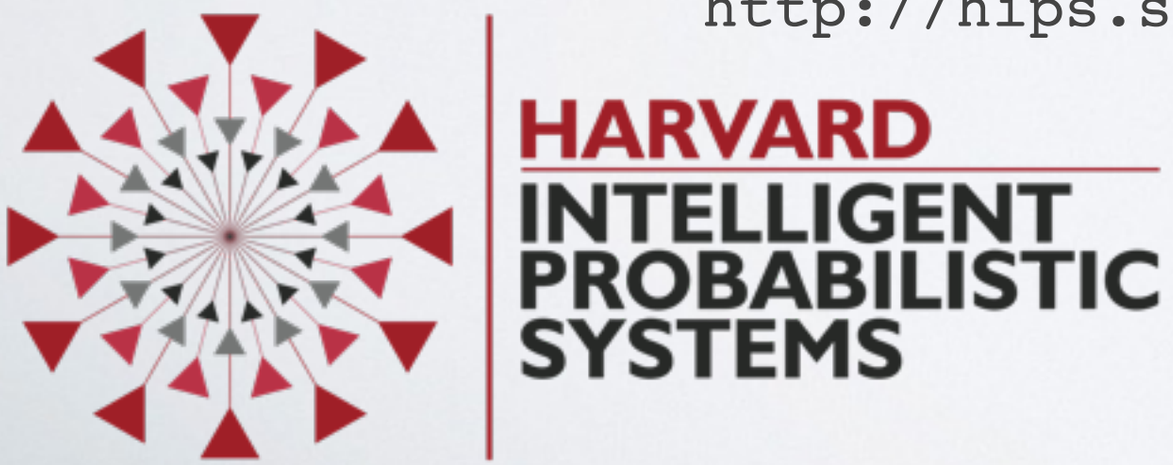
A Tutorial on Bayesian Optimization for Machine Learning

Ryan P. Adams
School of Engineering and Applied Sciences
Harvard University

http://hips.seas.harvard.edu



Machine Learning Meta-Challenges

- **Increasing Model Complexity**
  More flexible models have more parameters.

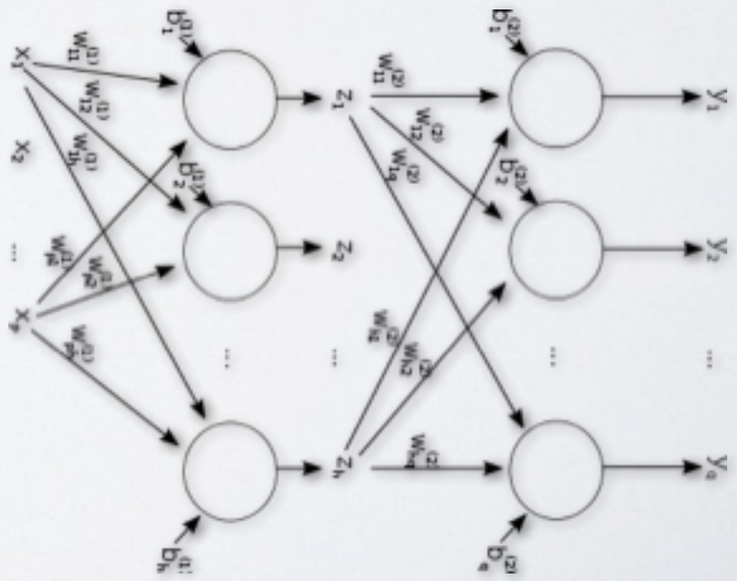
- **More Sophisticated Fitting Procedures**
  Non-convex optimization has many knobs to turn.

- **Less Accessible to Non-Experts**
  Harder to apply complicated techniques.

- **Results Are Less Reproducible**
  Too many important implementation details are missing.

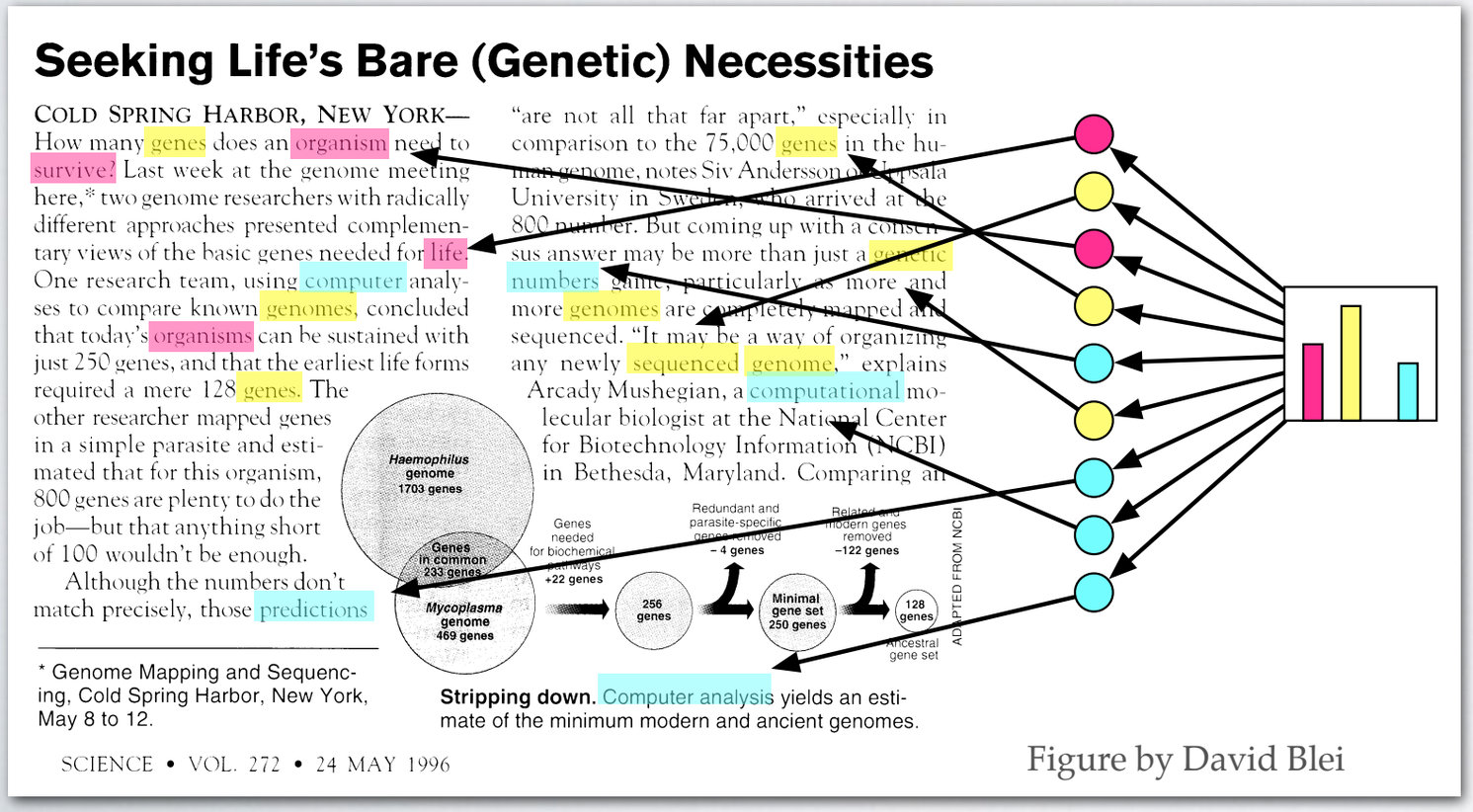
Example: Deep Neural Networks

- Resurgent interest in large neural networks.
- When well-tuned, very successful for visual object identification, speech recognition, comp bio, ...
- Big investments by Google, Facebook, Microsoft, etc.
- Many choices: number of layers, weight regularization, layer size, which nonlinearity, batch size, learning rate schedule, stopping conditions



Example: Online Latent Dirichlet Allocation

- Hoffman, Blei, and Bach (2010): Approximate inference for large-scale text analysis with latent Dirichlet allocation.
- When well-tuned, gives good empirical results.
- Many choices: Dirichlet parameters, number of topics, vocabulary size, learning rate schedule, batch size



Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

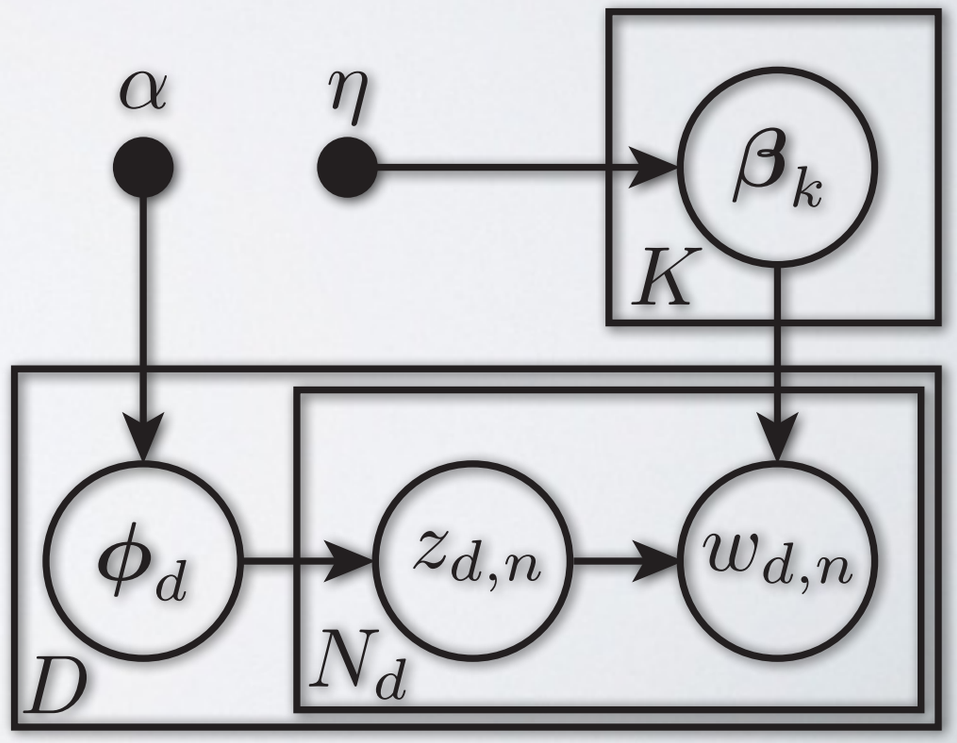
Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

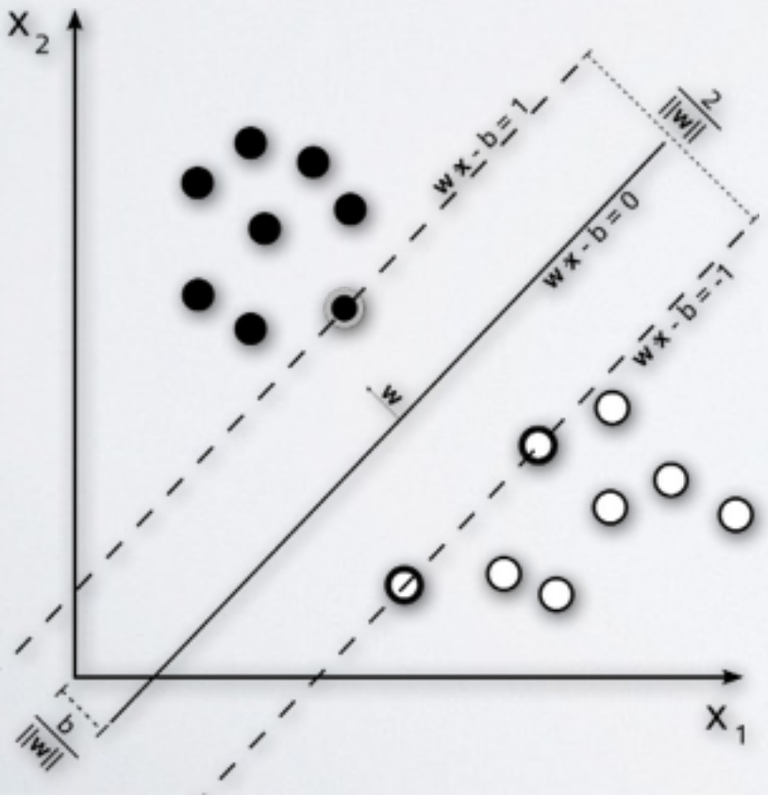
SCIENCE • VOL. 272 • 24 MAY 1996

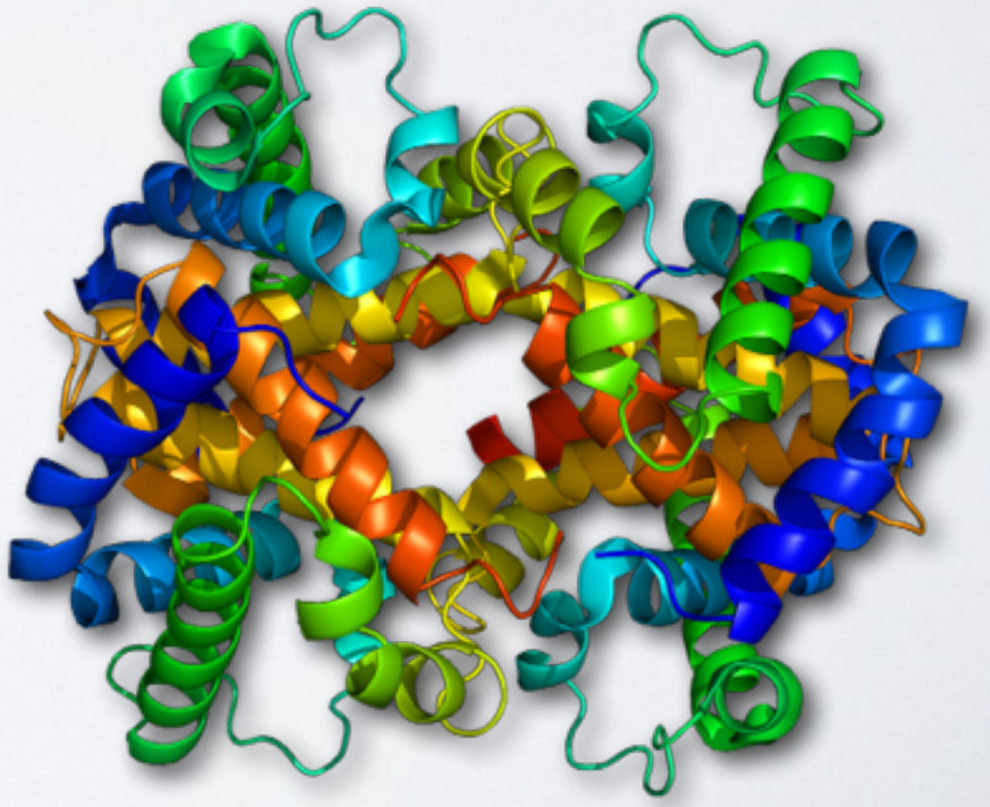
Figure by David Blei



Example: Classification of DNA Sequences

- Objective: predict which DNA sequences will bind with which proteins.
- Miller, Kumar, Packer, Goodman, and Koller (2012): Latent Structural Support Vector Machine
- Choices: margin parameter, entropy parameter, convergence criterion





Search for Good Hyperparameters?

- **Define an objective function.**
  Most often, we care about generalization performance.
  Use cross validation to measure parameter quality.

- **How do people currently search? Black magic.**
  Grid search
  Random search
  Grad student descent

- **Painful!**
  Requires many training cycles.
  Possibly noisy.

Can We Do Better? Bayesian Optimization

- **Build a probabilistic model for the objective.**
  Include hierarchical structure about units, etc.

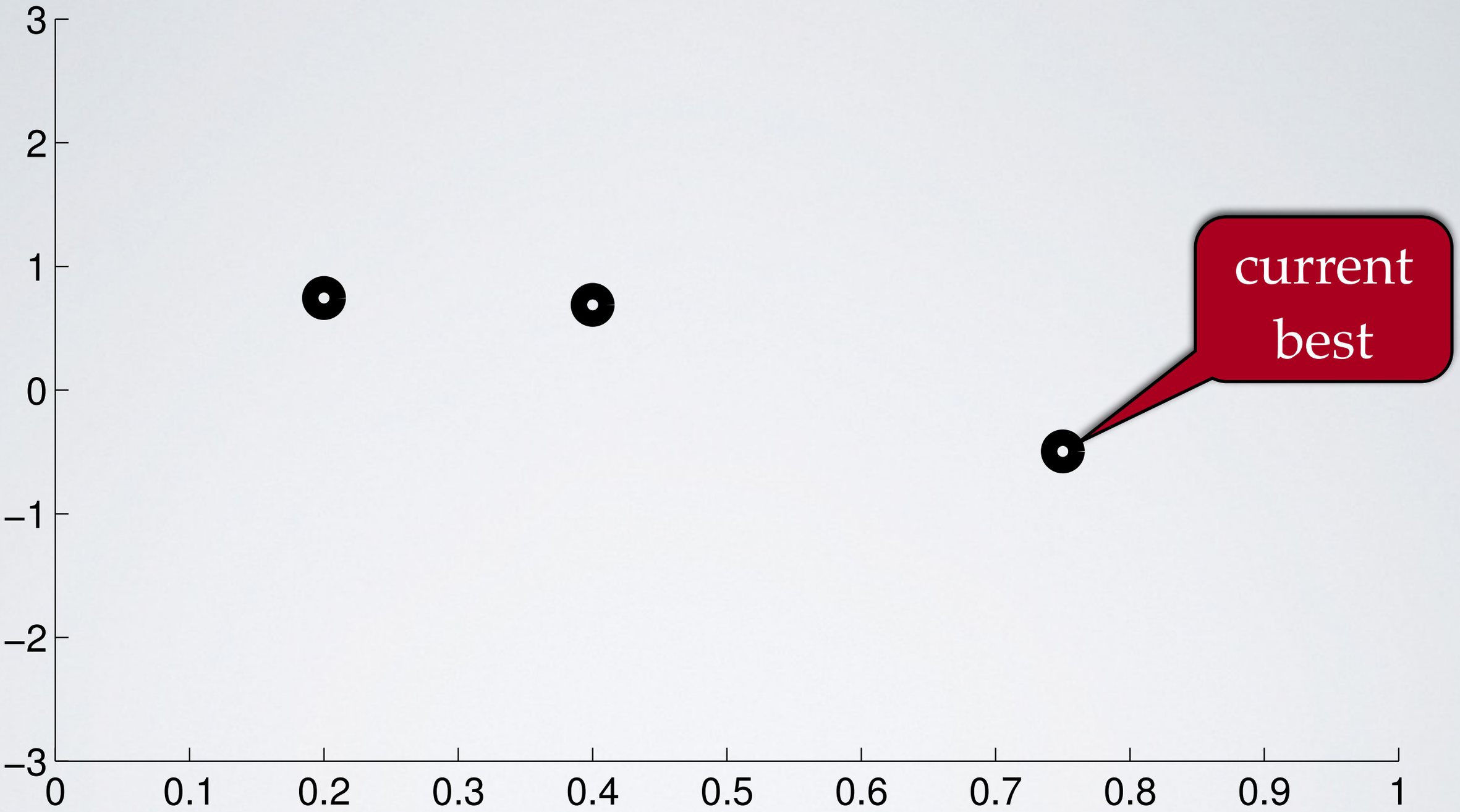
- **Compute the posterior predictive distribution.**
  Integrate out all the possible true functions.
  We use Gaussian process regression.

- **Optimize a cheap proxy function instead.**
  The model is much cheaper than that true objective.

