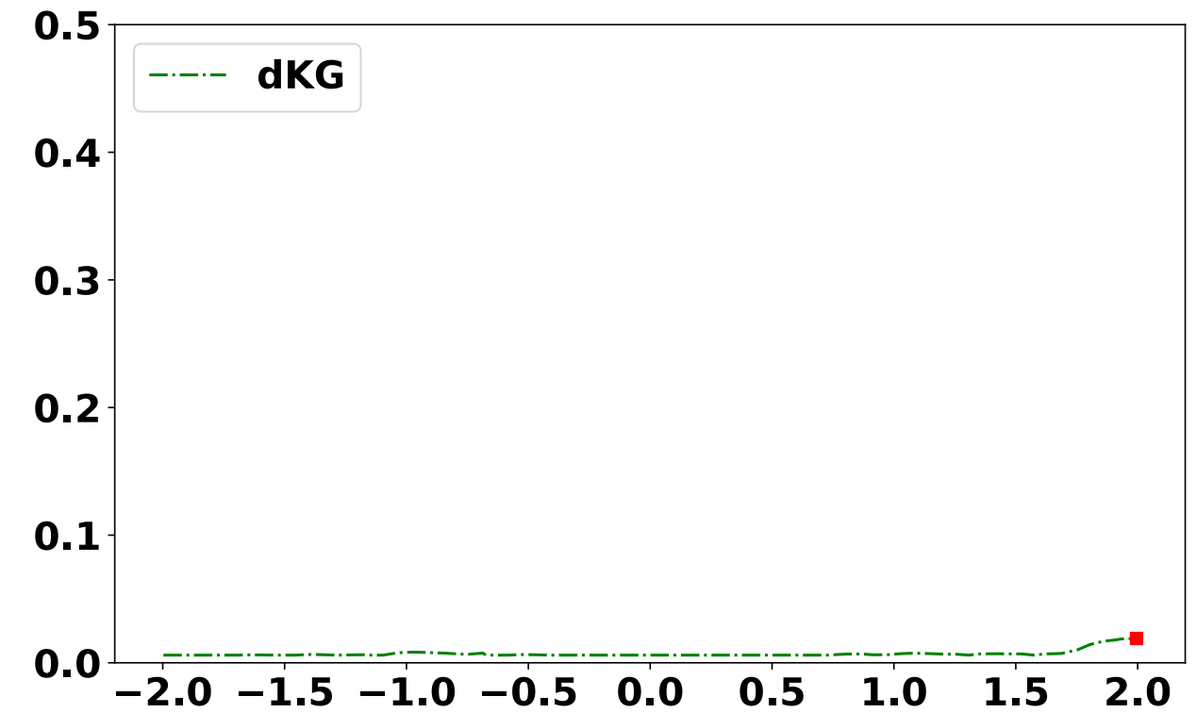


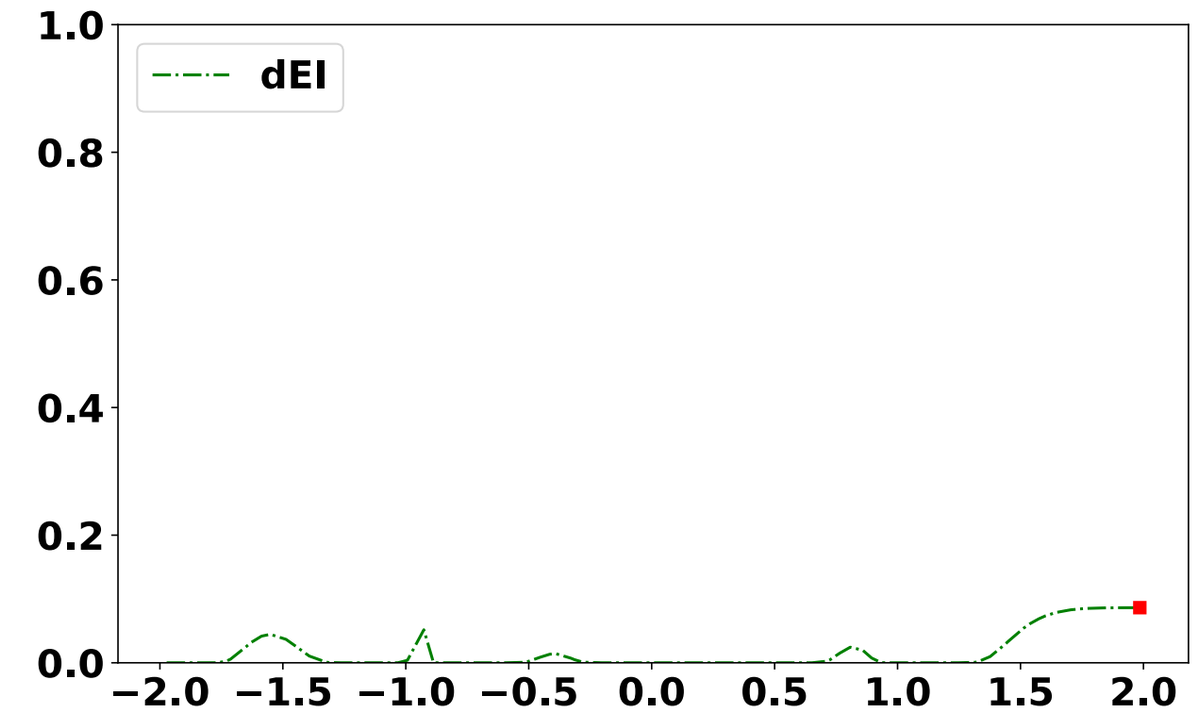


KG explores more effectively than EI when we have gradients









How can we optimize KG efficiently?