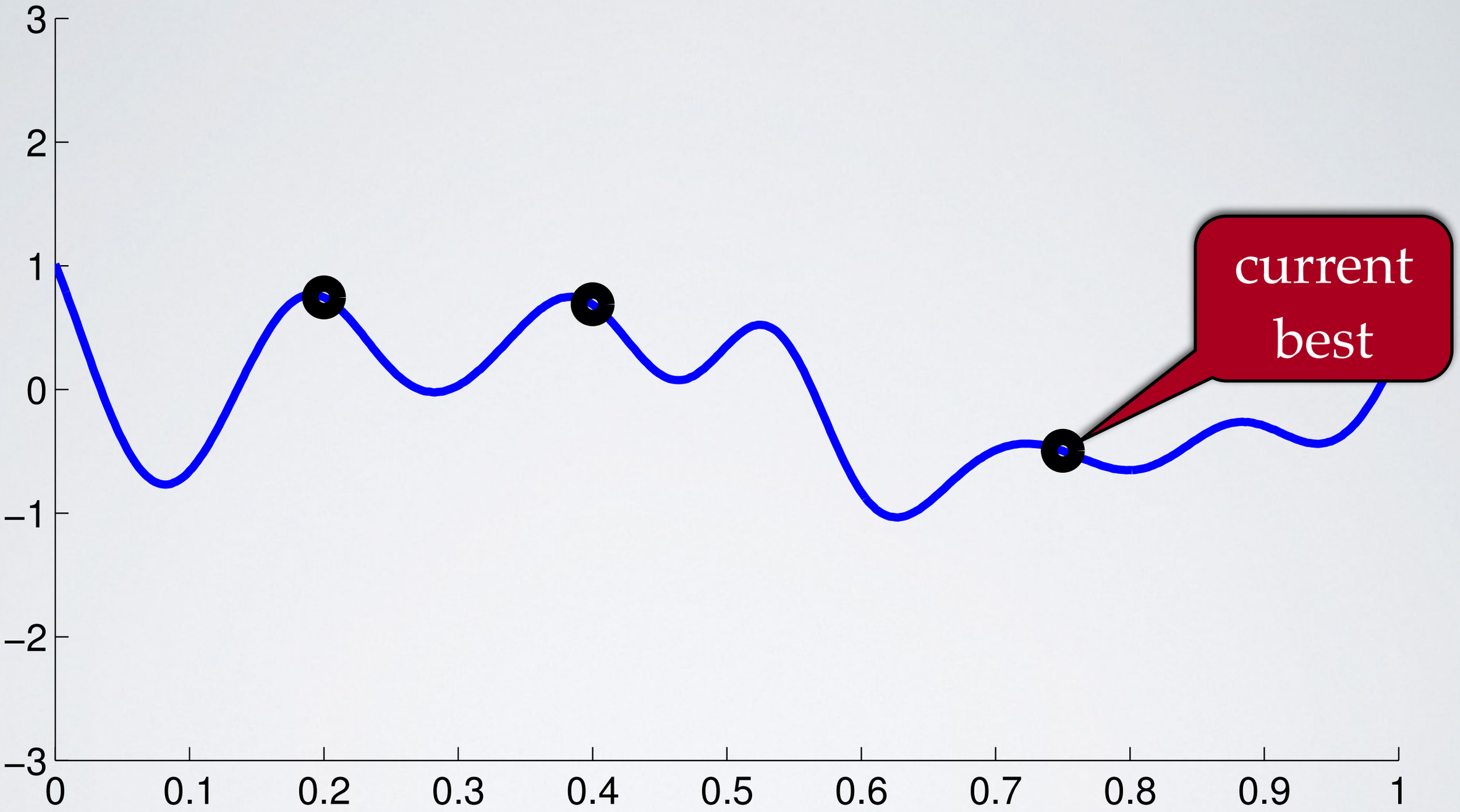
The main insight:

Make the proxy function exploit uncertainty to balance **exploration** against **exploitation**.

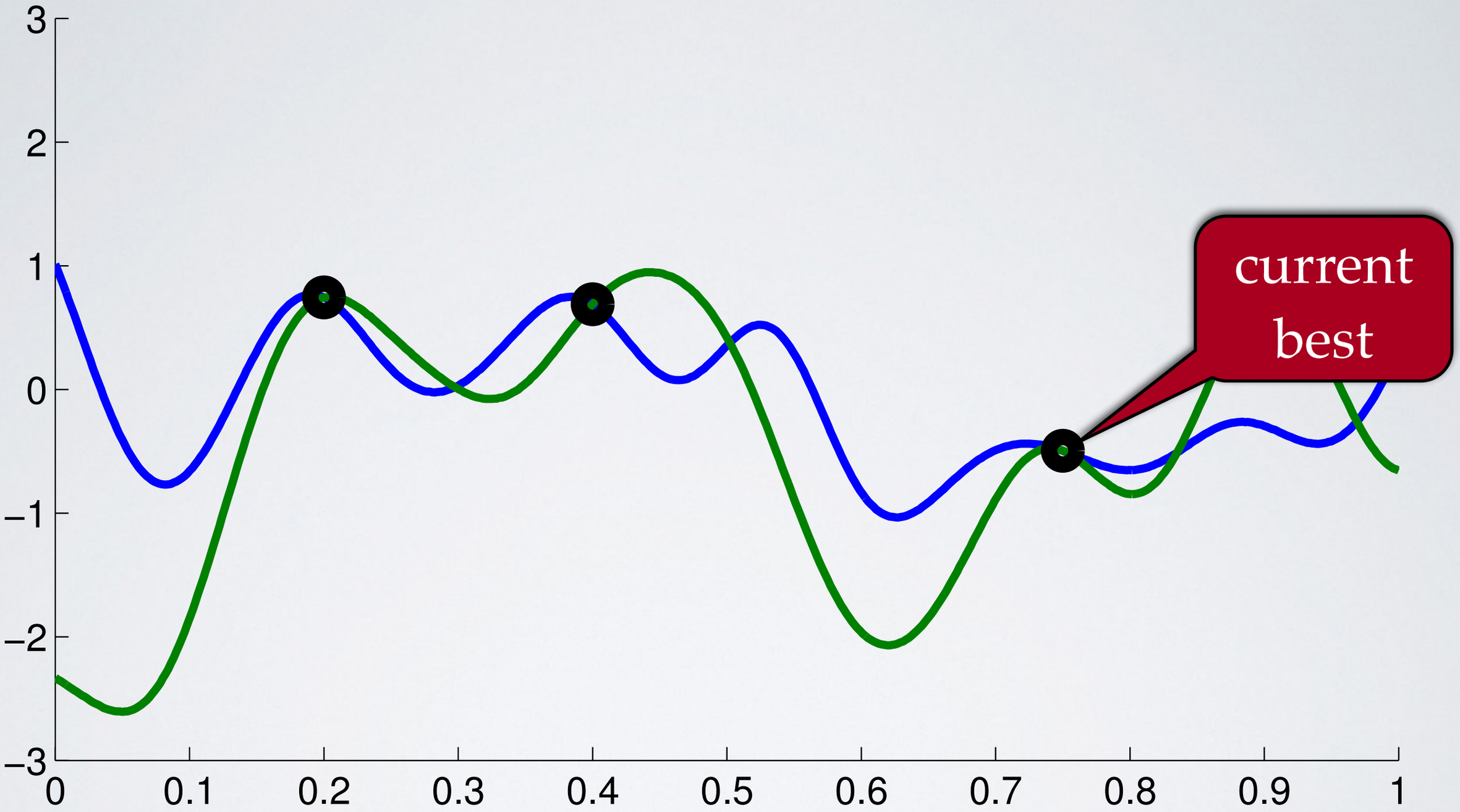
The Bayesian Optimization Idea



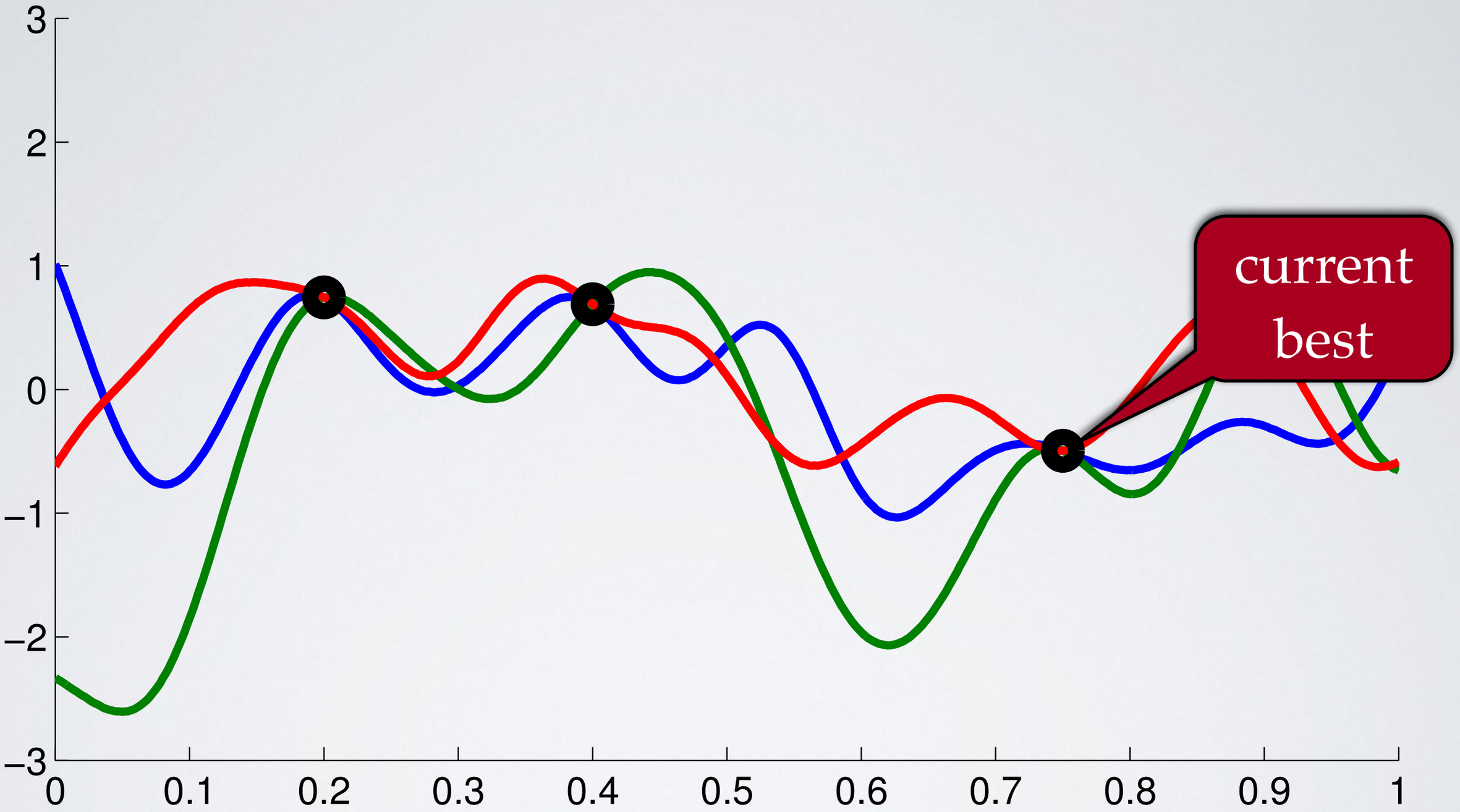
The Bayesian Optimization Idea



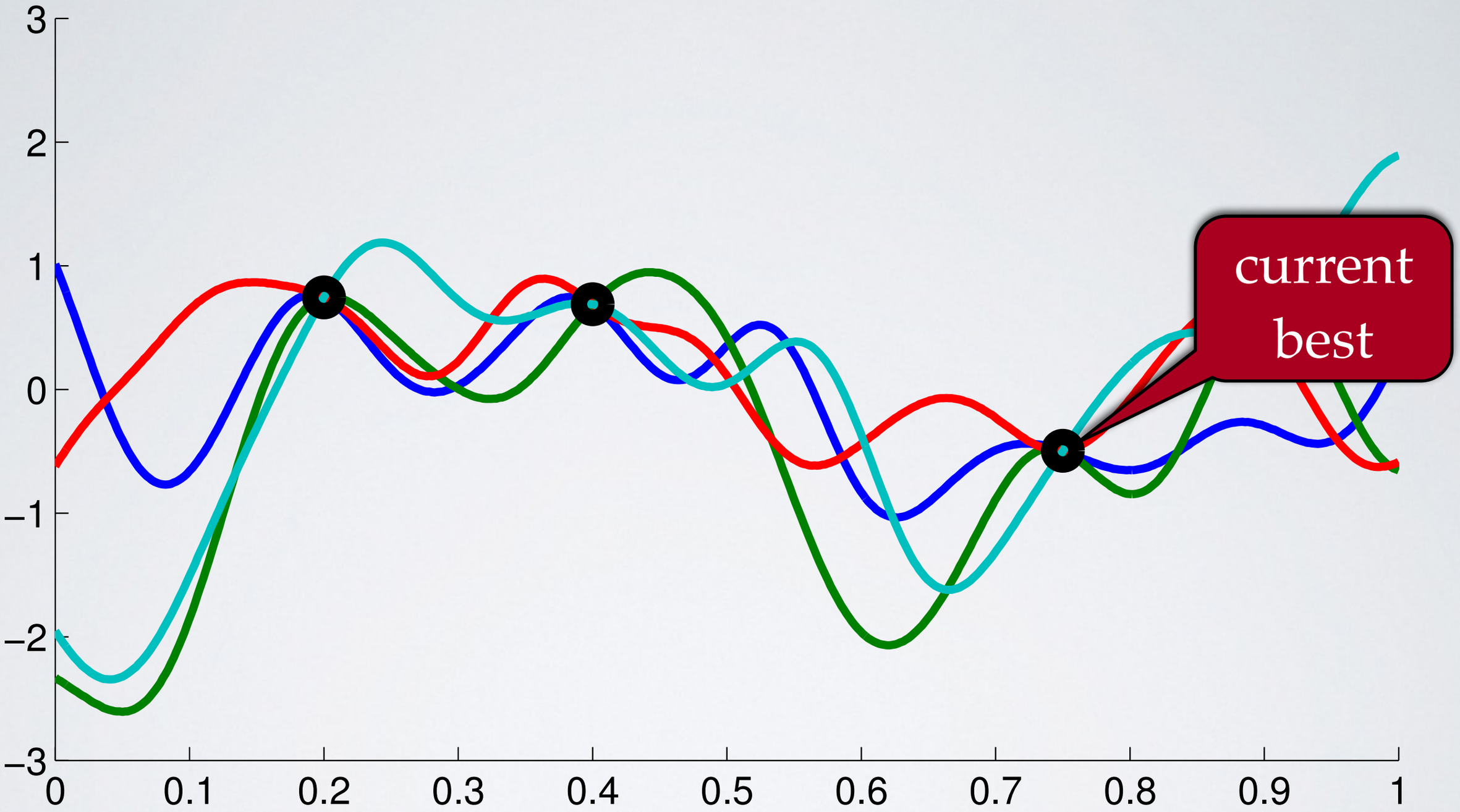
The Bayesian Optimization Idea



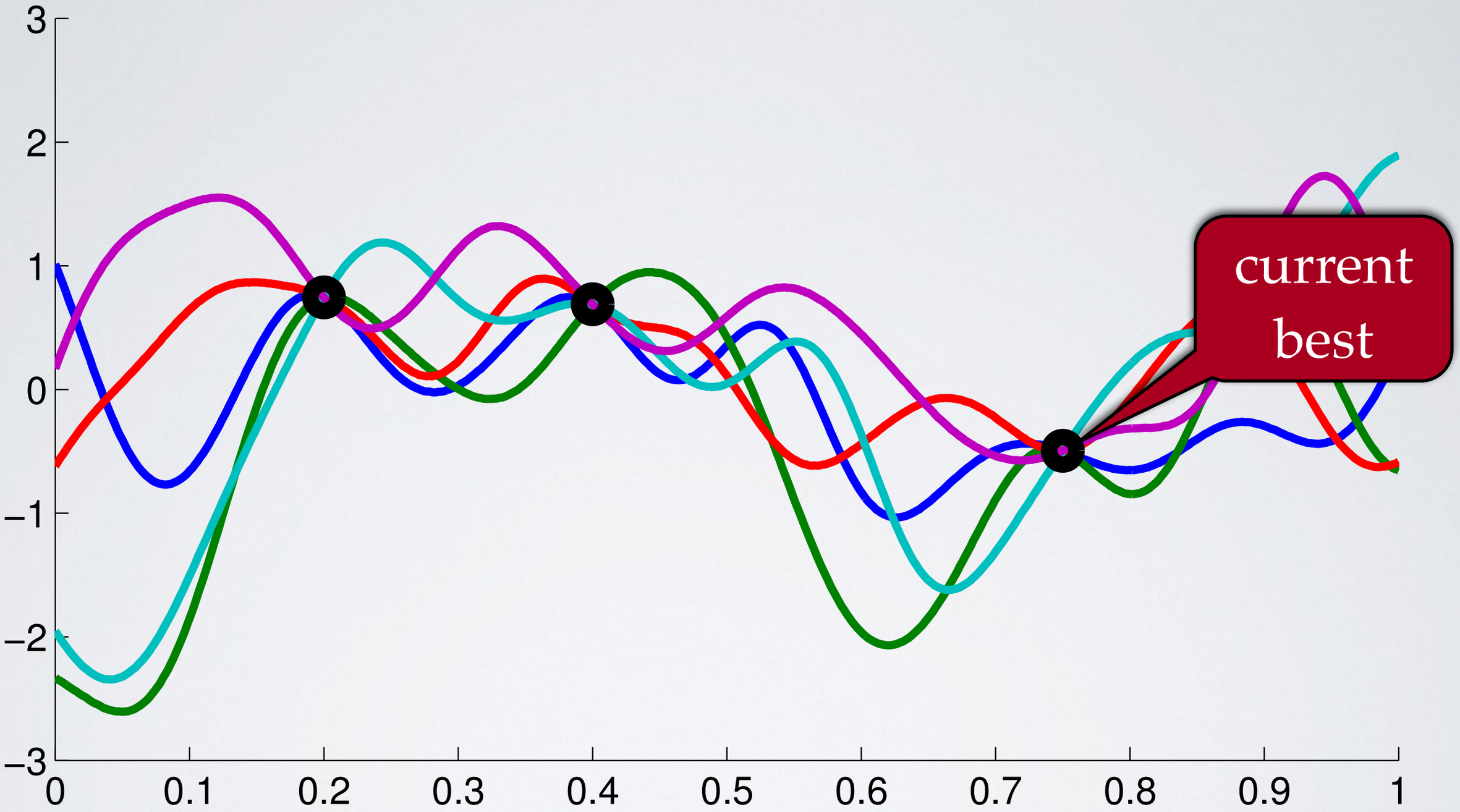
The Bayesian Optimization Idea



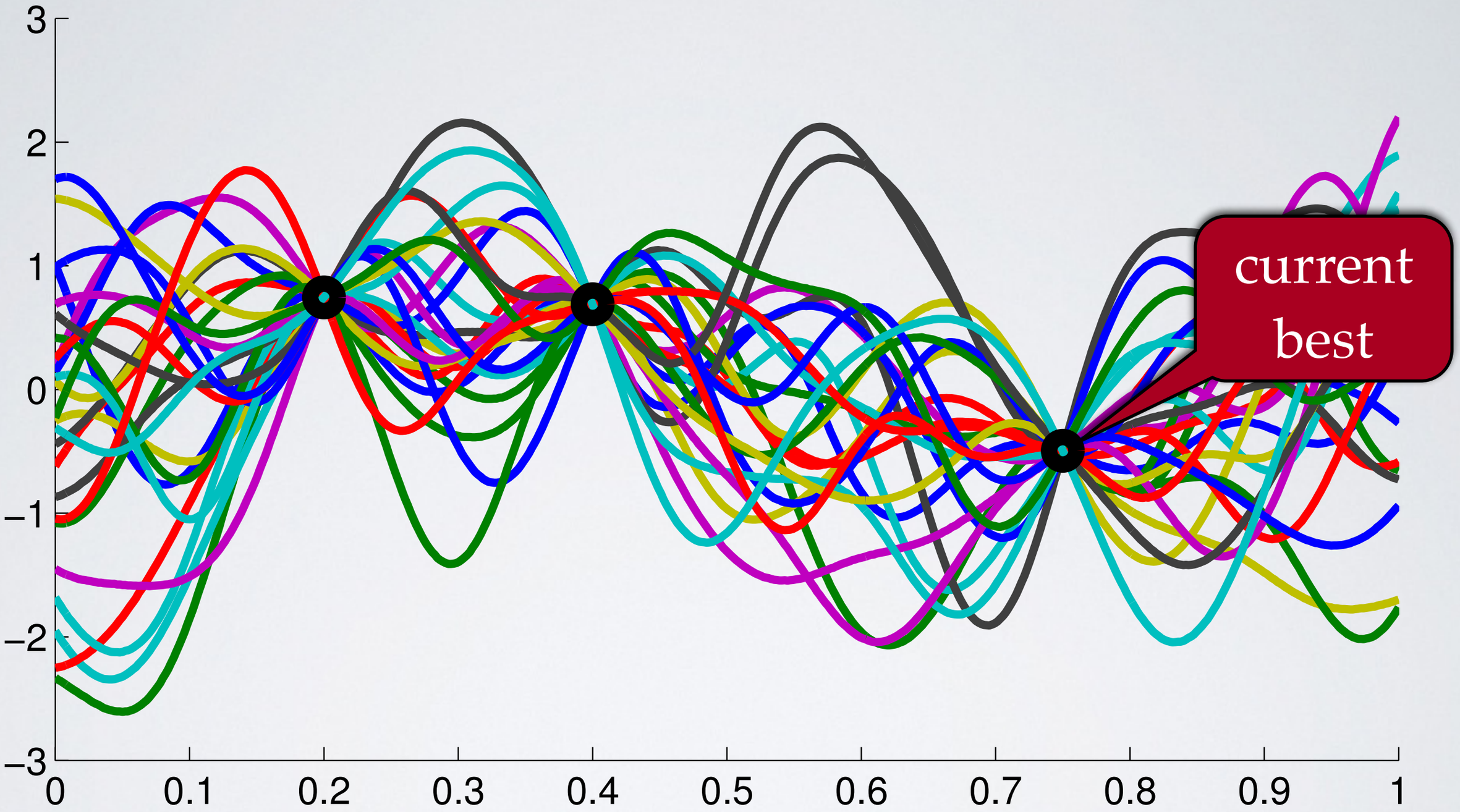
The Bayesian Optimization Idea



The Bayesian Optimization Idea

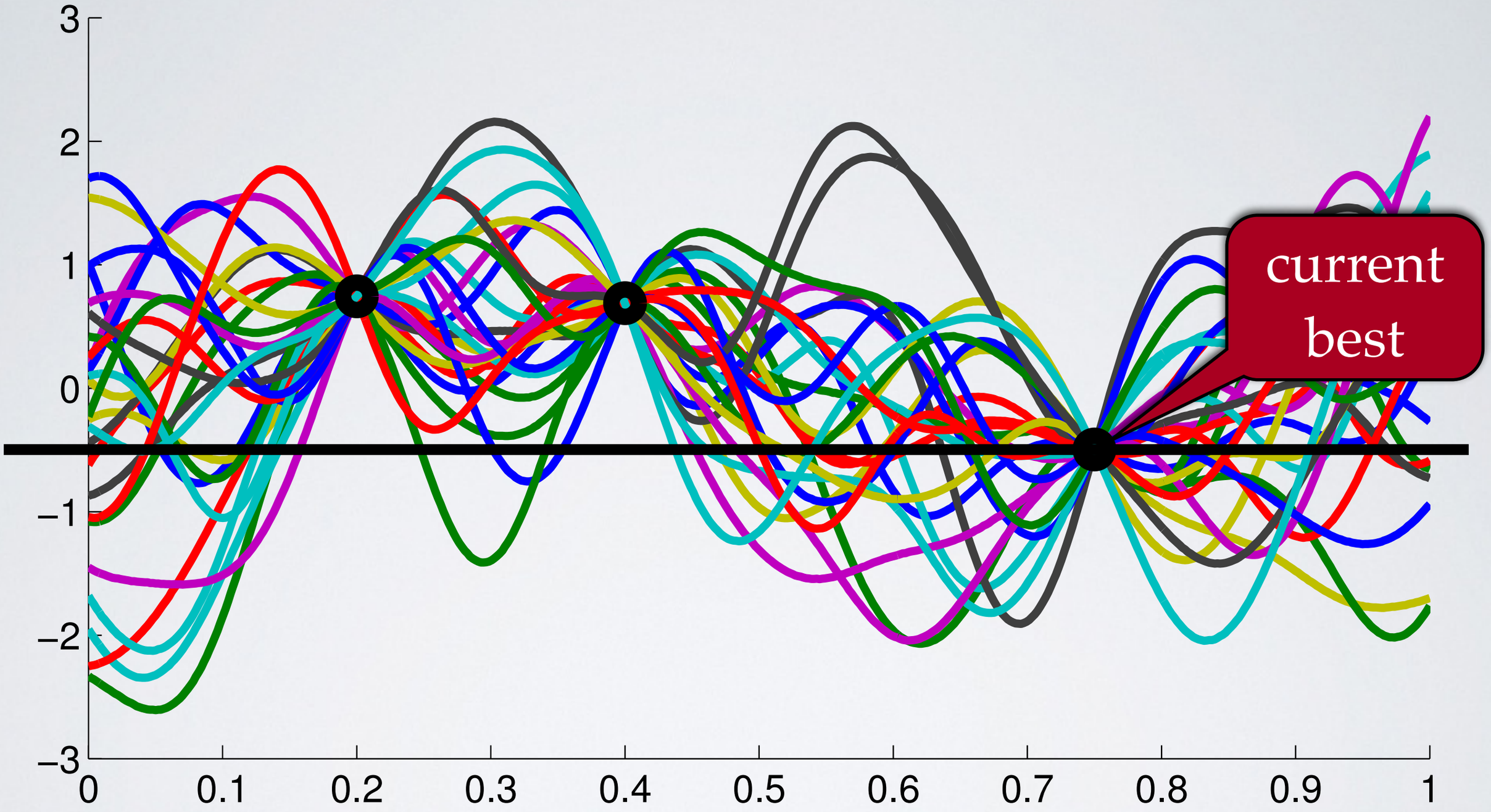


The Bayesian Optimization Idea



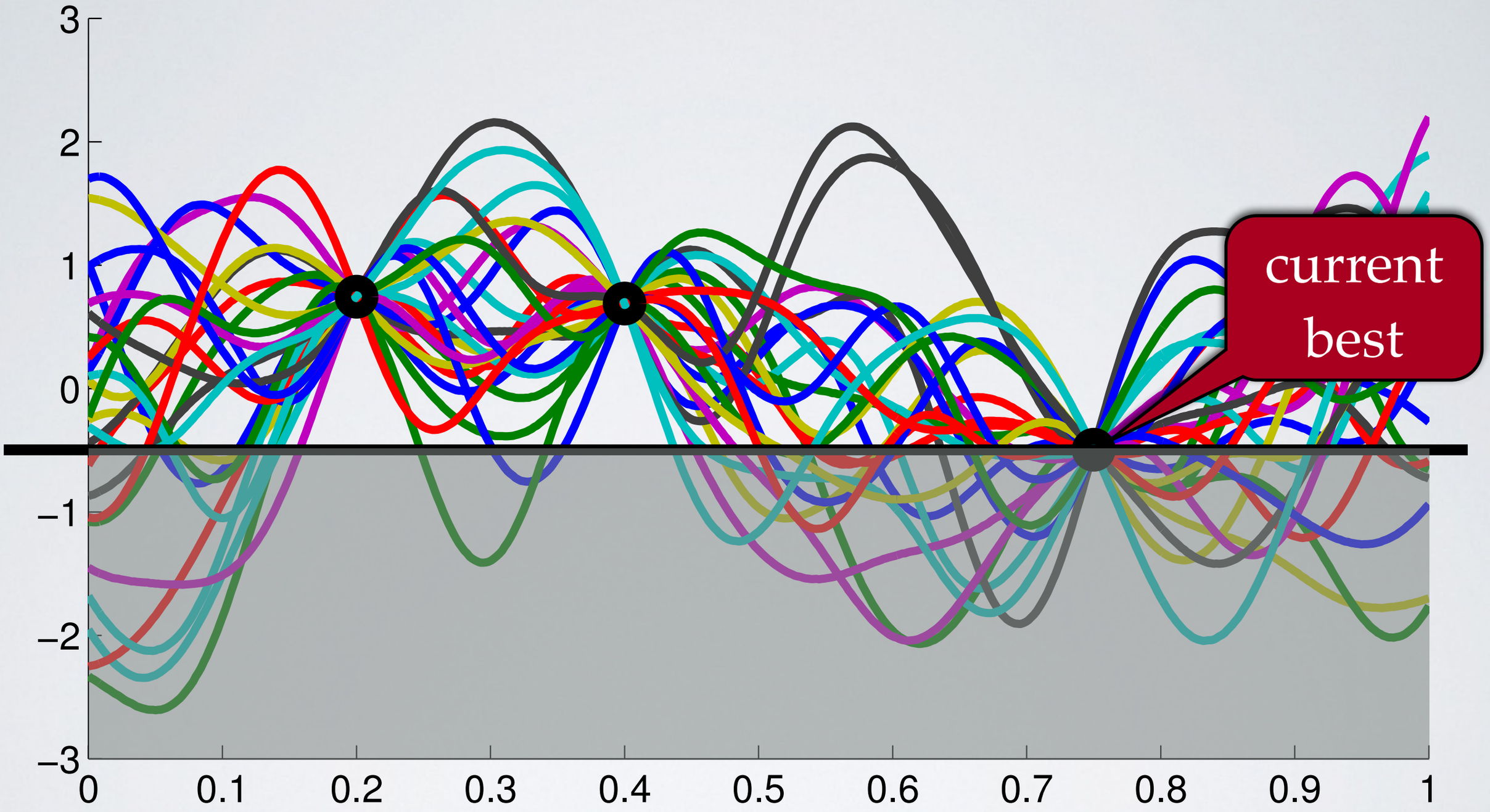
Where should we evaluate next
in order to improve the most?

The Bayesian Optimization Idea



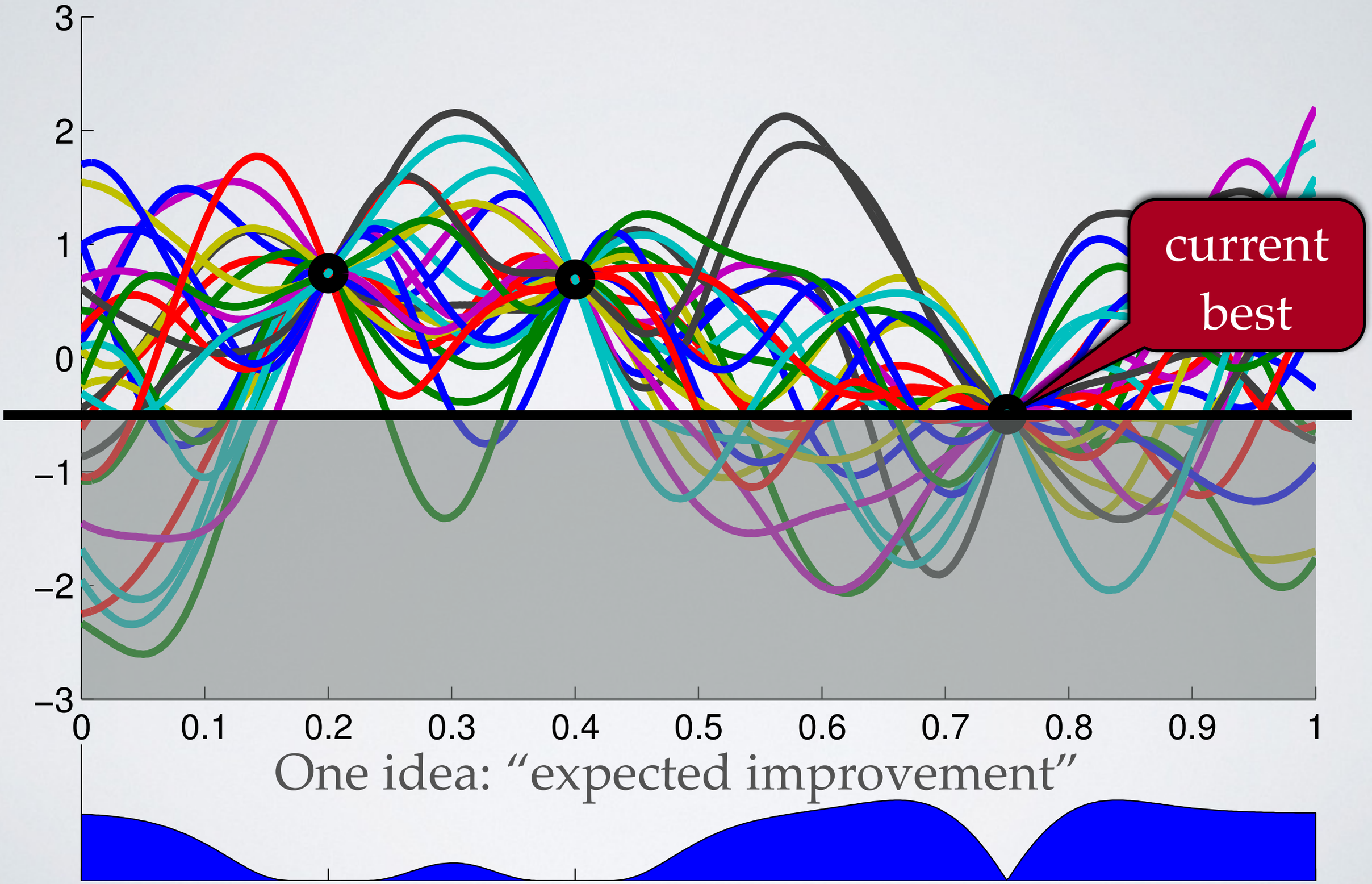
Where should we evaluate next
in order to improve the most?

The Bayesian Optimization Idea



Where should we evaluate next
in order to improve the most?

The Bayesian Optimization Idea



One idea: “expected improvement”

Historical Overview

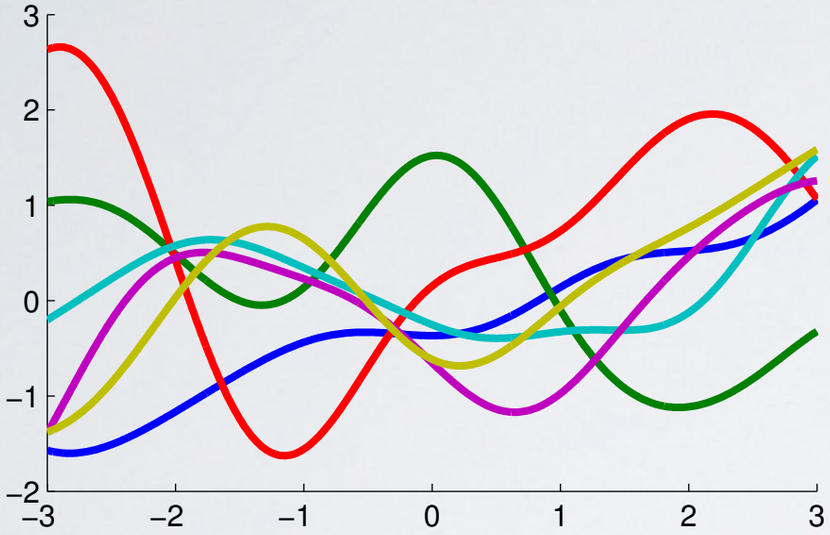
- Closely related to statistical ideas of *optimal design of experiments,* dating back to Kirstine Smith in 1918.
- As *response surface methods,* they date back to Box and Wilson in 1951.
- As *Bayesian optimization,* studied first by Kushner in 1964 and then Mockus in 1978.
- Methodologically, it touches on several important machine learning areas: active learning, contextual bandits, Bayesian nonparametrics
- Started receiving serious attention in ML in 2007, for example:
  Brochu, de Freitas & Ghosh, NIPS 2007 [preference learning]
  Krause, Singh & Guestrin, JMLR 2008 [optimal sensor placement]
  Srinivas, Krause, Kakade & Seeger, ICML 2010 [regret bounds]
  Brochu, Hoffman & de Freitas, UAI 2011 [portfolios]
- Interest exploded when it was realized that Bayesian optimization provides an excellent tool for finding good ML hyperparameters.

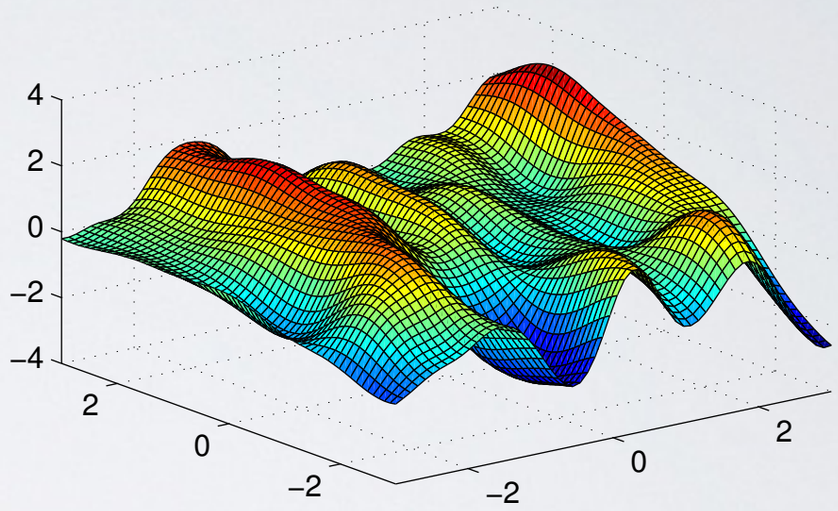
Today's Topics

- Review of Gaussian process priors
- Bayesian optimization basics
- Managing covariances and kernel parameters
- Accounting for the cost of evaluation
- Parallelizing training
- Sharing information across related problems
- Better models for nonstationary functions
- Random projections for high-dimensional problems
- Accounting for constraints
- Leveraging partially-completed training runs

Gaussian Processes as Function Models

Nonparametric prior on functions specified in terms of a positive definite kernel.

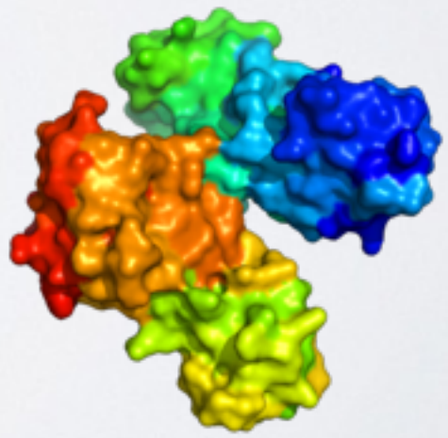





```
the quick brown fox
jumps over the lazy dog
```

strings (Haussler 1999)

permutations (Kondor 2008)



proteins (Borgwardt 2007)

Gaussian Processes

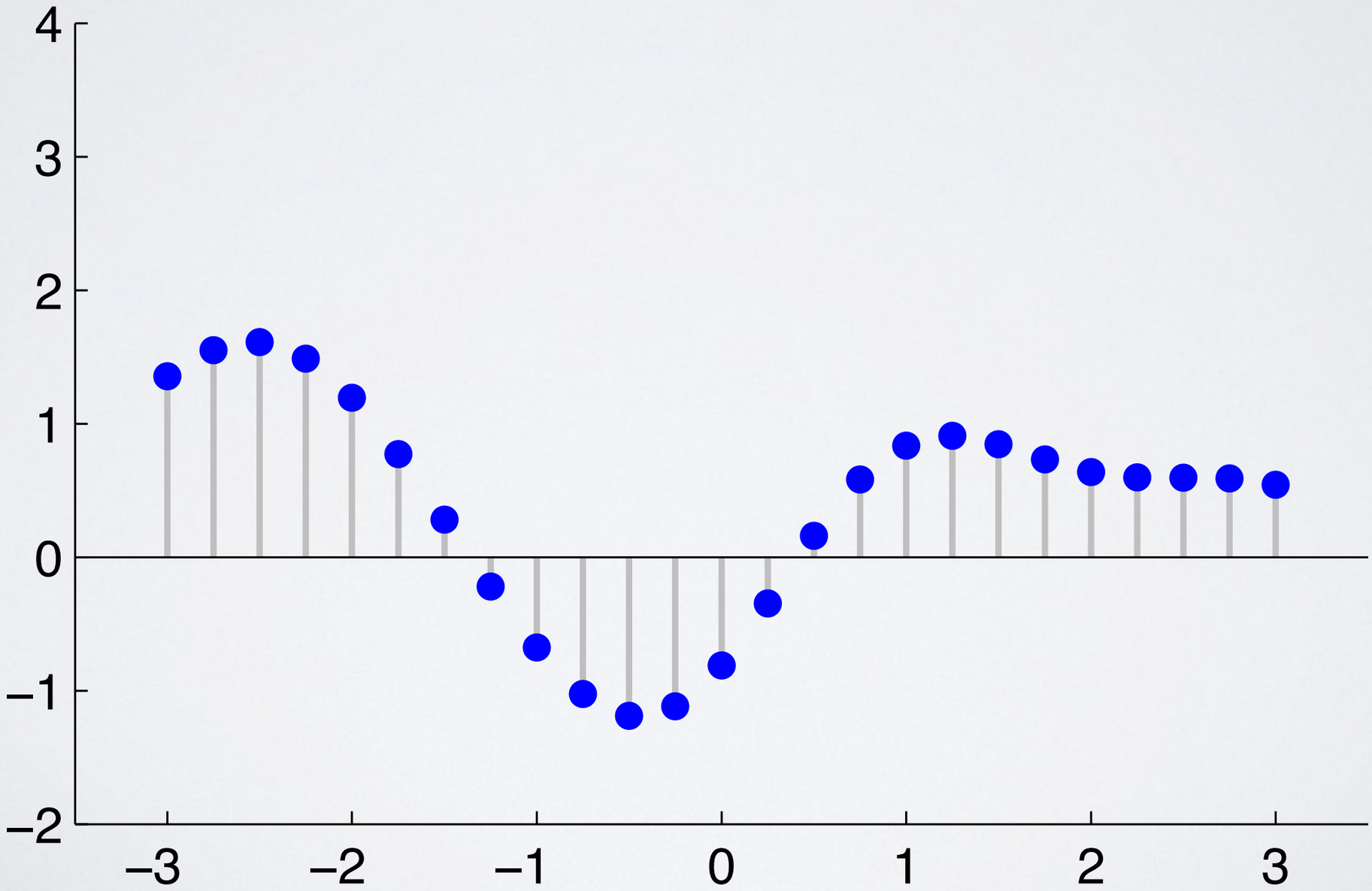
- Gaussian process (GP) is a distribution on functions.
- Allows tractable Bayesian modeling of functions without specifying a particular finite basis.
- Input space (where we're optimizing) $\mathcal{X}$
- Model scalar functions $f : \mathcal{X} \to \mathbb{R}$
- Positive definite covariance function $C : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$
- Mean function $m : \mathcal{X} \to \mathbb{R}$