Here's how to **compute** KG

For i in 1:replications

- Simulate $Y(x_{1:q}) = [F(x_1),\nabla F(x_1),...,F(x_q),\nabla F(x_q)]$ from the posterior on F
  (plus noise, if observations are noisy)
- Calculate $\mu^*_{n+1} = \min_{x'} \mu_{n+1}(x')$ using $Y(x_{1:q})$
- Calculate $\mu^*_n - \mu^*_{n+1}$

$KG(x_{1:q})$ is the average of the simulated $\mu^*_n - \mu^*_{n+1}$

Our approach for optimizing parallel EI also works for optimizing KG

1. Estimate $\nabla KG(x_{1:q})$ using infinitesimal perturbation analysis (IPA) & the envelope theorem
2. Use multistart stochastic gradient ascent to maximize $KG(x_{1:q})$

Here's how to **estimate** $\nabla KG(x_{1:q})$

To estimate $\nabla KG(x_{1:q})$, use:

infinitesimal perturbation analysis (IPA) (Ho & Cao 2012)

$$\nabla KG(x) = \nabla E_n[\mu^*_n - \mu^*_{n+1}(Y(x),x)]$$
$$= E_n[\nabla \mu^*_n - \mu^*_{n+1}(Y(x),x)]$$

& the envelope theorem (Milgrom & Segal 2002)

$$\nabla \mu^*_{n+1}(Y(x),x) = \nabla \min_{x'} \mu_{n+1}(x';Y(x),x)$$
$$= \nabla \min_{x'} \mu_{n+1}(x'; m(x) + C(x)Z, x)$$
$$= \nabla \mu_{n+1}(x^*; m(x) + C(x)Z, x)$$

we write x instead of $x_{1:q}$ to reduce clutter

Here's how to **estimate** $\nabla KG(x_{1:q})$

1. Simulate a vector Z of independent standard normals

2. Calculate m = E[Y], C = Chol(Cov[Y]),
   where $Y = [F(x_1), \nabla F(x_1), \ldots, F(x_q), \nabla F(x_q)]$

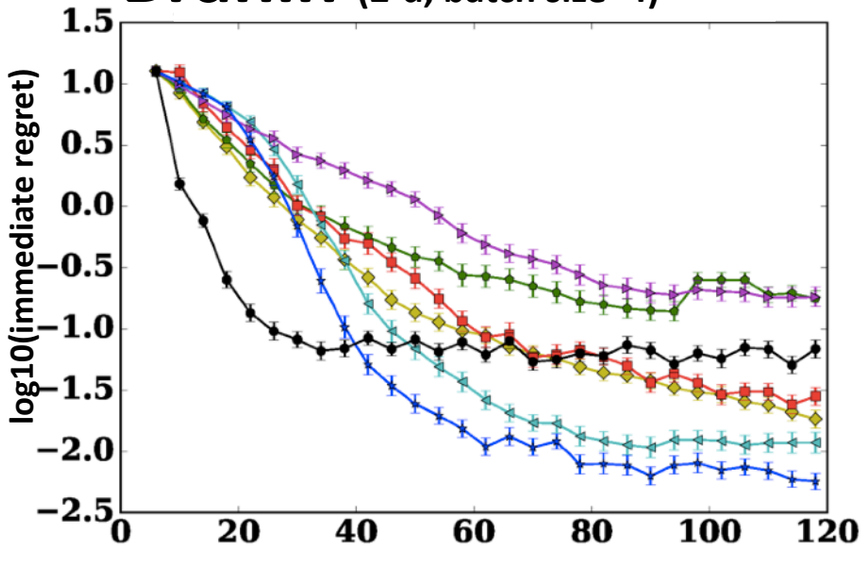
3. Solve $\min_{x'} \mu_{n+1}(x'; m(x) + C(x)Z, x)$ using L-BFGS-B.
   Let x* be the solution.

4. Our estimator of $\nabla KG(x)$ is $\nabla \mu_{n+1}(x^*; m(x) + C(x)Z, x)$,
   calculating the gradient holding x* fixed

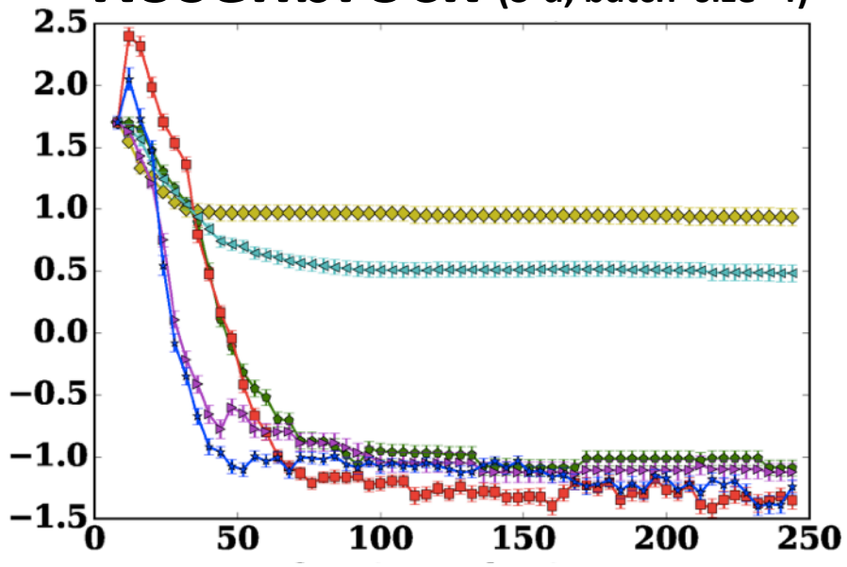
we write x instead of $x_{1:q}$ to reduce clutter

KG outperforms EI substantially, if you observe gradients

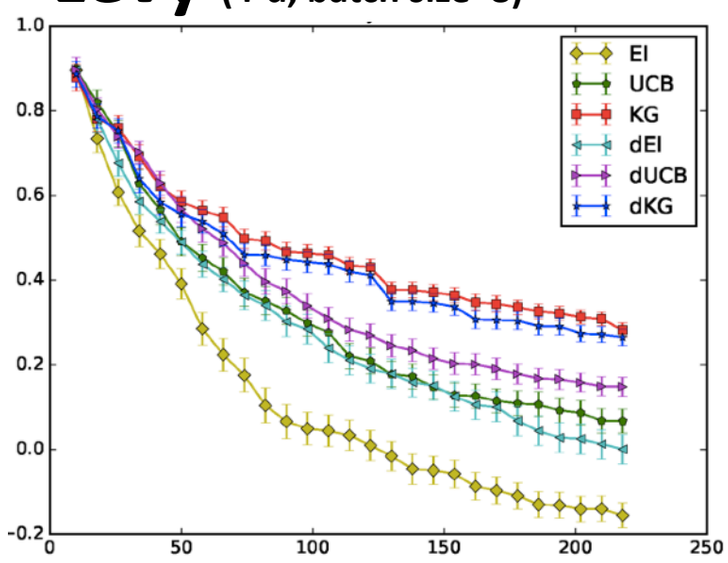
Branin (2-d, batch size=4)