Gaussian Processes

Any finite set of N points in $\mathcal{X}$, $\{x_n\}_{n=1}^N$ induces a homologous N-dimensional Gaussian distribution on $\mathbb{R}^N$, taken to be the distribution on $\{y_n = f(x_n)\}_{n=1}^N$

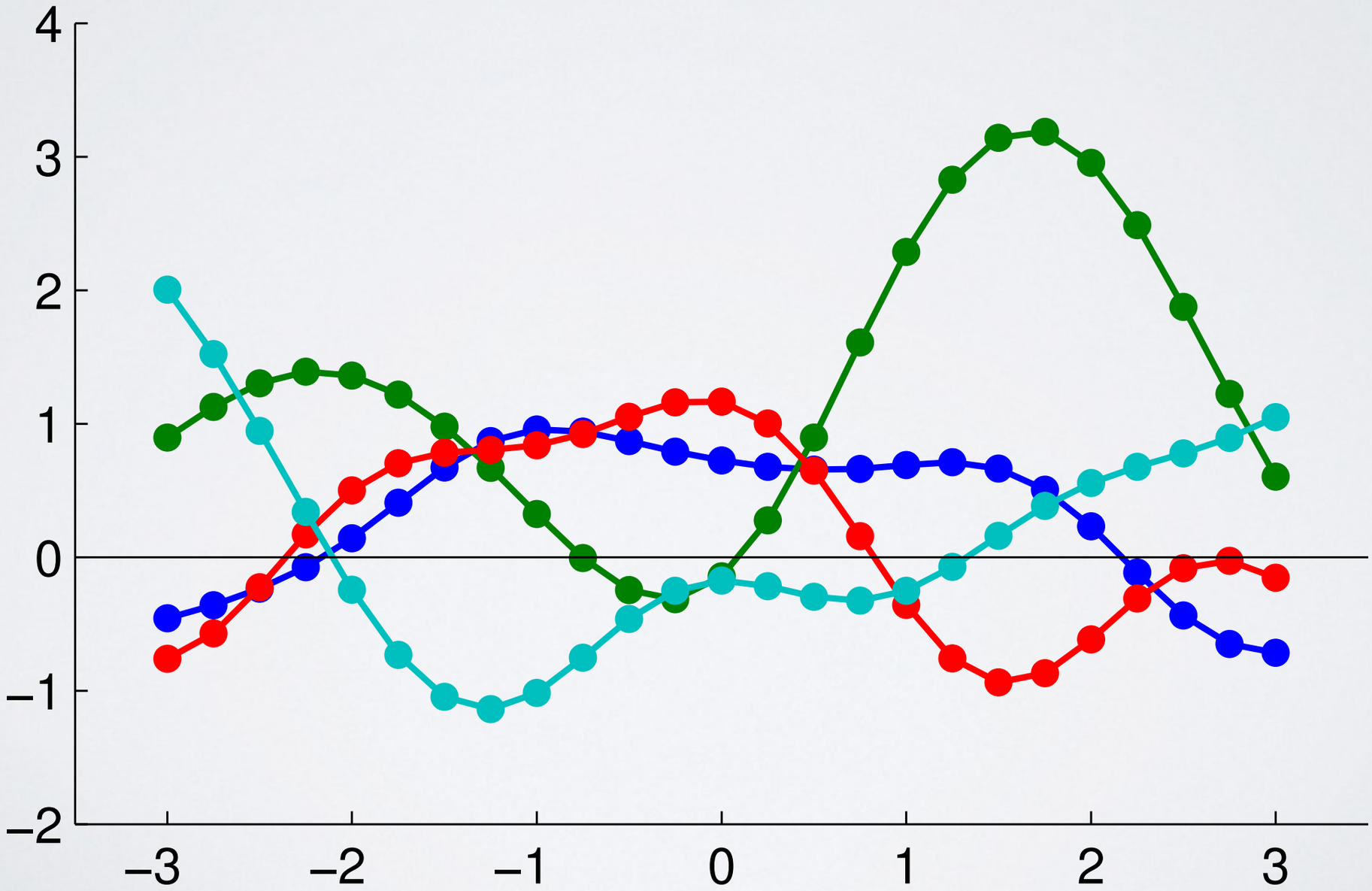
Gaussian Processes

Any finite set of N points in $\mathcal{X}$, $\{x_n\}_{n=1}^N$ induces a homologous N-dimensional Gaussian distribution on $\mathbb{R}^N$, taken to be the distribution on $\{y_n = f(x_n)\}_{n=1}^N$



Gaussian Processes

Any finite set of N points in $\mathcal{X}$, $\{x_n\}_{n=1}^N$ induces a homologous N-dimensional Gaussian distribution on $\mathbb{R}^N$, taken to be the distribution on $\{y_n = f(x_n)\}_{n=1}^N$



Gaussian Processes

- Due to Gaussian form, closed-form solutions for many useful questions about finite data.
- Marginal likelihood:

$$\ln p(\boldsymbol{y} \mid \boldsymbol{X}, \theta) = -\frac{N}{2} \ln 2\pi - \frac{1}{2} \ln |\boldsymbol{K}_\theta| - \frac{1}{2} \boldsymbol{y}^\mathsf{T} \boldsymbol{K}_\theta^{-1} \boldsymbol{y}$$

- Predictive distribution at test points $\{\boldsymbol{x}_m\}_{m=1}^M$:

$$\boldsymbol{y}^{\mathsf{test}} \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{\Sigma})$$

$$\boldsymbol{m} = \boldsymbol{k}_\theta^\mathsf{T} \boldsymbol{K}_\theta^{-1} \boldsymbol{y} \qquad \boldsymbol{\Sigma} = \boldsymbol{\kappa}_\theta - \boldsymbol{k}_\theta^\mathsf{T} \boldsymbol{K}_\theta^{-1} \boldsymbol{k}_\theta$$

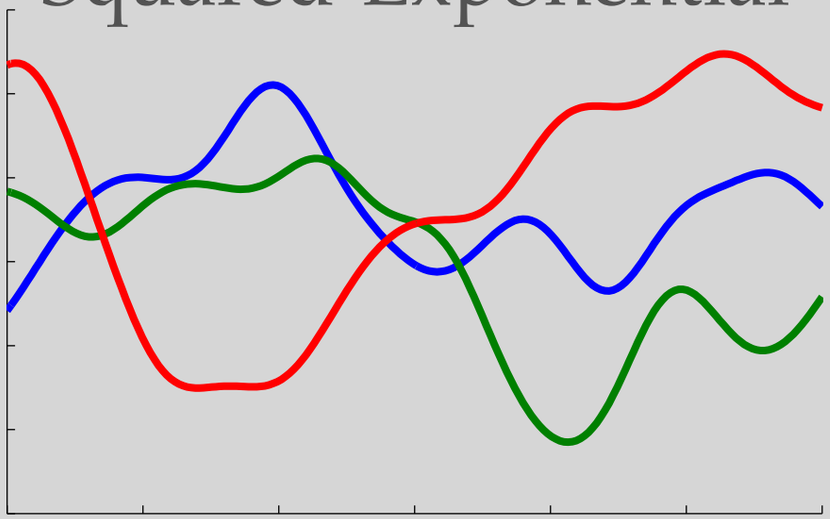
- We compute these matrices from the covariance:

$$[\boldsymbol{K}_\theta]_{n,n'} = C(\boldsymbol{x}_n, \boldsymbol{x}_{n'}\,;\,\theta)$$

$$[\boldsymbol{k}_\theta]_{n,m} = C(\boldsymbol{x}_n, \boldsymbol{x}_m\,;\,\theta) \qquad [\boldsymbol{\kappa}_\theta]_{m,m'} = C(\boldsymbol{x}_m, \boldsymbol{x}_{m'}\,;\,\theta)$$

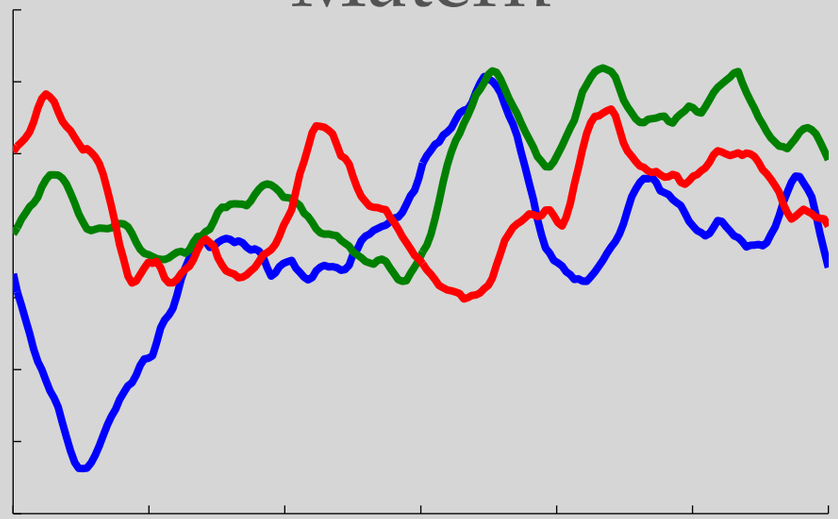
Examples of GP Covariances

Squared-Exponential



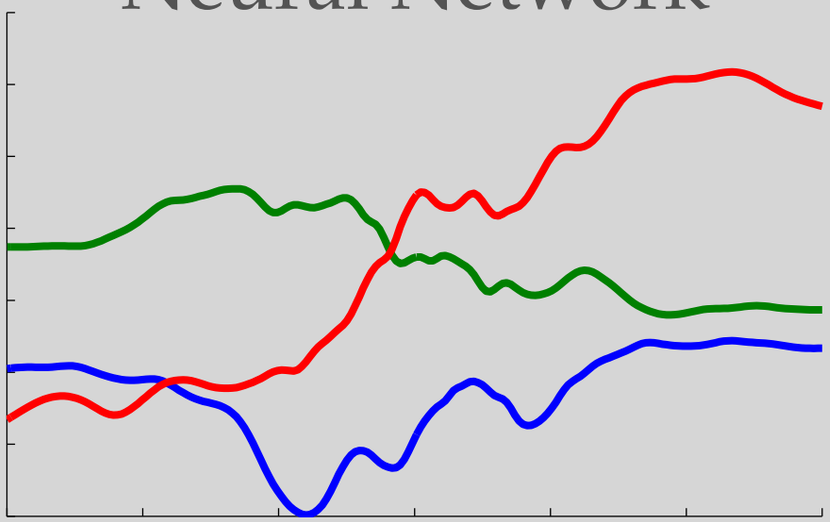
$$C(x, x') = \alpha \exp\left\{-\frac{1}{2}\sum_{d=1}^{D}\left(\frac{x_d - x'_d}{\ell_d}\right)^2\right\}$$

Matérn



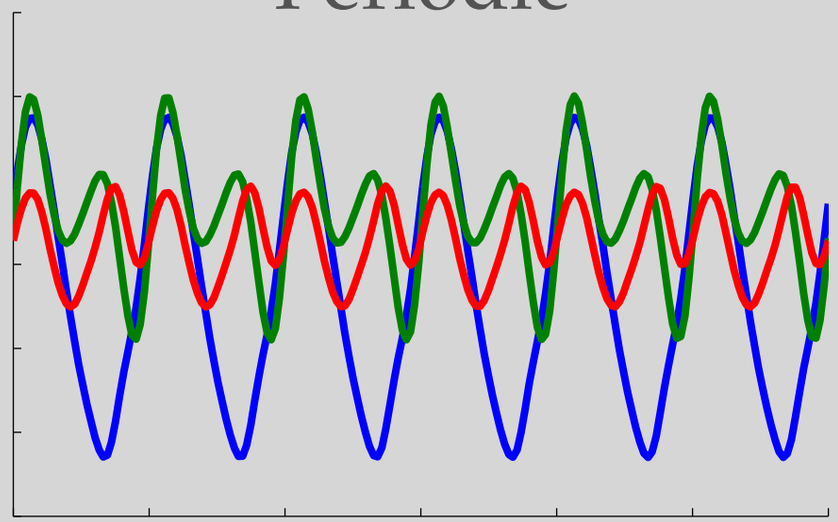
$$C(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}\, r}{\ell}\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}\, r}{\ell}\right)$$

"Neural Network"



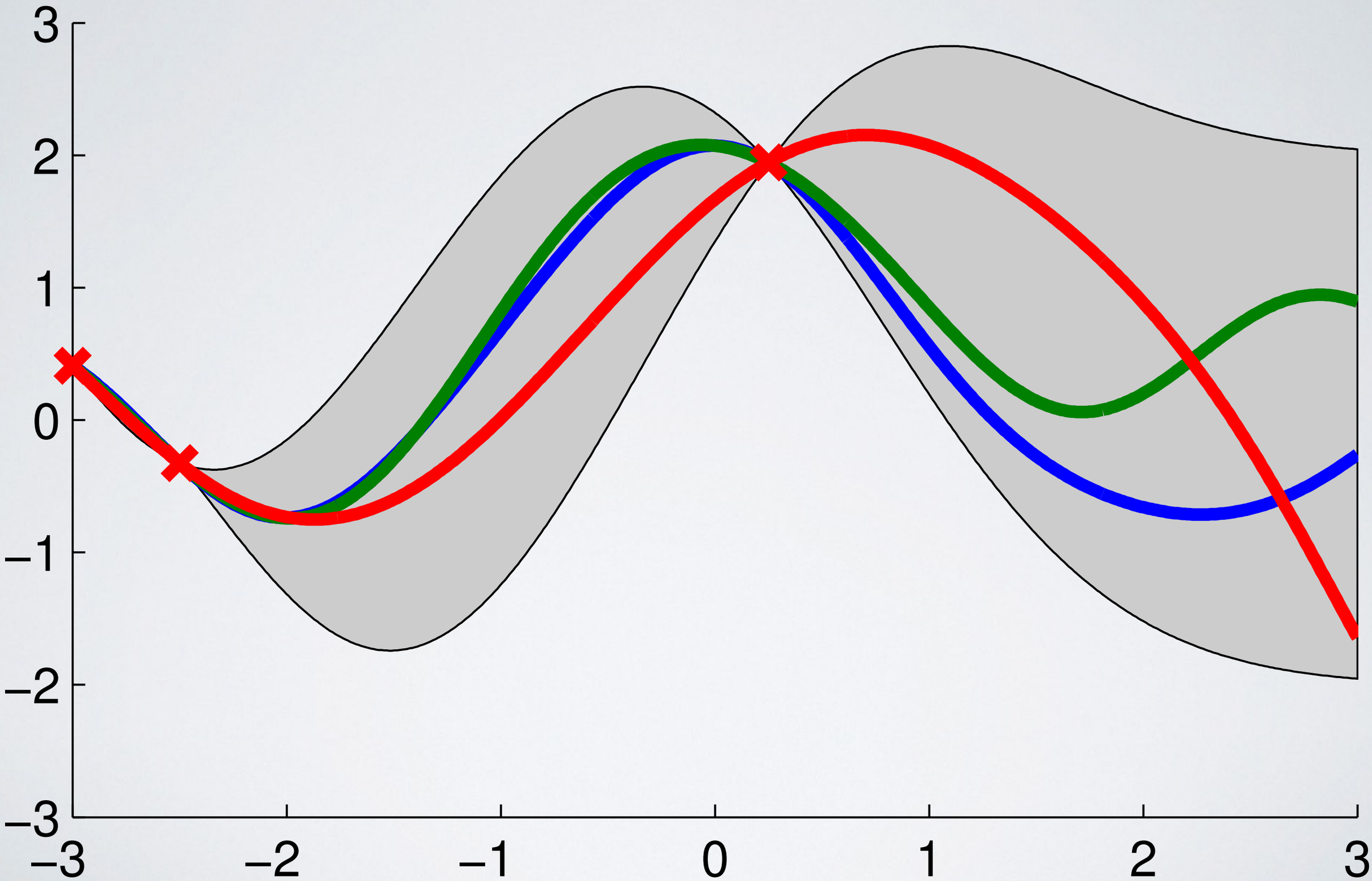
$$C(x, x') = \frac{2}{\pi}\sin^{-1}\left\{\frac{2x^{\mathsf{T}}\Sigma x'}{\sqrt{(1 + 2x^{\mathsf{T}}\Sigma x)(1 + 2x'^{\mathsf{T}}\Sigma x')}}\right\}$$

Periodic



$$C(x, x') = \exp\left\{-\frac{2\sin^2\left(\frac{1}{2}(x - x')\right)}{\ell^2}\right\}$$

GPs Provide Closed-Form Predictions



Using Uncertainty in Optimization

- Find the minimum: $x^\star = \arg\min_{x \in \mathcal{X}} f(x)$
- We can evaluate the objective pointwise, but do not have an easy functional form or gradients.
- After performing some evaluations, the GP gives us easy closed-form marginal means and variances.
- **Exploration**: Seek places with high variance.
- **Exploitation**: Seek places with low mean.
- The **acquisition function** balances these for our proxy optimization to determine the next evaluation.

Closed-Form Acquisition Functions

- The GP posterior gives a predictive mean function $\mu(x)$ and a predictive marginal variance function $\sigma^2(x)$

$$\gamma(x) = \frac{f(x_{\mathsf{best}}) - \mu(x)}{\sigma(x)}$$

- Probability of Improvement (Kushner 1964):

$$a_{\mathsf{PI}}(x) = \Phi(\gamma(x))$$

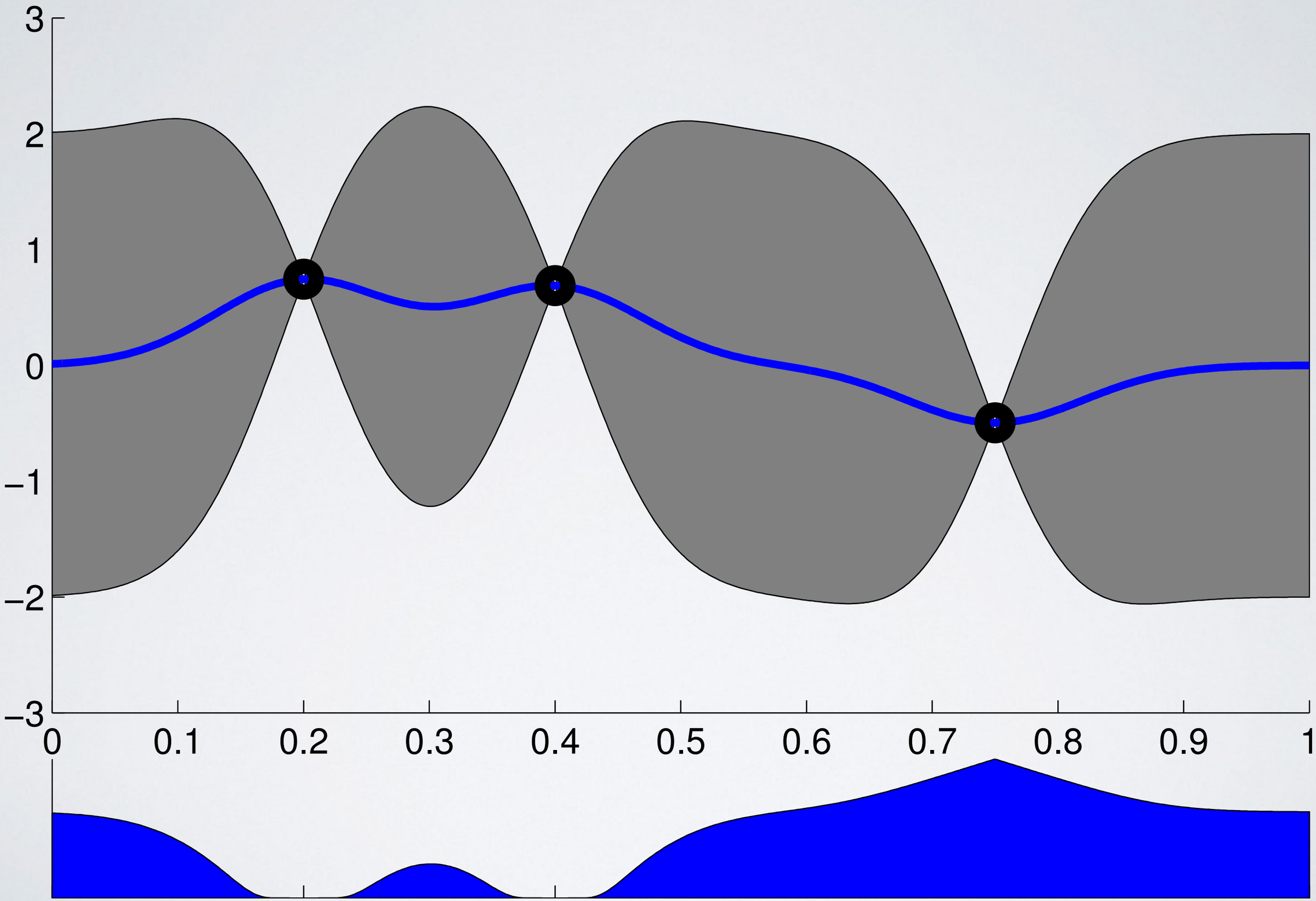
- Expected Improvement (Mockus 1978):

$$a_{\mathsf{EI}}(x) = \sigma(x)(\gamma(x)\Phi(\gamma(x)) + \mathcal{N}(\gamma(x)\,;\,0,1))$$

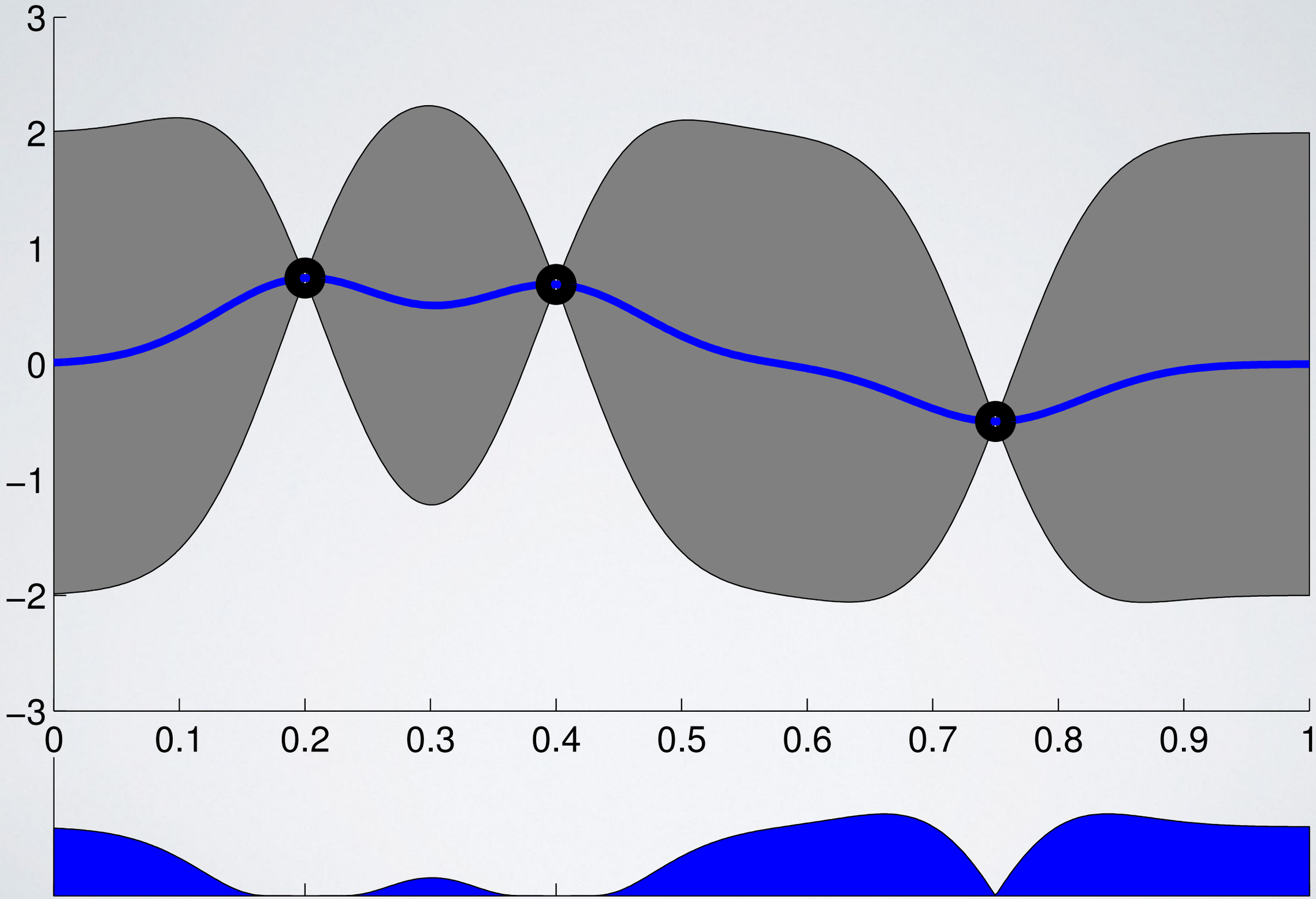
- GP Upper Confidence Bound (Srinivas et al. 2010):

$$a_{\mathsf{LCB}}(x) = \mu(x) - \kappa\,\sigma(x)$$

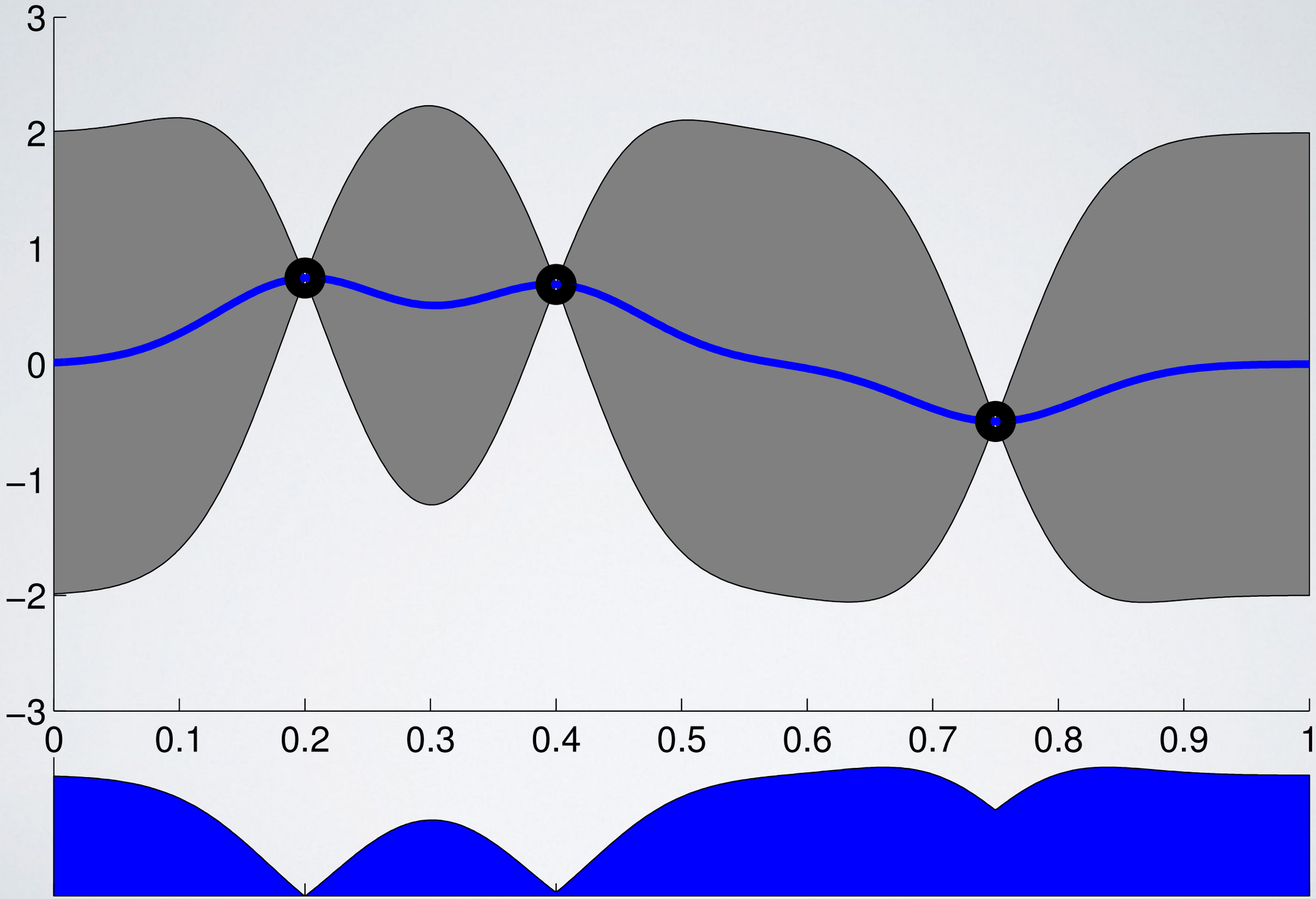
Probability of Improvement



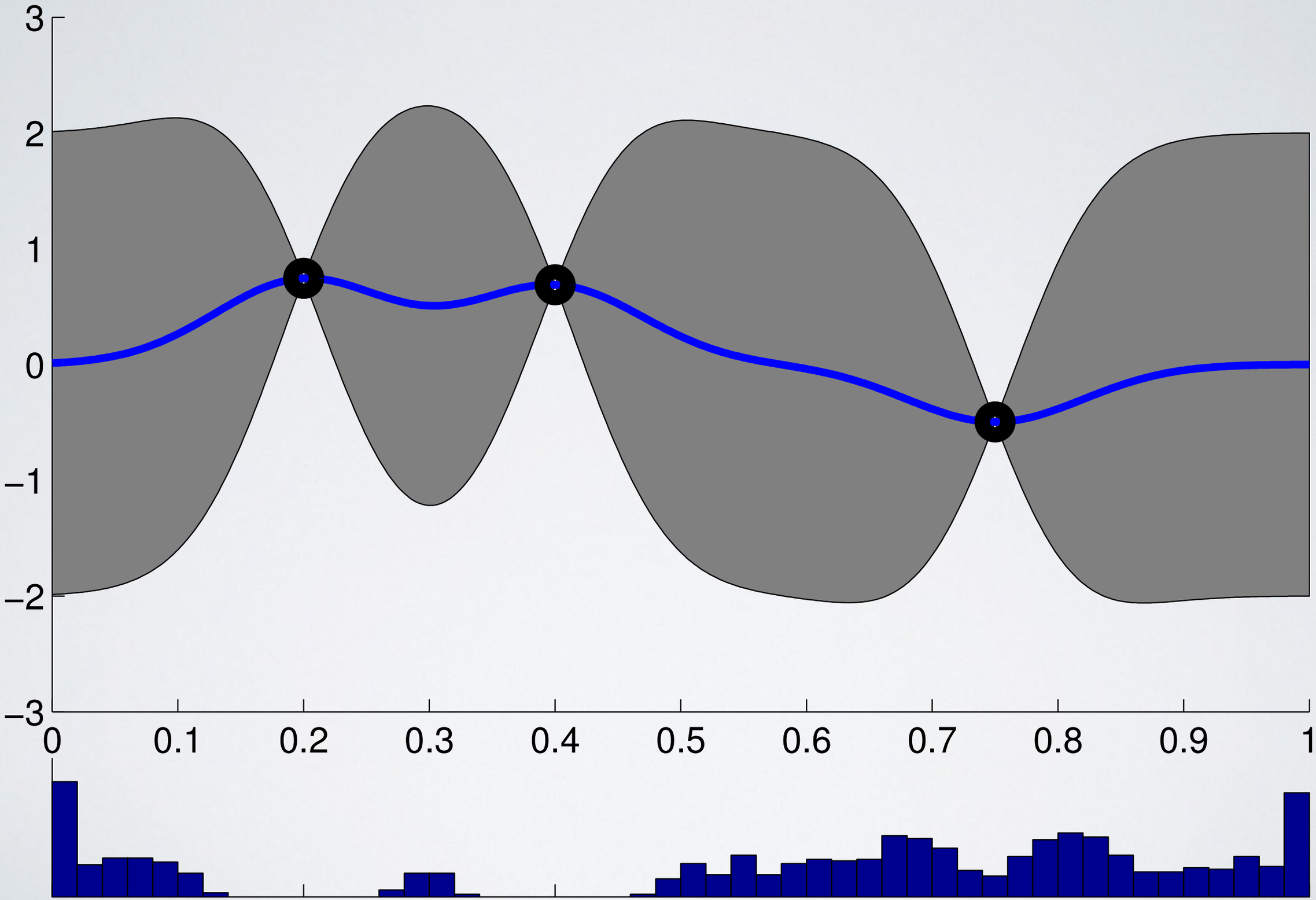
Expected Improvement



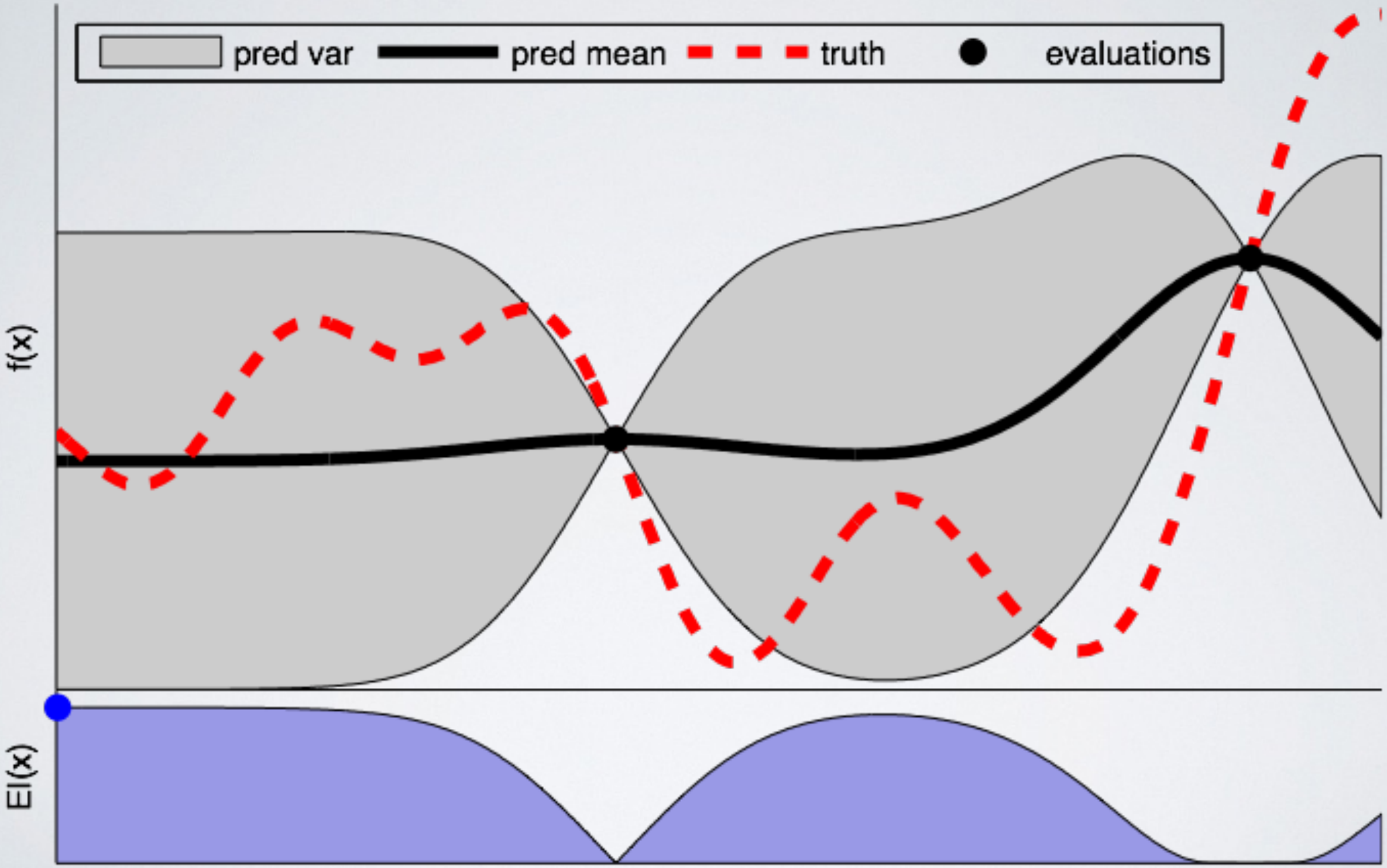
GP Upper (Lower) Confidence Bound



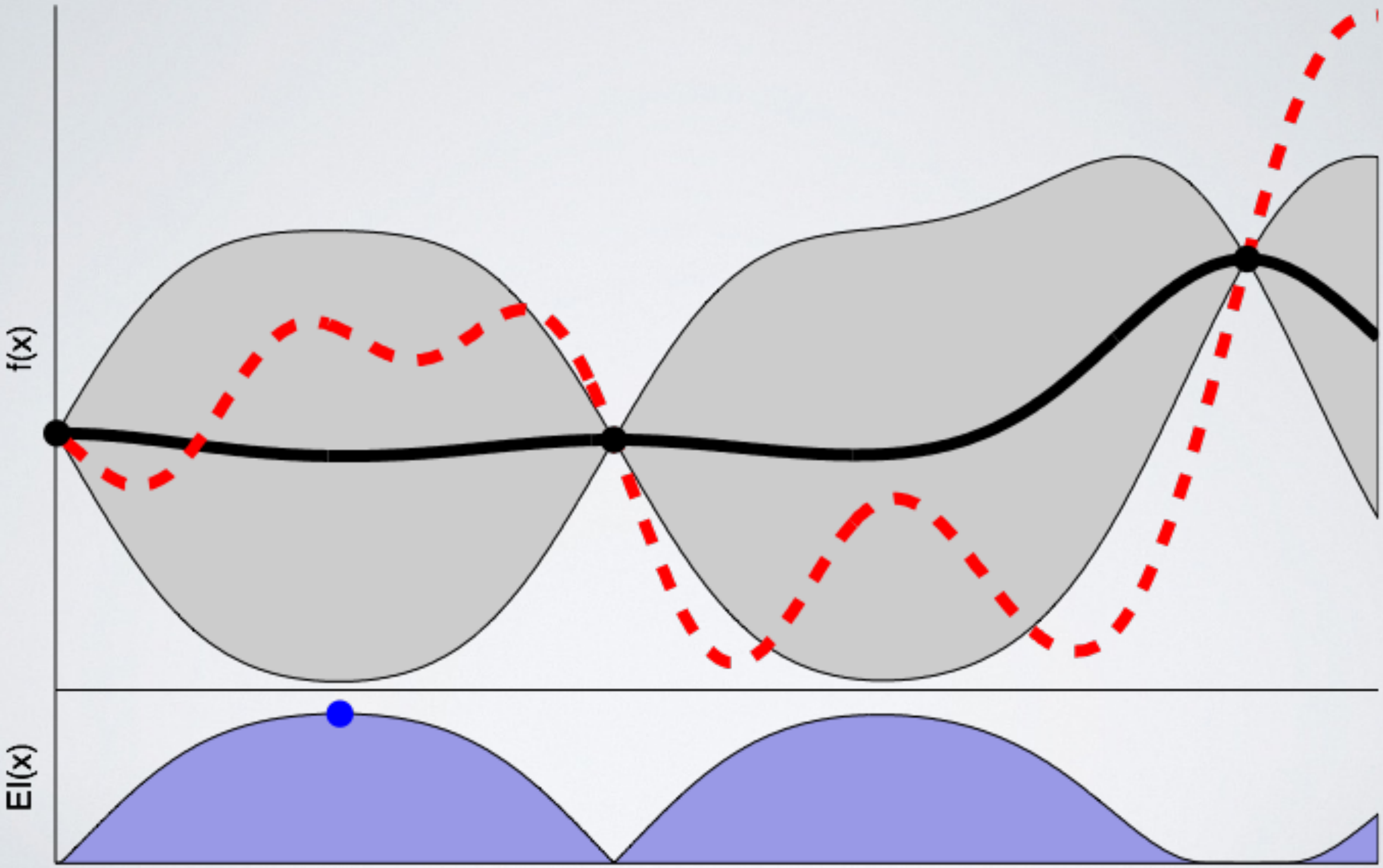
Distribution Over Minimum (Entropy Search)