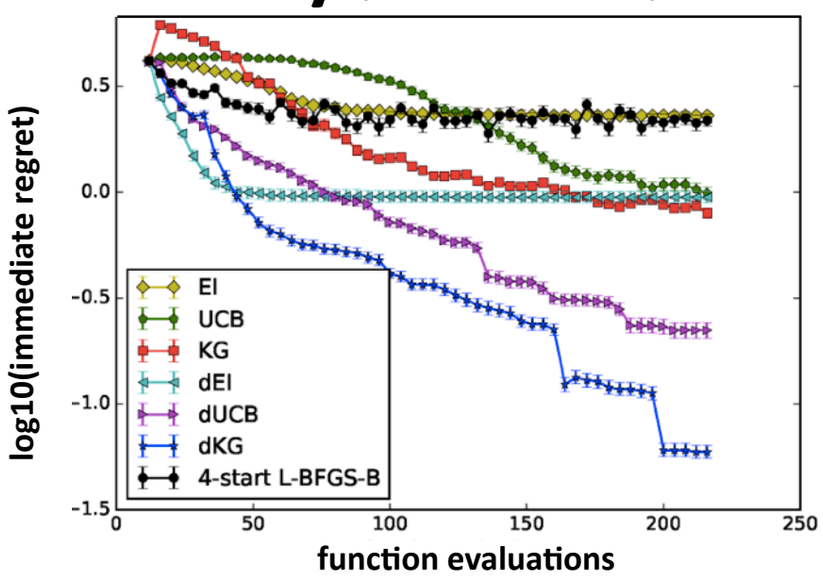
Rosenbrock (3-d, batch size=4)



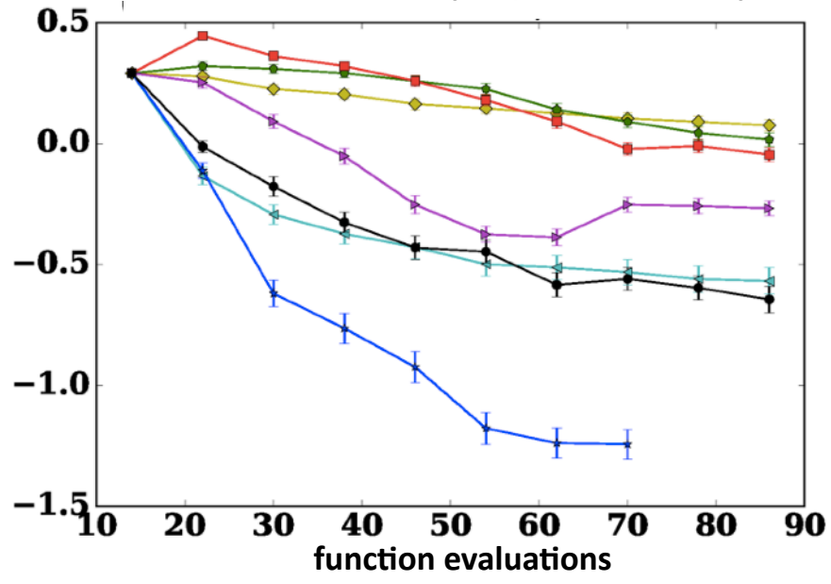
Levy (4-d, batch size=8)



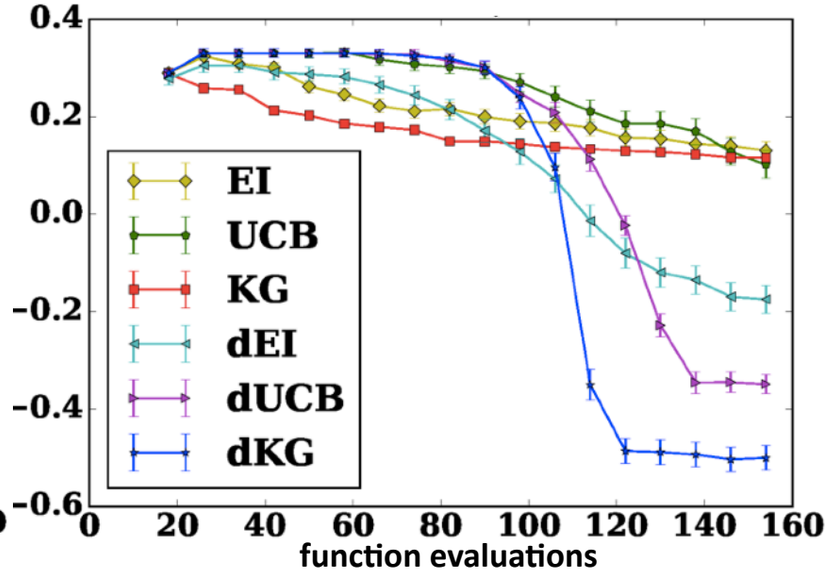
Ackley (5-d, batch size=4)



Hartmann (6-d, batch size=8)

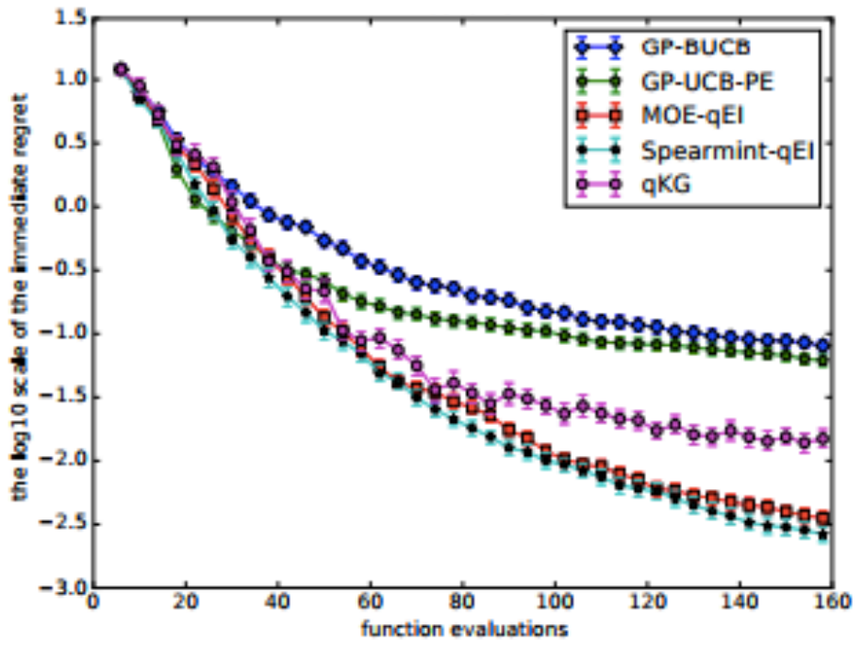


Cosine (8-d, batch size=4)

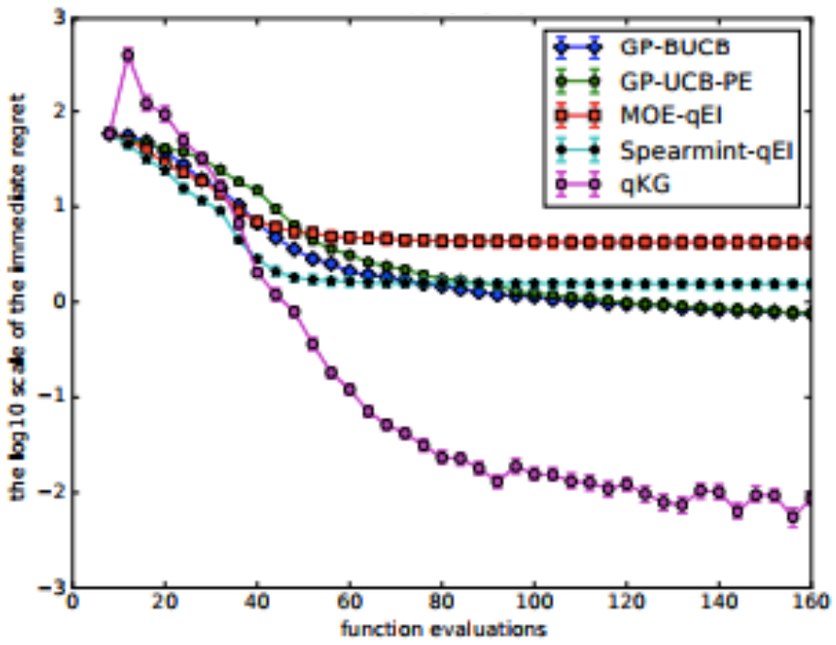


KG outperforms EI substantially if you **don't observe gradients**, but have noise or are in dim > 1

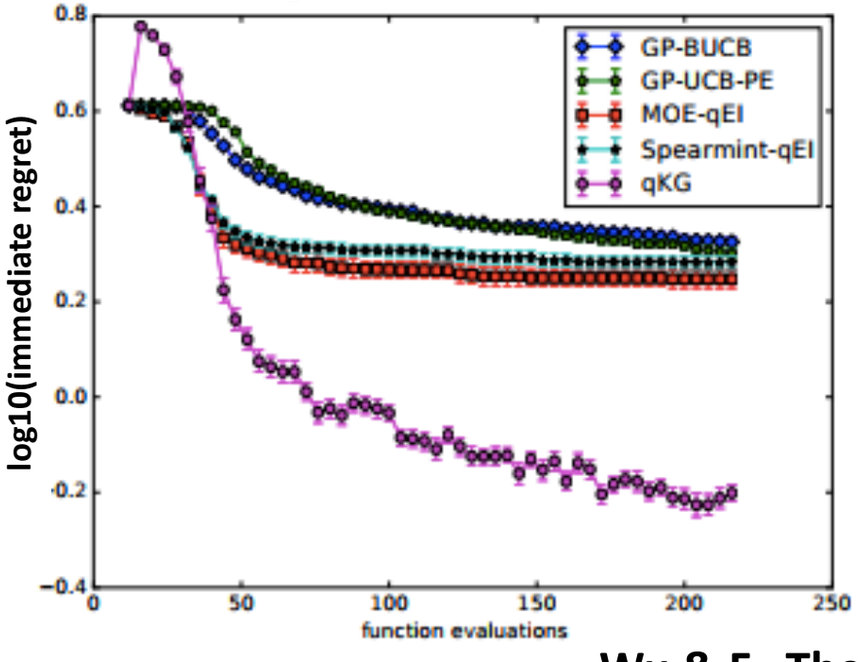
Branin (2-d, batch size=4, no noise)



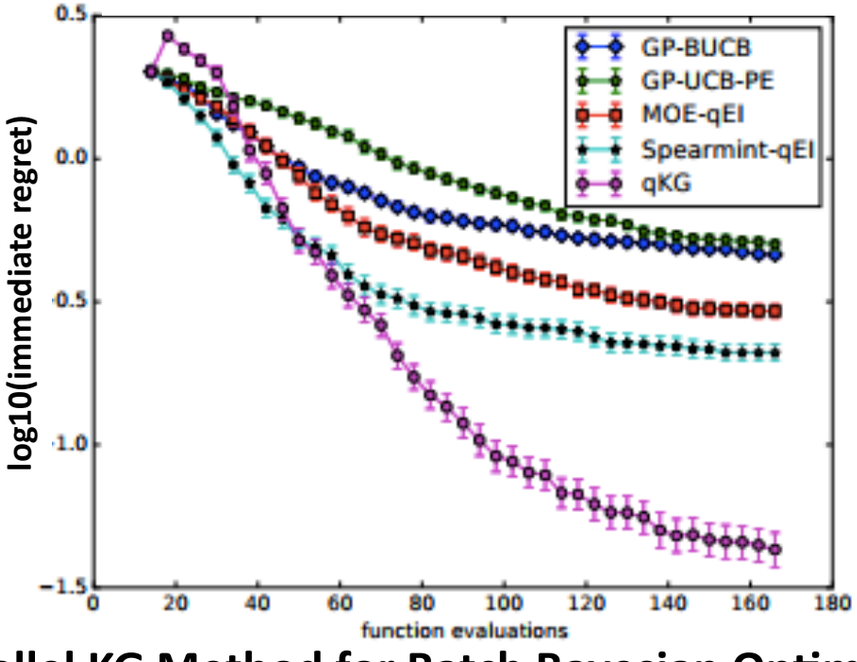
Rosenbrock (3-d, batch size=4, no noise)



Ackley (5-d, batch size=4, no noise)