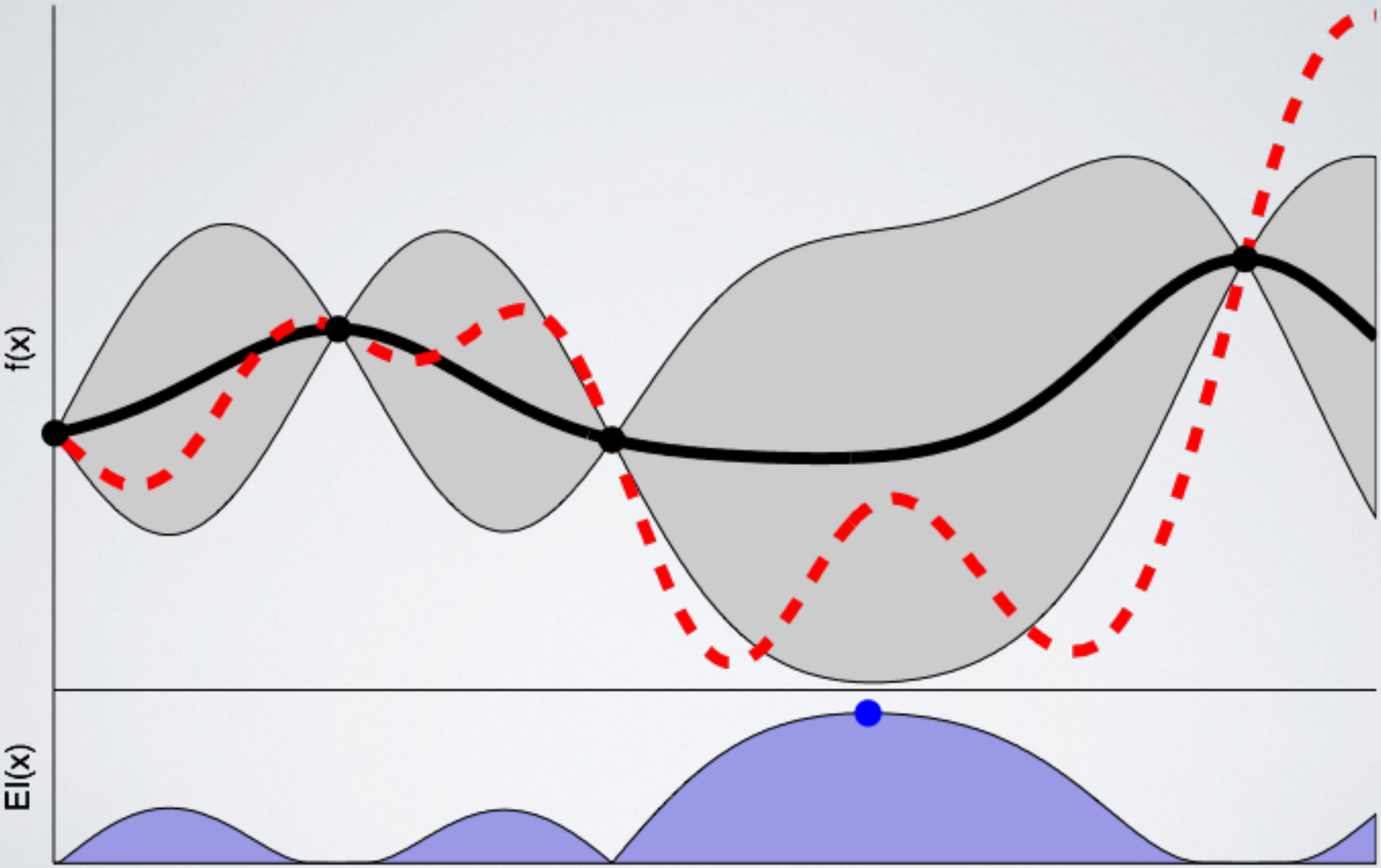
Illustrating Bayesian Optimization



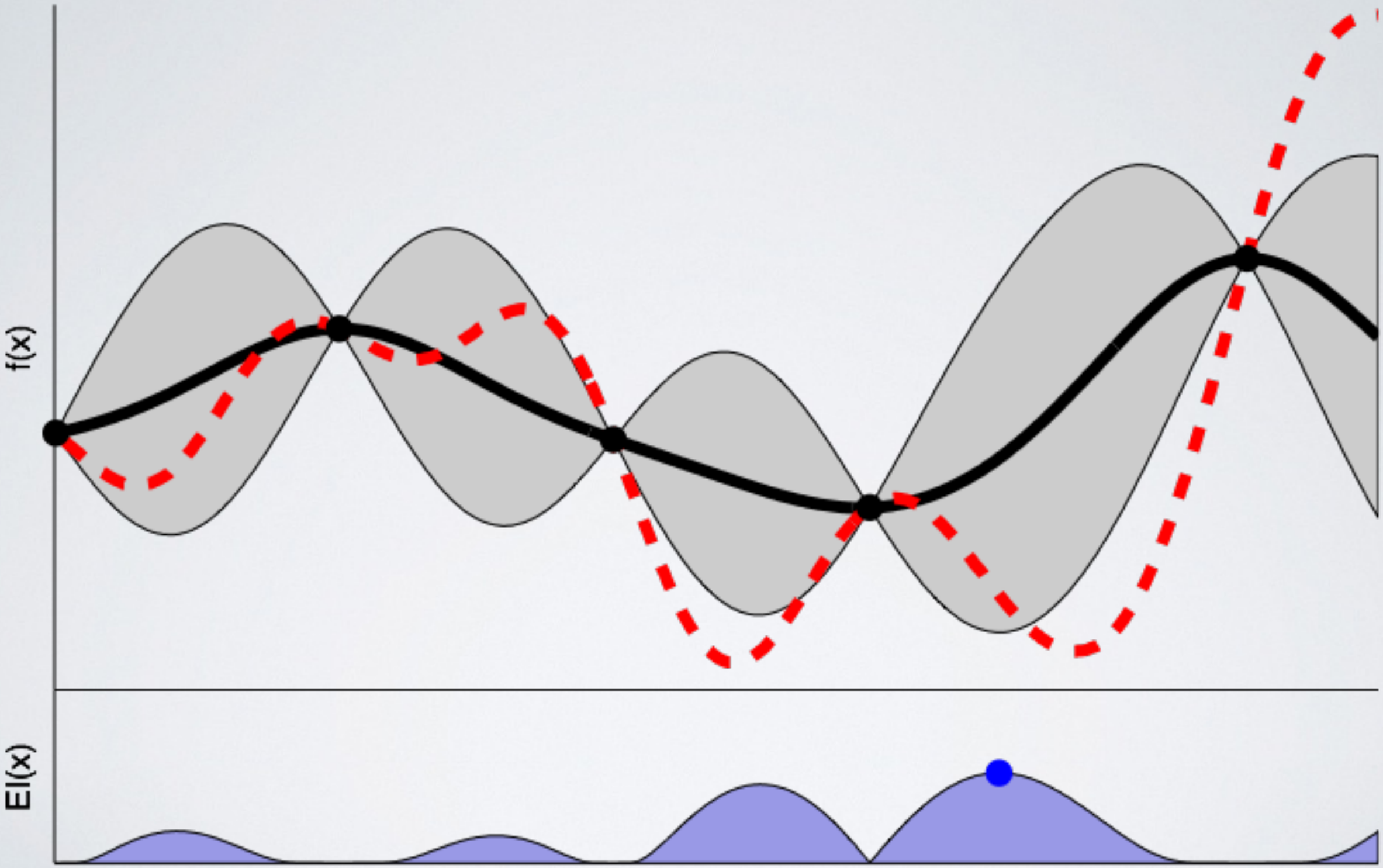
Illustrating Bayesian Optimization



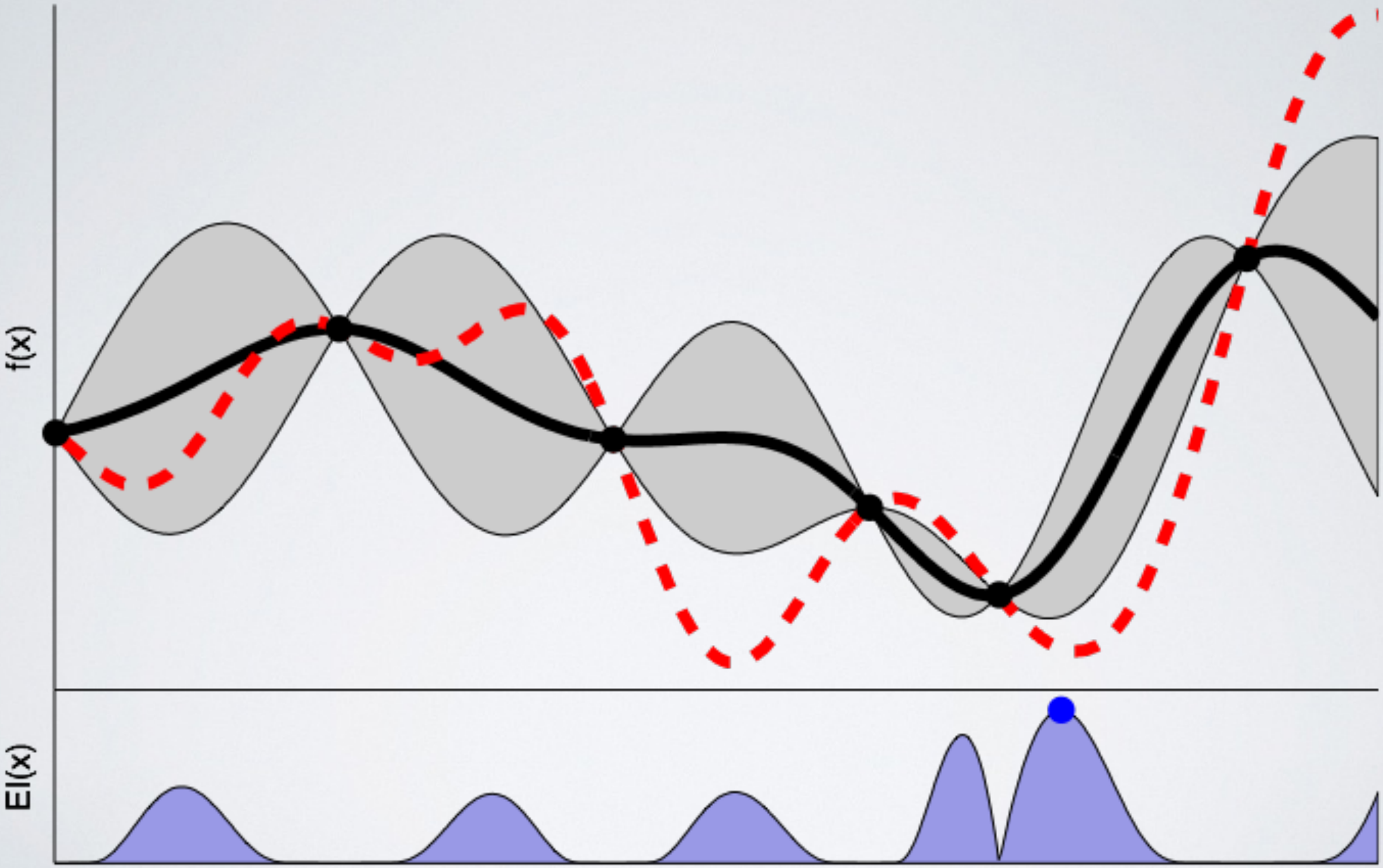
Illustrating Bayesian Optimization



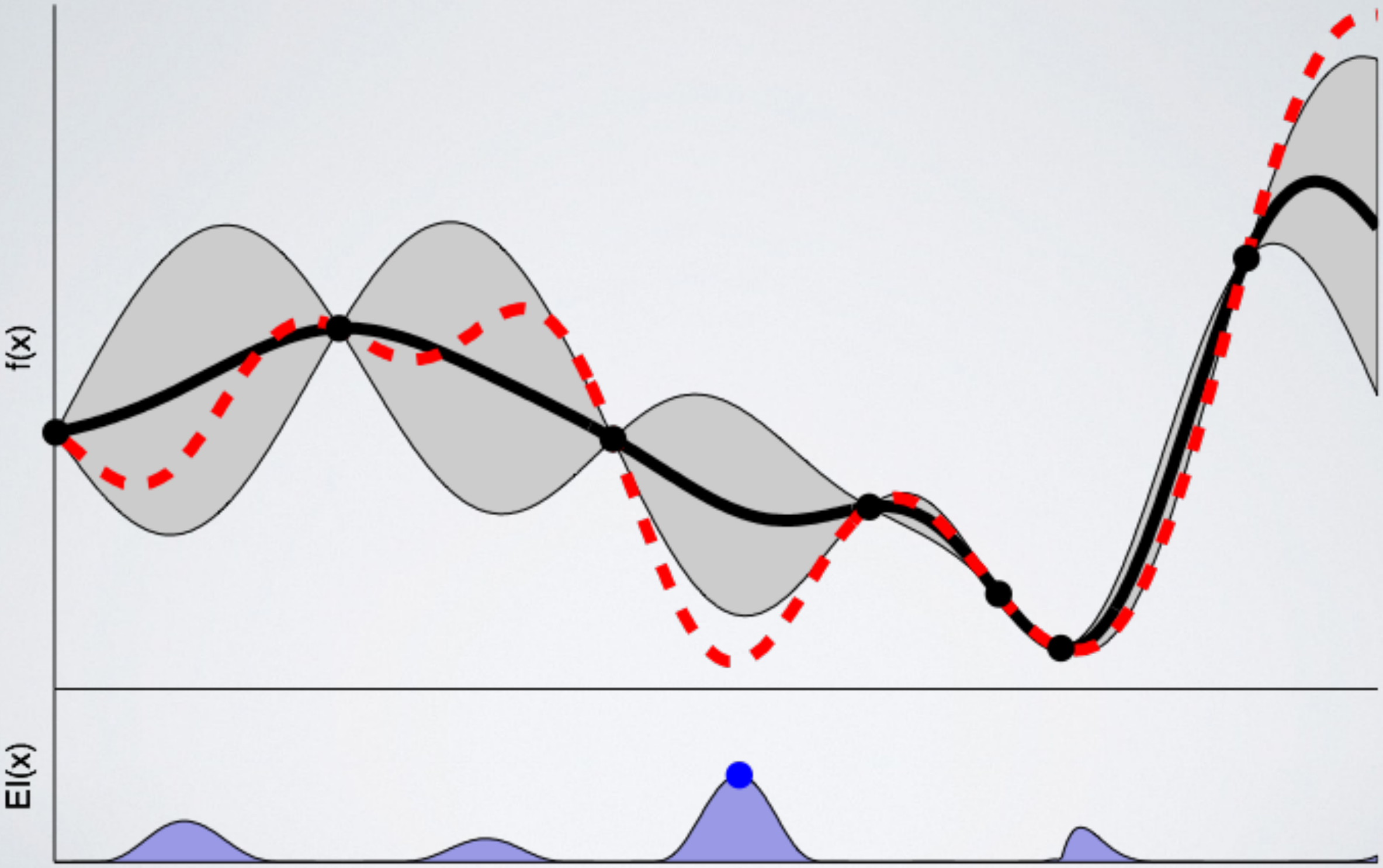
Illustrating Bayesian Optimization



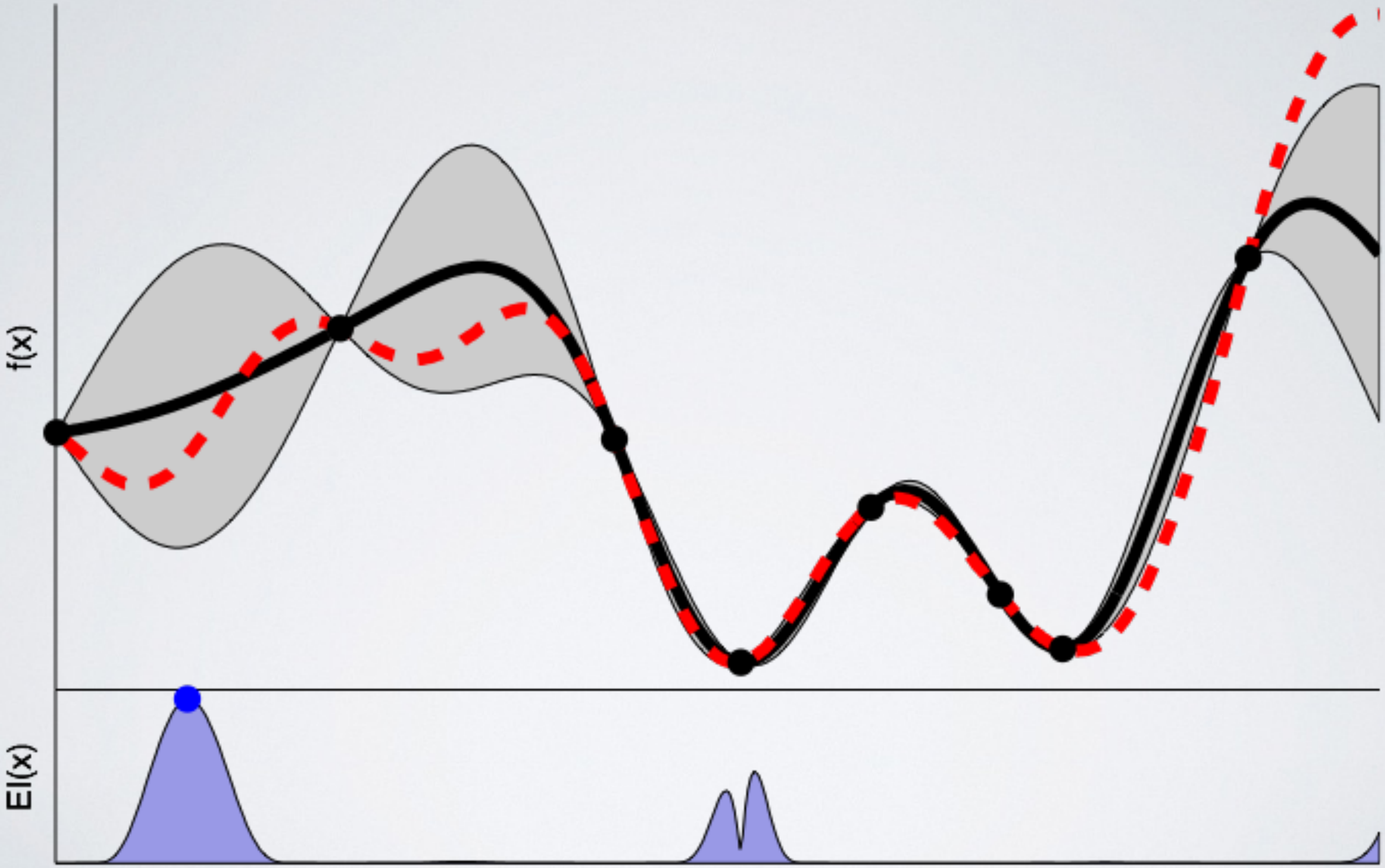
Illustrating Bayesian Optimization



Illustrating Bayesian Optimization



Illustrating Bayesian Optimization



Why Doesn't Everyone Use This?

These ideas have been around for *decades*.
Why is Bayesian optimization in broader use?

- **Fragility and poor default choices.**
  Getting the function model wrong can be catastrophic.
- **There hasn't been standard software available.**
  It's a bit tricky to build such a system from scratch.
- **Experiments are run sequentially.**
  We want to take advantage of cluster computing.
- **Limited scalability in dimensions and evaluations.**
  We want to solve big problems.

Fragility and Poor Default Choices

Ironic Problem:

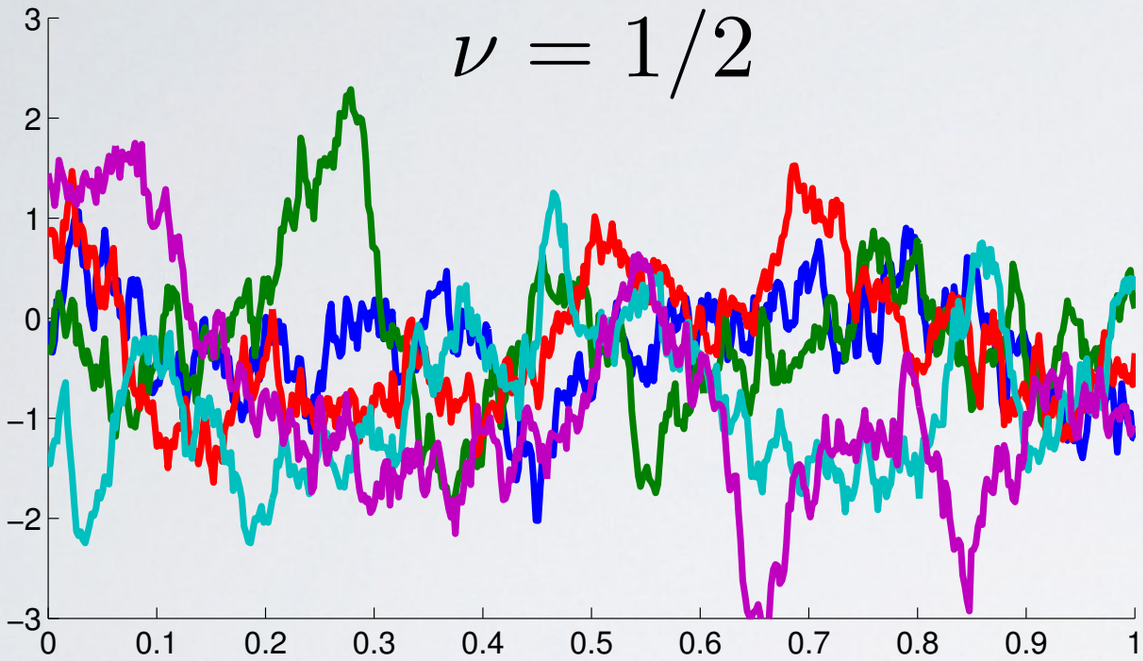
Bayesian optimization has its own hyperparameters!