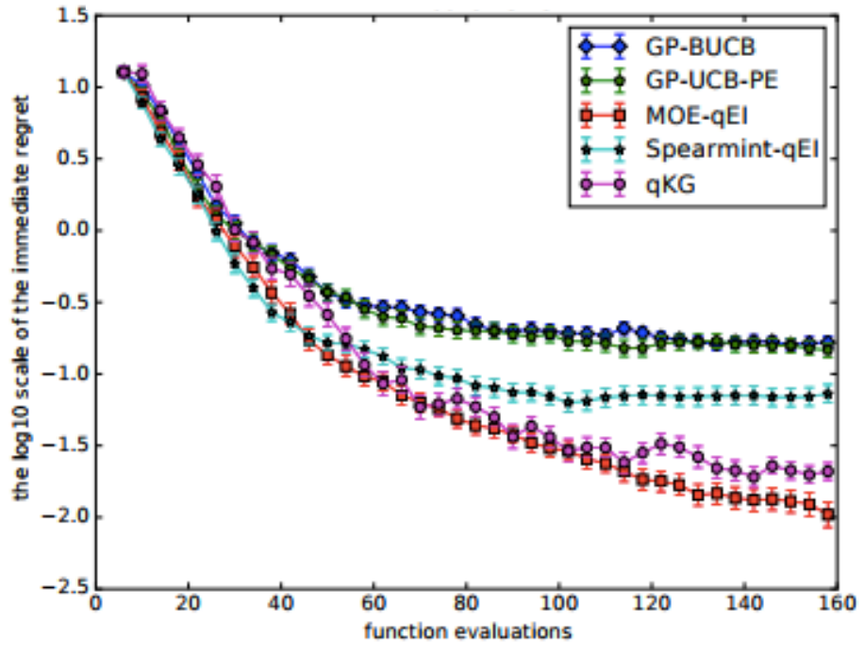
Hartmann (6-d, batch size=4, no noise)



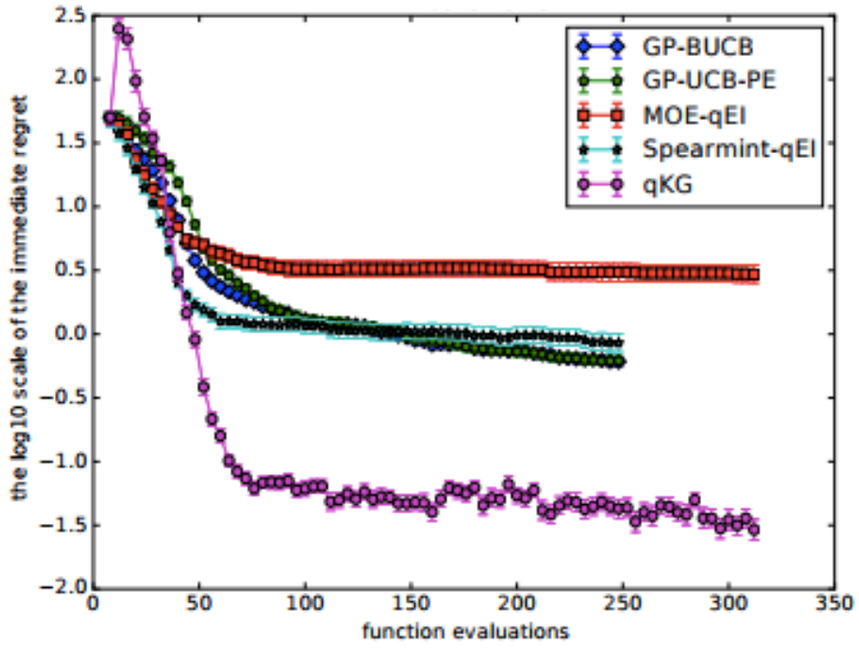
Wu & F., The Parallel KG Method for Batch Bayesian Optimization, NIPS'16

KG outperforms EI substantially if you **don't observe gradients**, but have noise or are in dim > 1

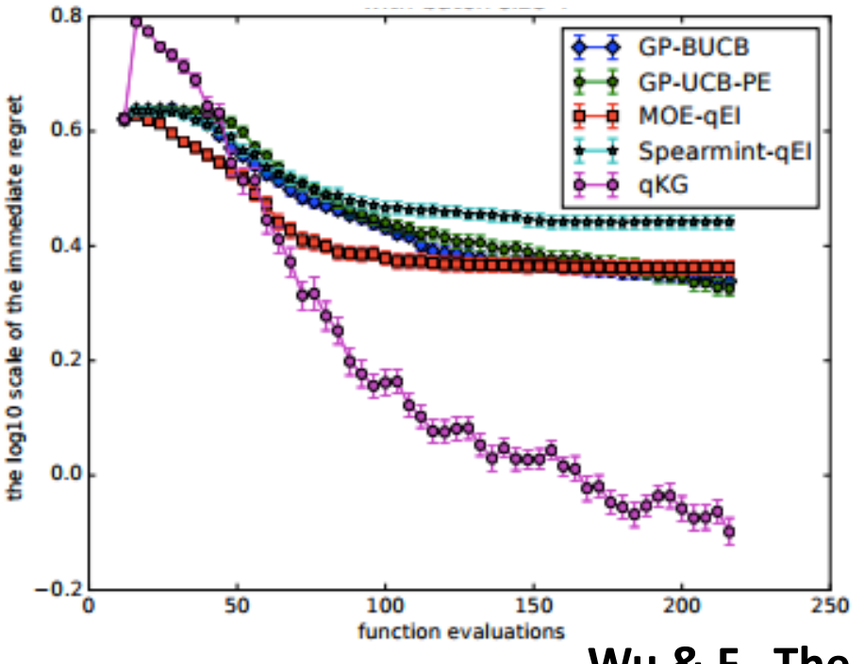
Branin (2-d, batch size=4, noisy)