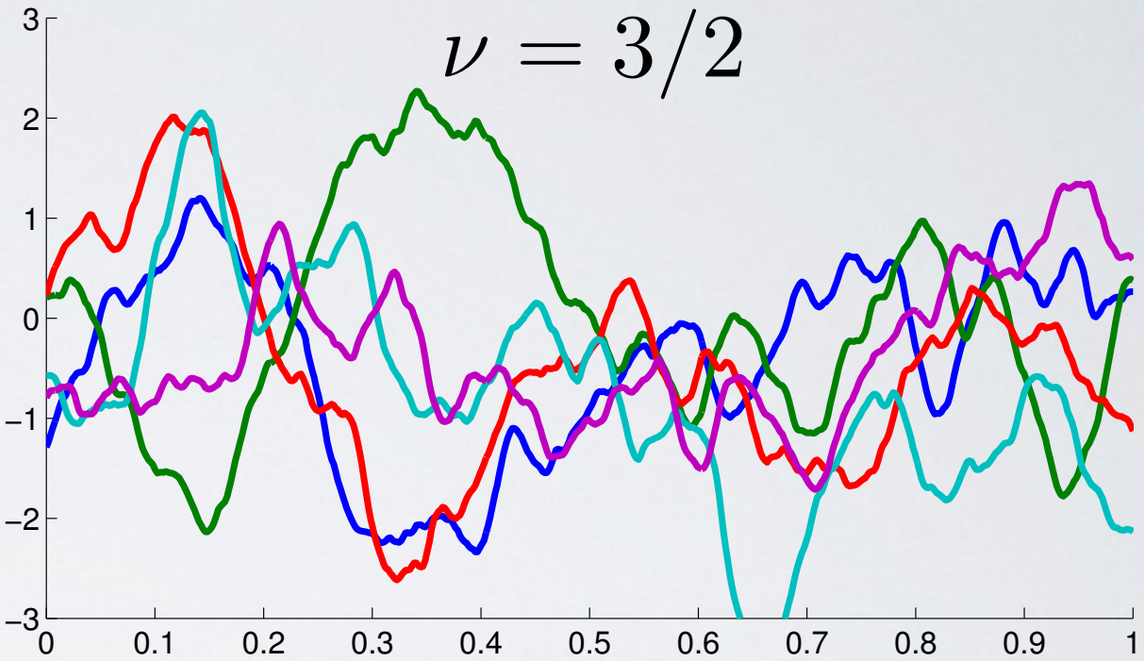
- **Covariance function selection**
  This turns out to be crucial to good performance.
  The default choice for regression is *way* too smooth.
  Instead: use adaptive Matèrn 3/5 kernel.

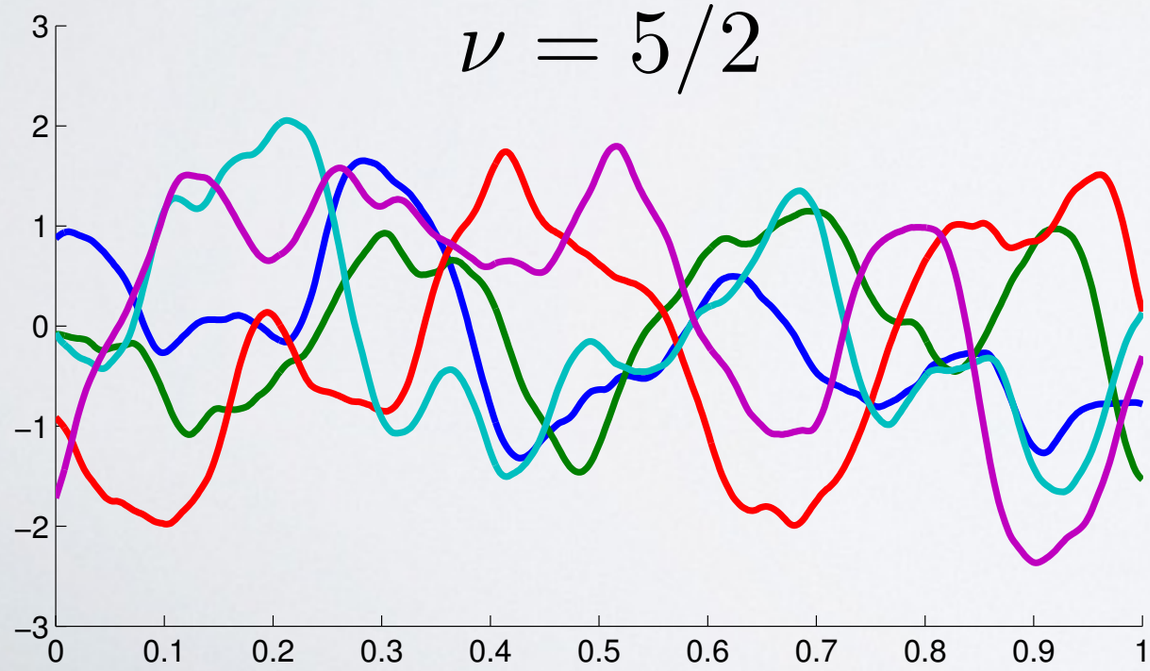
- **Gaussian process hyperparameters**
  Typical empirical Bayes approach can fail horribly.
  Instead: use Markov chain Monte Carlo integration.
  Slice sampling means no additional parameters!

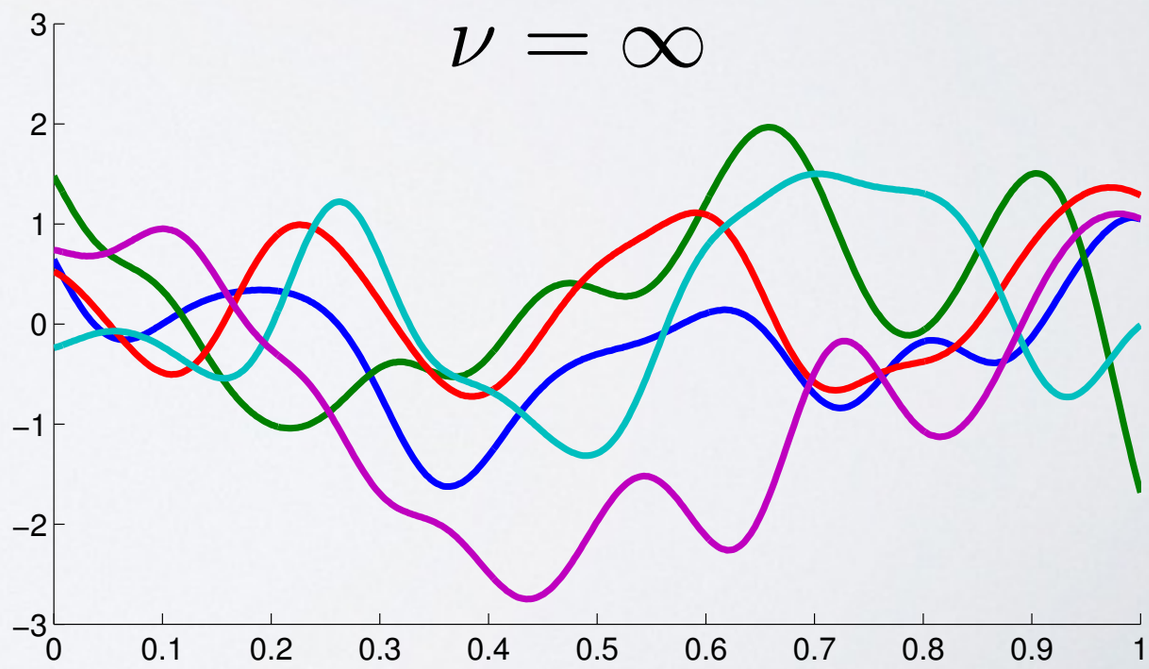
Covariance Function Choice

$$C(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\, r}{\ell} \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}\, r}{\ell} \right)$$



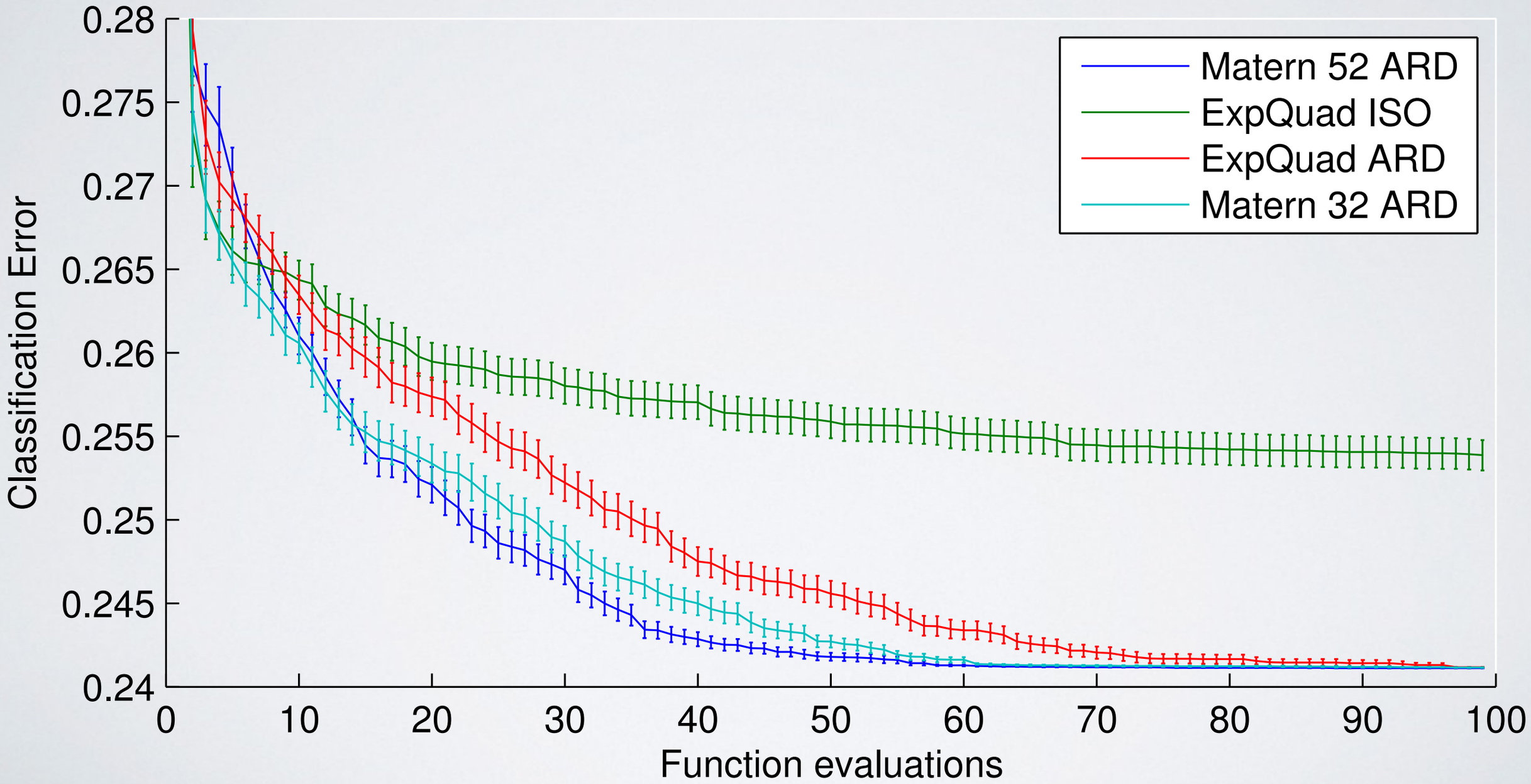






Choosing Covariance Functions

Structured SVM for Protein Motif Finding
Miller et al (2012)



Snoek, Larochelle & RPA, NIPS 2012

MCMC for GP Hyperparameters

- Covariance hyperparameters are often optimized rather than marginalized, typically in the name of convenience and efficiency.
- Slice sampling of hyperparameters (e.g., Murray and Adams 2010) is comparably fast and easy, but accounts for uncertainty in length scale, mean, and amplitude.
- Integrated Acquisition Function:

$$\hat{a}(x) = \int a(x\,;\,\theta)\, p(\theta \,|\, \{x_n, y_n\}_{n=1}^N)\, \mathrm{d}\theta$$

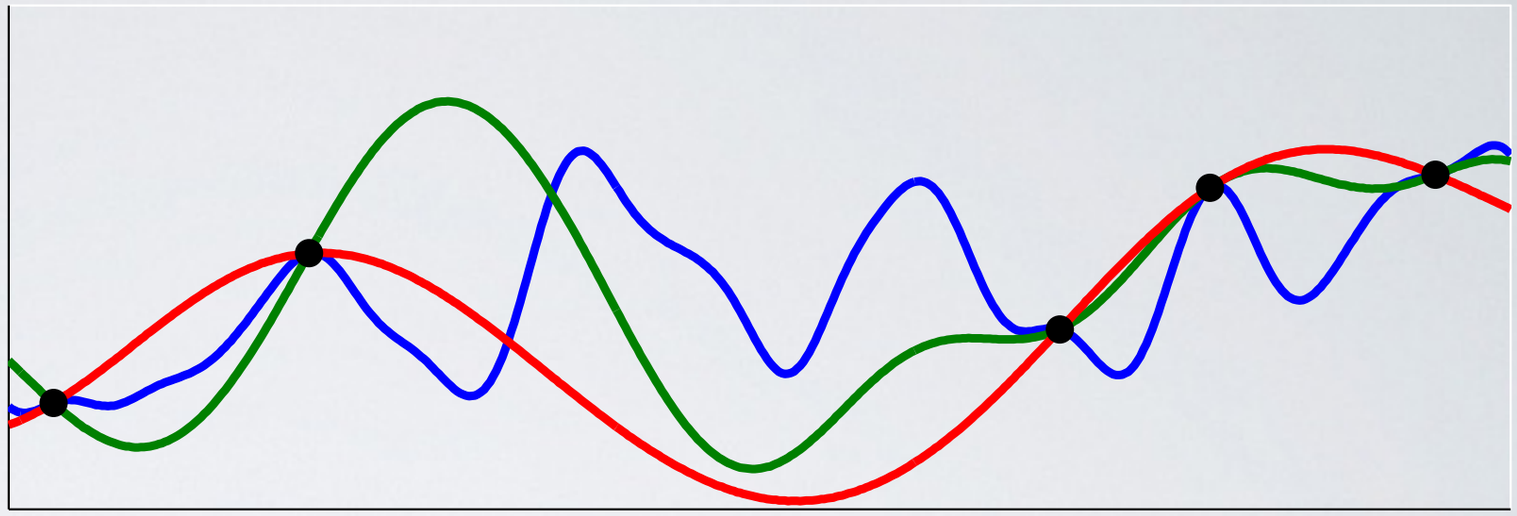
$$\approx \frac{1}{K} \sum_{k=1}^{K} a(x\,;\,\theta^{(k)}) \qquad \theta^{(k)} \sim p(\theta \,|\, \{x_n, y_n\}_{n=1}^N)$$

- For a theoretical discussion of the implications of inferring hyperparameters with BayesOpt, see recent work by Wang and de Freitas (http://arxiv.org/abs/1406.7758)

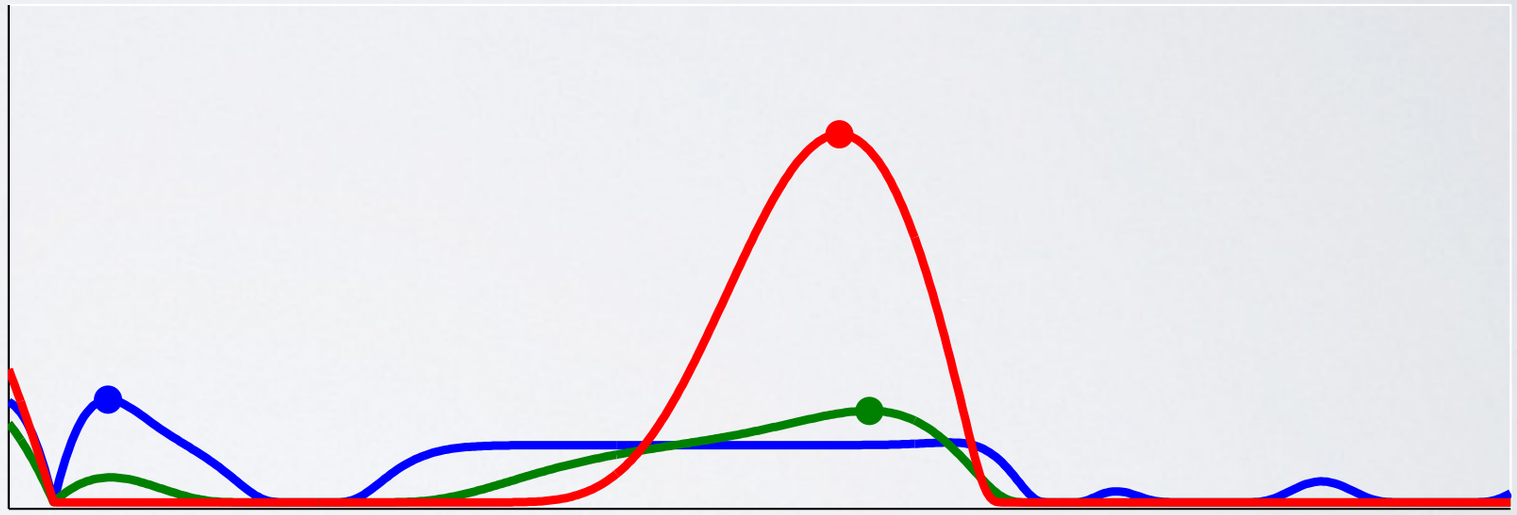
Snoek, Larochelle & RPA, NIPS 2012

Integrating Out GP Hyperparameters

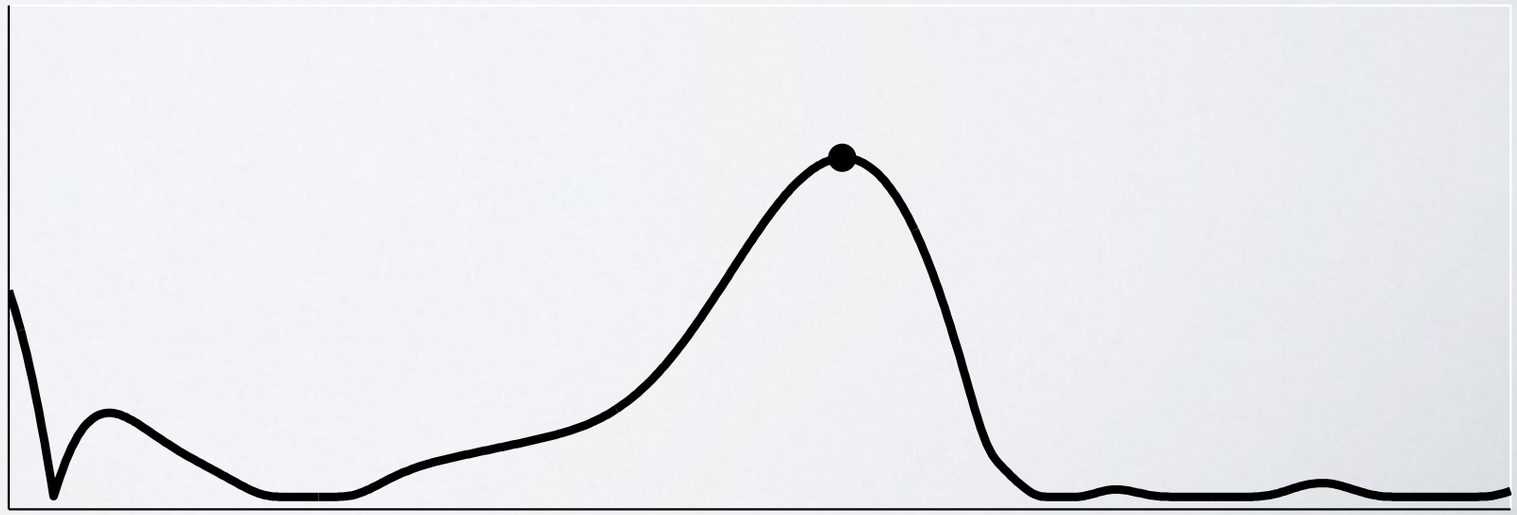
Posterior samples with three different length scales