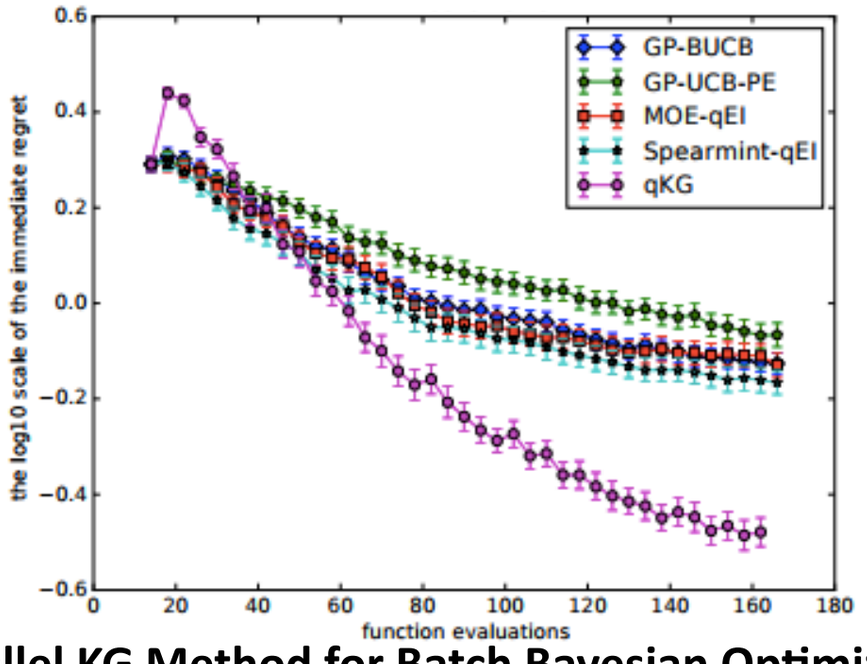
Rosenbrock (3-d, batch size=4, noisy)



Ackley (5-d, batch size=4, noisy)



Hartmann (5-d, batch size=4, noisy)



Wu & F., The Parallel KG Method for Batch Bayesian Optimization, NIPS'16

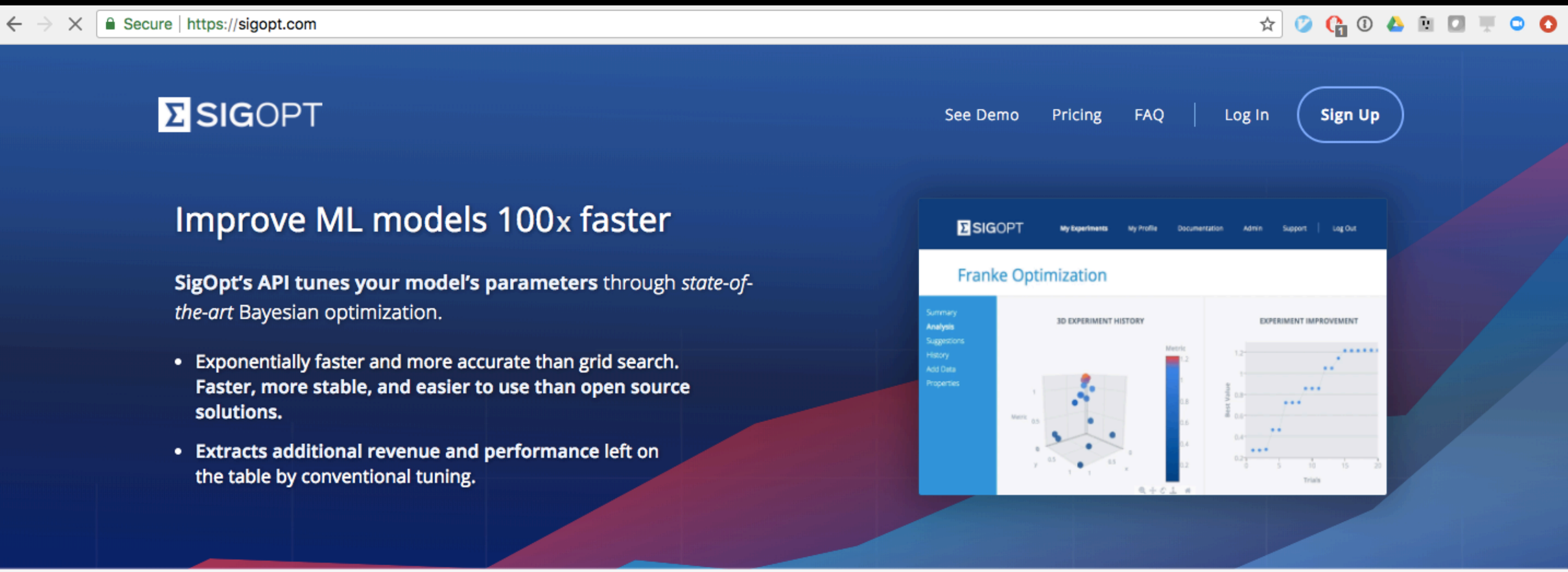
Agenda

- Gaussian process regression
- Expected improvement (EI)
- Parallel BayesOpt (using EI)
- Parallel BayesOpt with gradients and/or noise (using KG)
- **Code & Research directions**

Here are some open-source codes for Bayesian optimization

- Cornell Metrics Optimization Engine (MOE): https://github.com/wujian16/Cornell-MOE based on Yelp's MOE codebase https://github.com/Yelp/MOE
- GPyOpt: https://github.com/SheffieldML/GPyOpt
- Spearmint: https://github.com/JasperSnoek/spearmint

There are also commercial services



What didn't we cover?

- Statistical models beyond Gaussian processes
  - Warping
  - Student-t processes
  - Neural networks
  - Models for high-dimensional problems
- Other acquisition functions:
  - Entropy Search and predictive entropy search
  - Thompson sampling
  - Probability of improvement
- Convergence results & regret guarantees
- Problems with exotic structure
  - Multi-fidelity & multi-information source optimization
  - Optimization with trace observations
  - Multi-task optimization

Research Directions

- Applications
- Why do 1-step algorithms work so well?
  Meaningful theoretical guarantees
- Going beyond Gaussian processes
- High-dimensional BayesOpt
- Problems with exotic structure

Thanks! Any Questions?

- Cornell Metrics Optimization Engine (MOE): https://github.com/wujian16/Cornell-MOE
- "A Tutorial on Bayesian Optimization" https://arxiv.org/abs/1807.02811

Backup

KG converges to a globally optimal solution over a discrete domain

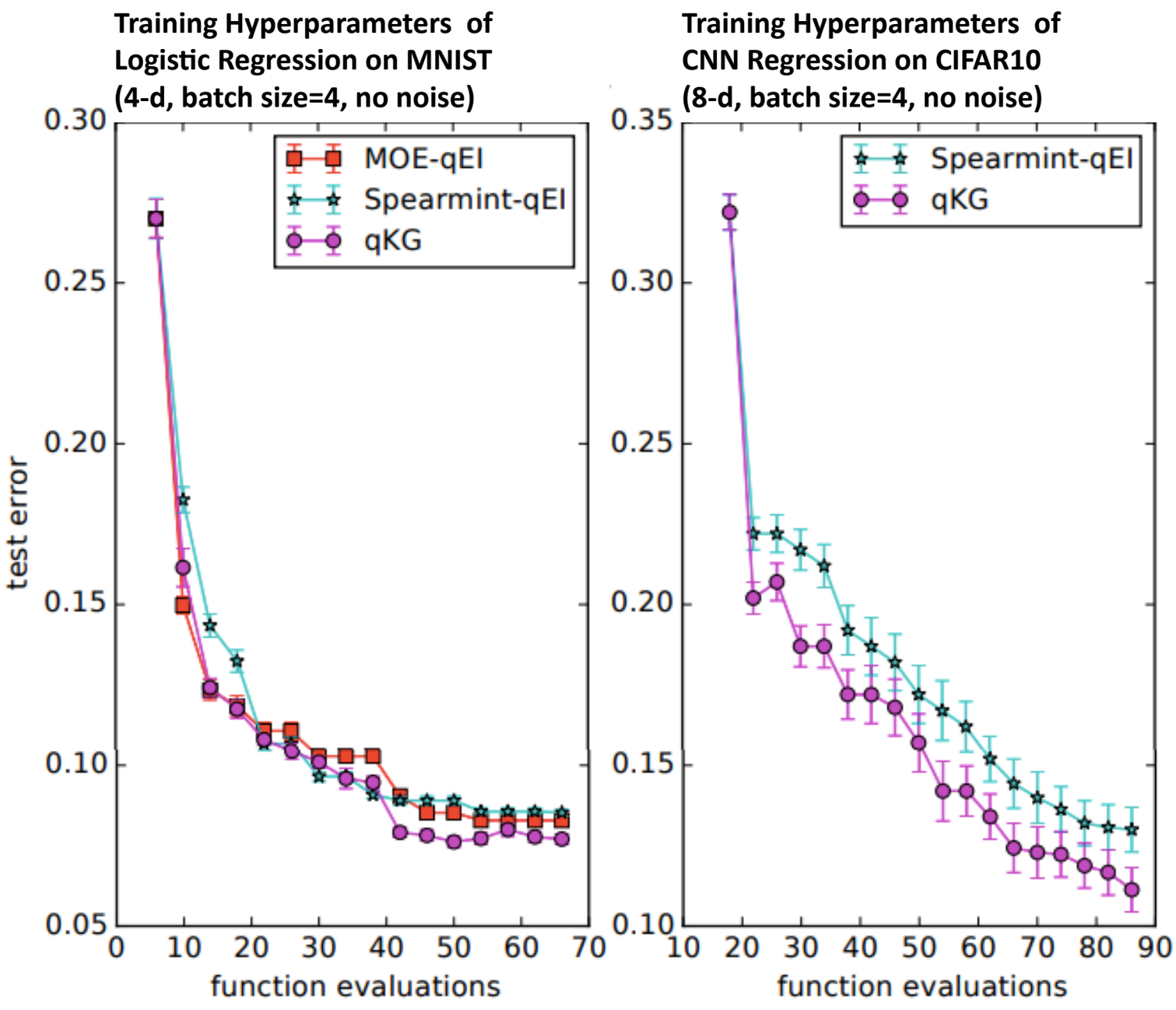
Theorem: When the domain A is discrete & finite, the KG algorithm is consistent, i.e.,

$$\lim_{N\to\infty} f(x^*(\mathrm{dKG}, N)) = \max_{x\in A} f(x)$$

almost surely under the prior, where *x*(KG,N)* is the solution computed by KG after N batches of samples.

Wu, Poloczek, Wilson & F., Bayesian Optimization with Gradients, NIPS'17

KG provides substantial value over EI when we **don't have gradients** when there is noise, or we are in > 1 dim.



Wu & F., The Parallel KG Method for Batch Bayesian Optimization, NIPS'16

GP regression handles gradients naturally [Rasmussen & Williams 2006]

1. $f \sim GP(\mu, K)$
2. $x \rightarrow [f(x), \nabla f(x)]$ is a vector-valued function.
3. Our GP prior on f implies this GP prior on $f, \nabla f$:

$$\begin{bmatrix} f \\ \nabla f \end{bmatrix} \sim \mathrm{GP}\left(\begin{bmatrix} \mu \\ \nabla \mu \end{bmatrix}, \begin{bmatrix} K & \nabla K \\ \nabla K & \nabla^2 K \end{bmatrix} \right)$$

   (Assuming some regularity conditions on K)

4. Do standard GP regression on $f, \nabla f$
   Calculate the posterior on the objective f as needed

Bayesian optimization is really **flexible**

Elicit a prior distribution on the function f

while (budget is not exhausted) {

Find the information source whose value of information is the largest.

Query that information source

Update the posterior distribution.

}

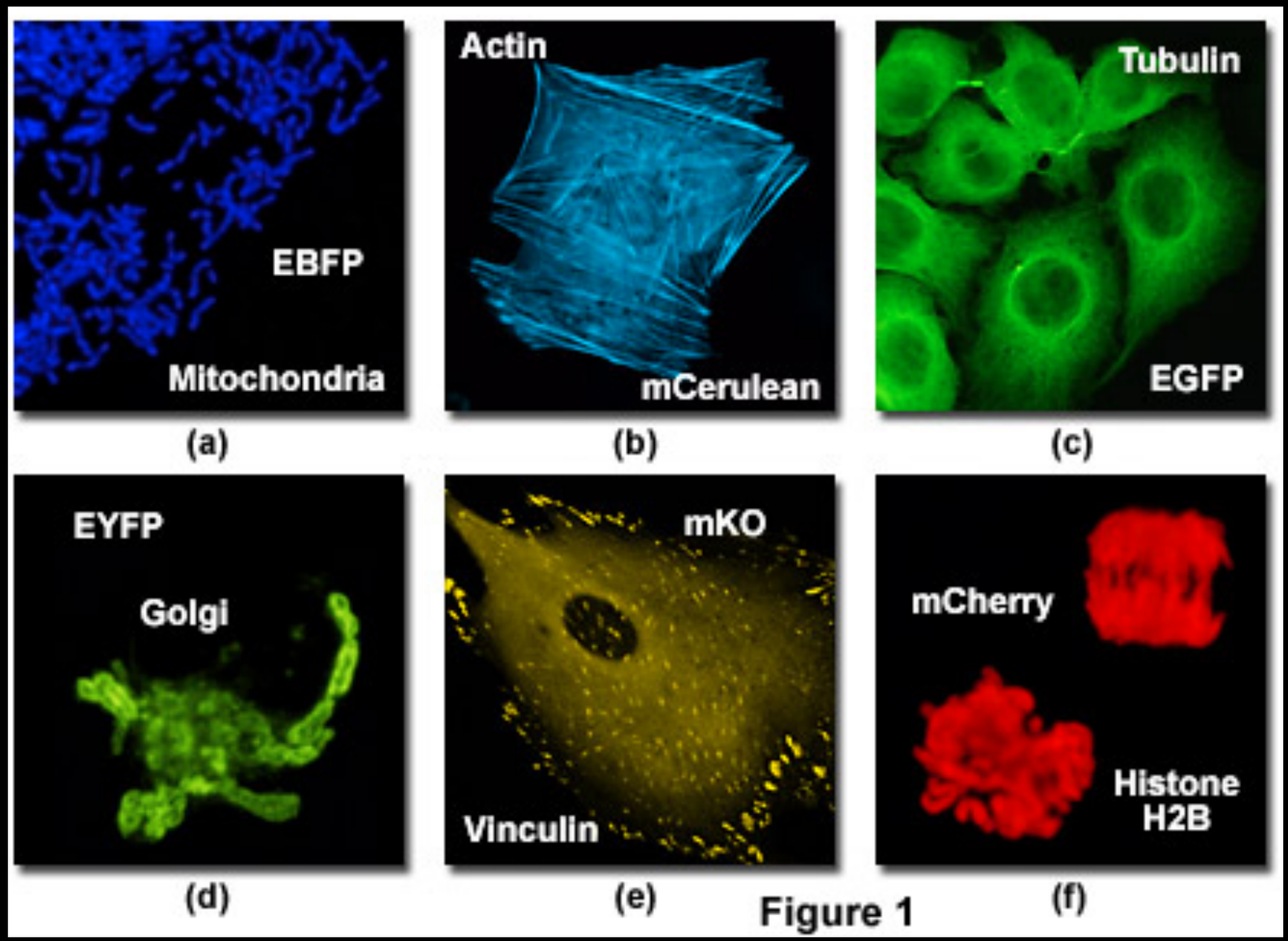
We can use Bayesian optimization when we have multiple models, with different accuracies on different subsystems

Matthias Poloczek

Jialei Wang

Poloczek, Wang & F., NIPS 2017

We can use Bayesian optimization when searching over combinatorial spaces



Examples of fluorescent protein tags (not from our collaboration)

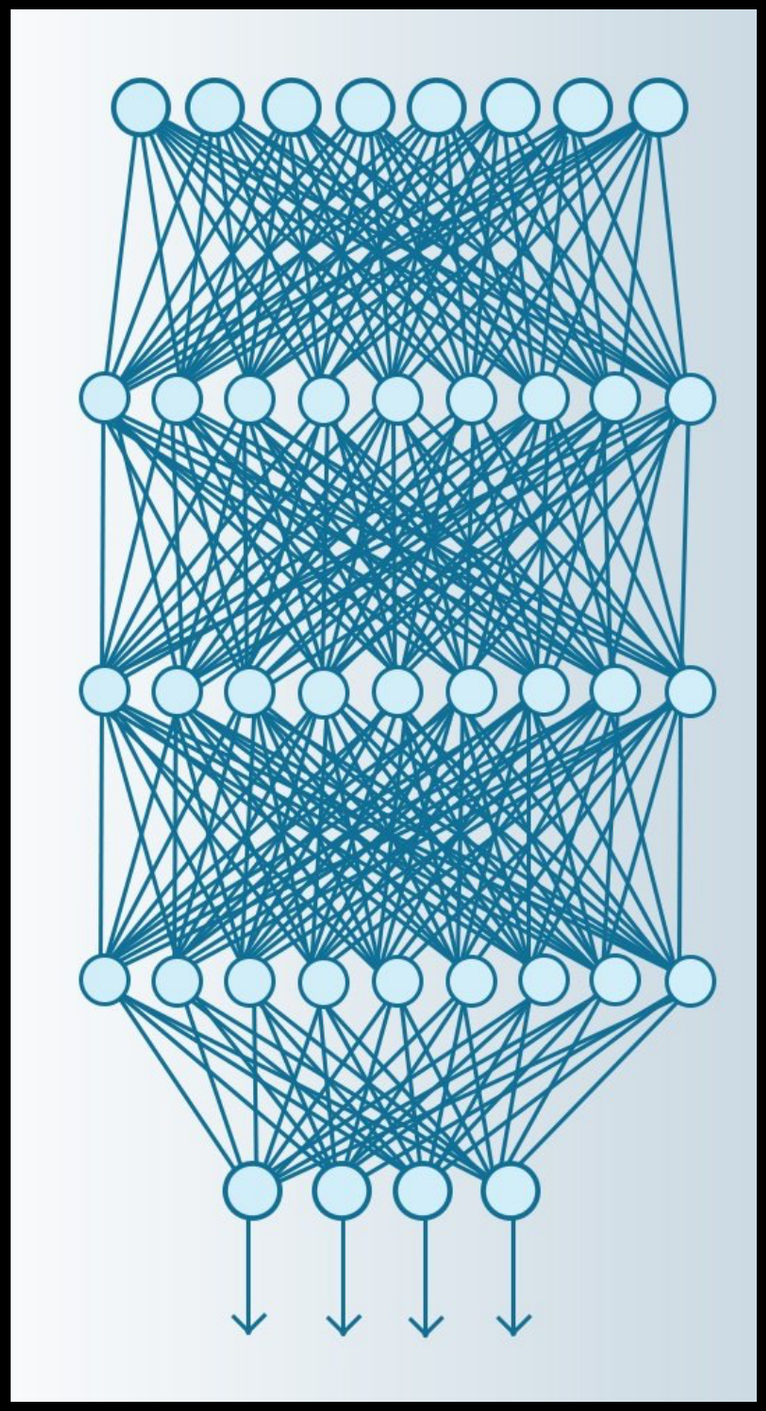
Jialei Wang

Michael Burkart

Nathan Gianneschi

in submission

We can use Bayesian optimization when optimizing integrals & sums



$$\min_{x \in A \subset \mathbb{R}^d} \int F(x, w)\, p(dw)$$

$$\max_{x \in A \subset \mathbb{R}^d} \frac{1}{n} \sum F(x, w)$$

Saul Toscano-Palmerin

Bayesian Optimization with Expensive Integrands
https://arxiv.org/abs/1803.08661, in submission