Length scale specific expected improvement

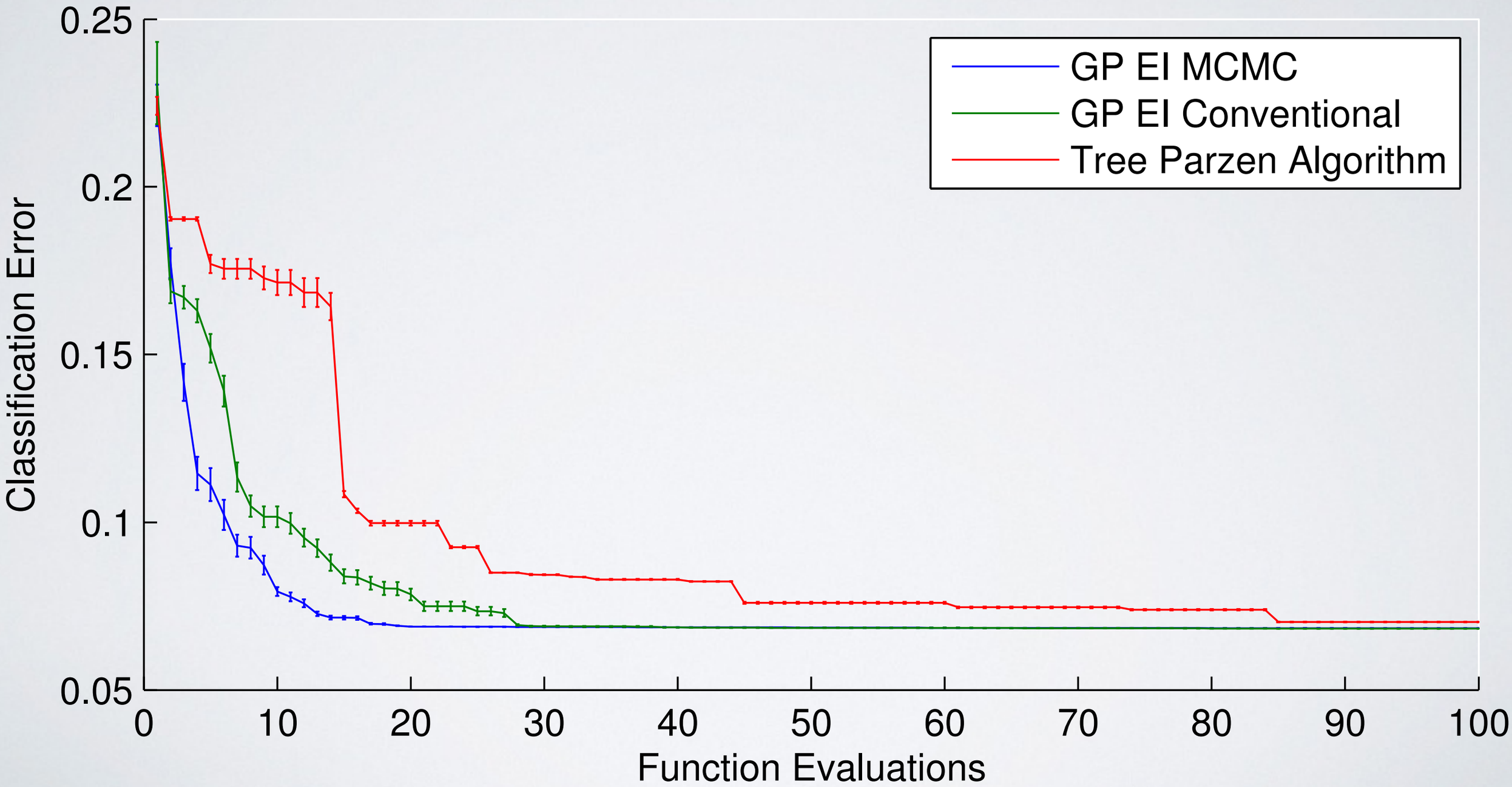


Integrated expected improvement



MCMC for Hyperparameters

Logistic regression for handwritten digit recognition



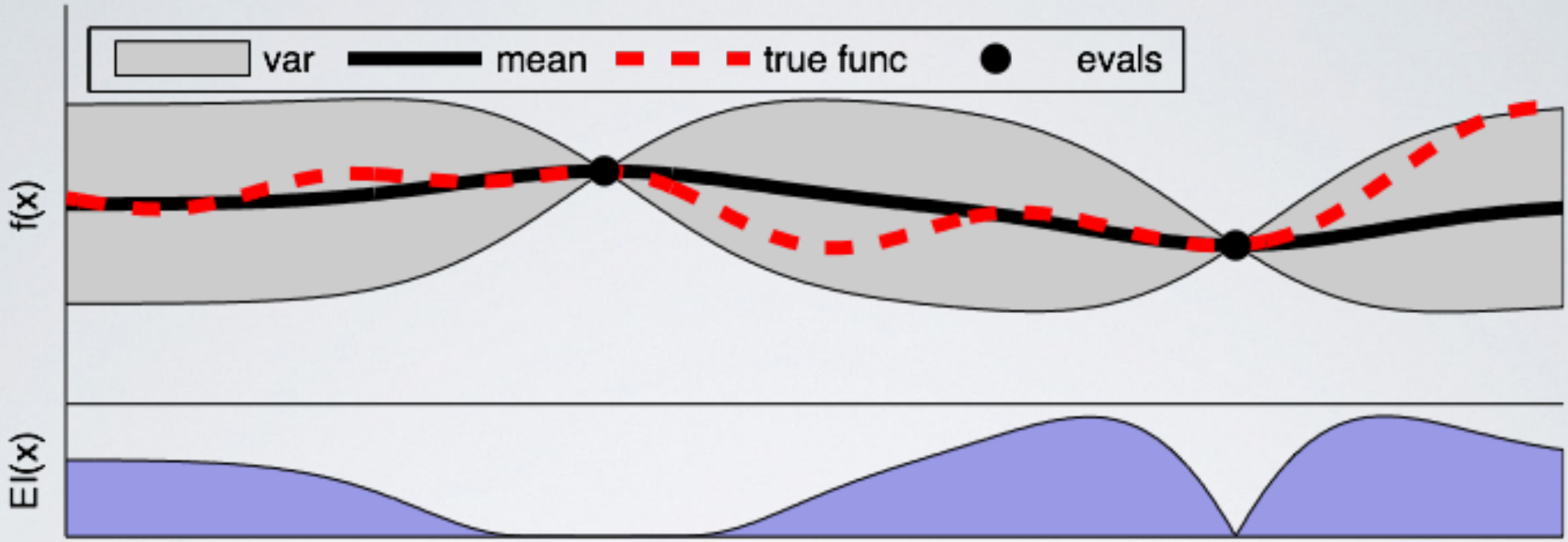
Snoek, Larochelle & RPA, NIPS 2012

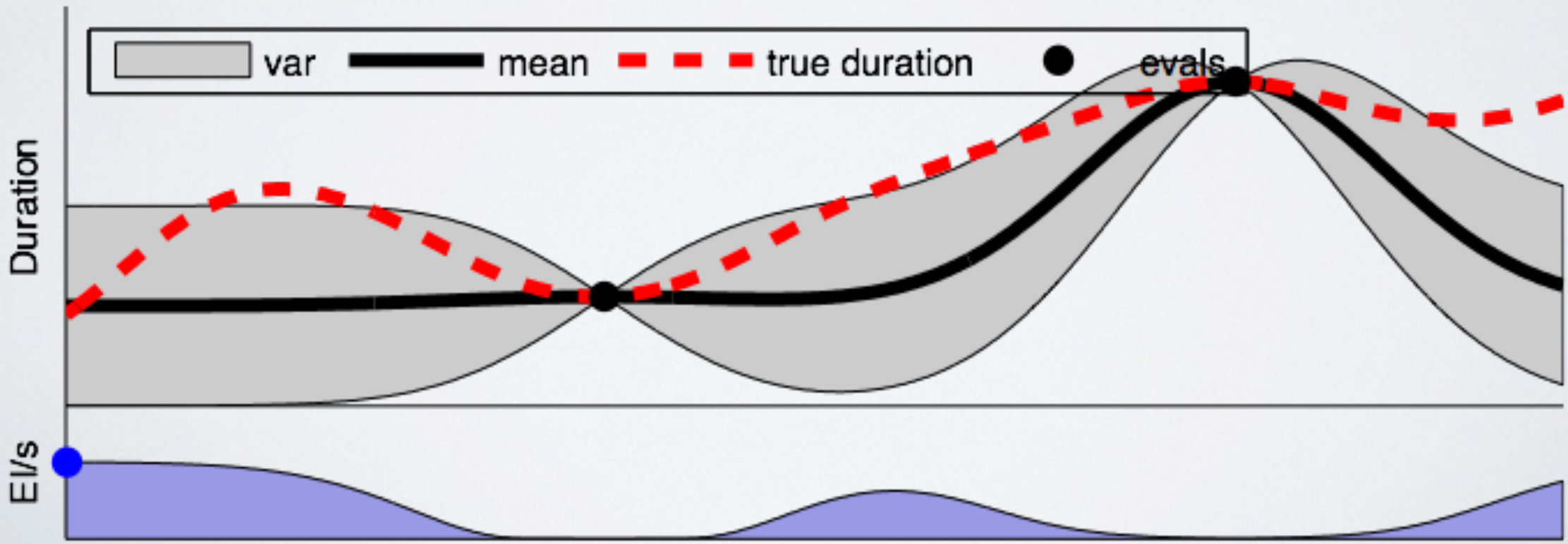
Cost-Sensitive Bayesian Optimization

- Practical problems often have varying costs, e.g., durations, hardware requirements, materiel.
- We often have a budget or a deadline and would ideally like to account for these limited resources.
- It may be sensible to first characterize the function with cheap evaluations before trying expensive ones.
- These costs are probably unknown, but they can themselves be modeled with Gaussian processes.
- One idea: expected improvement per second

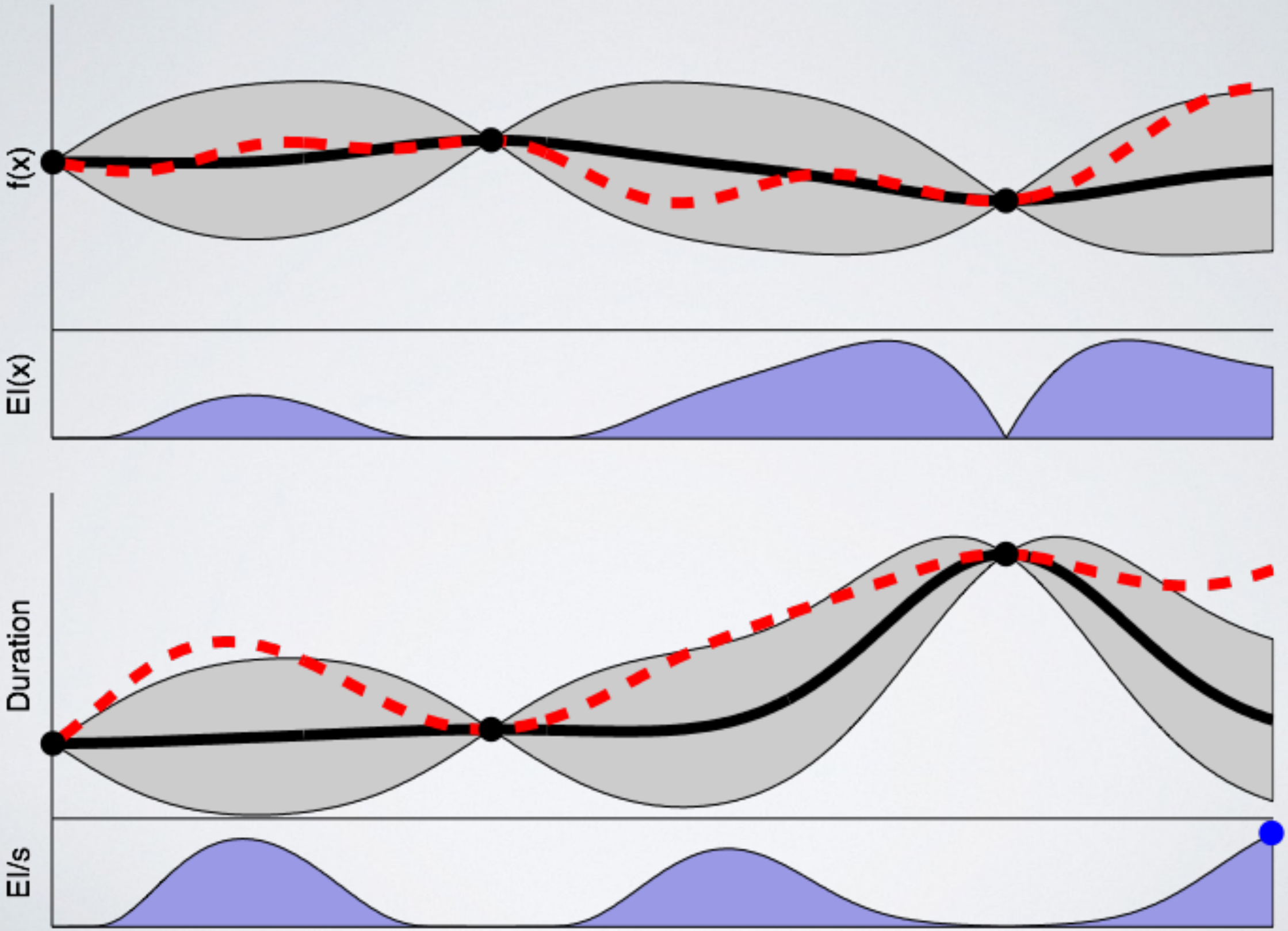
Snoek, Larochelle & RPA, NIPS 2012

Expected Improvement per Second

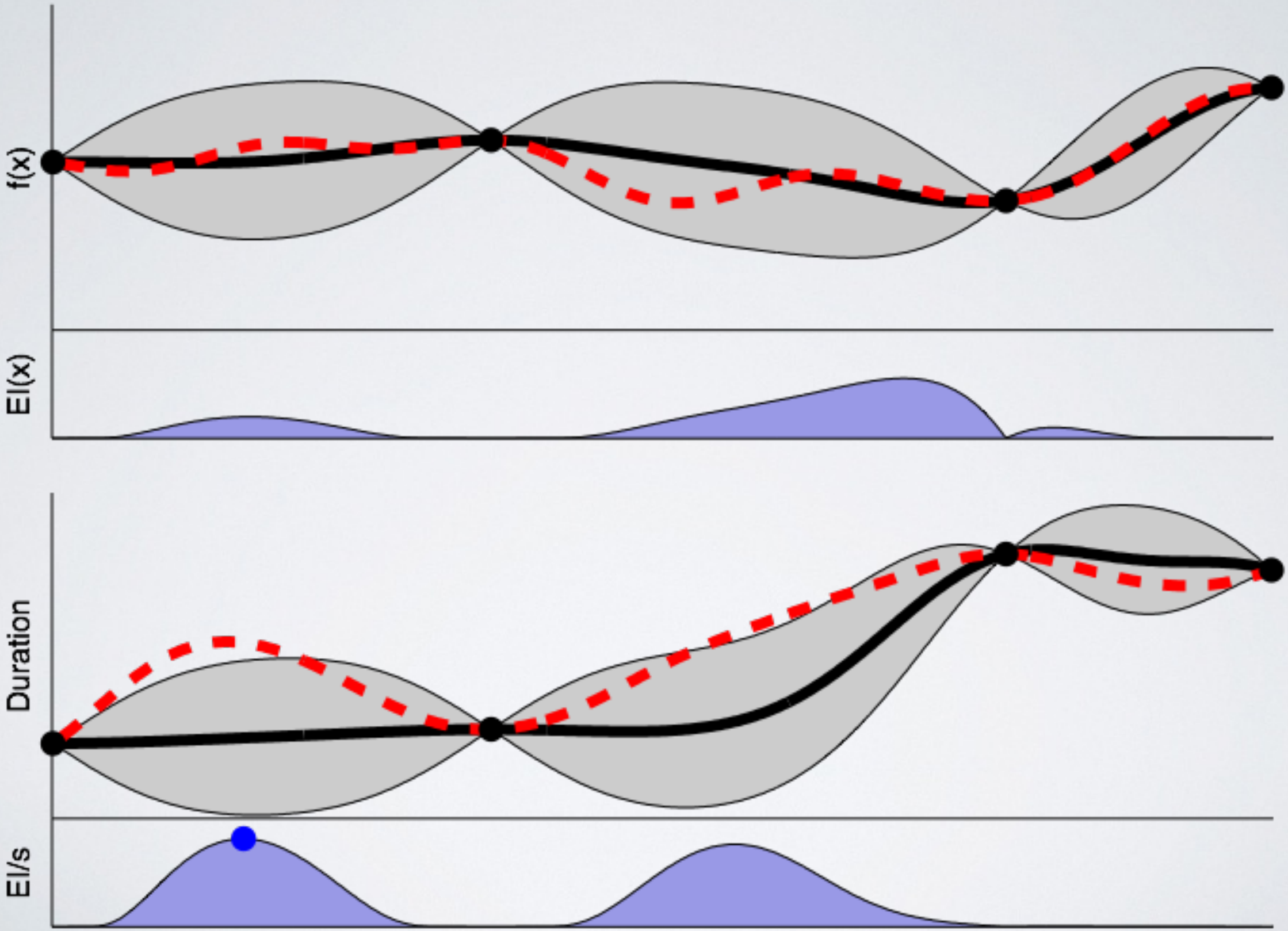




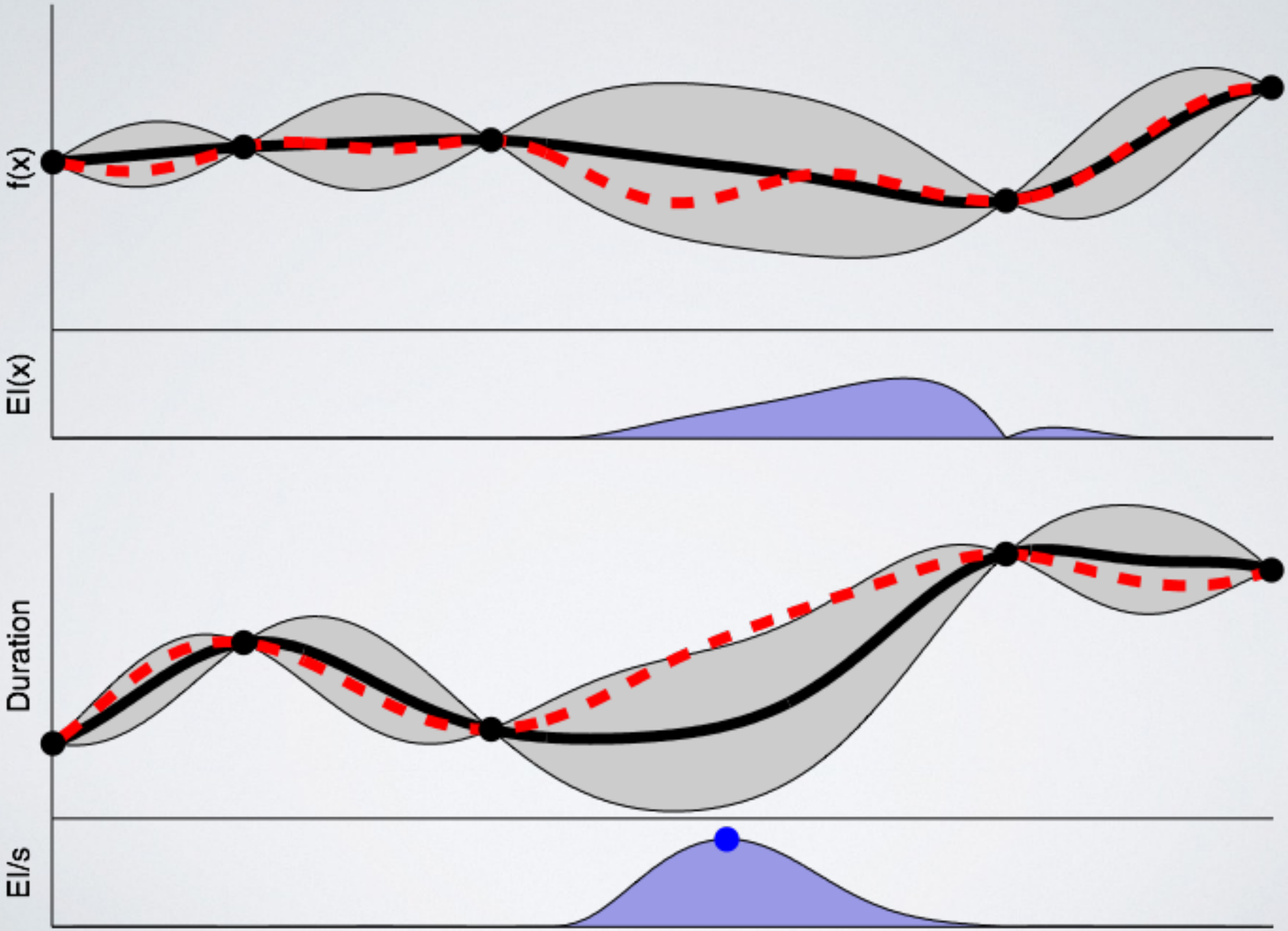
Expected Improvement per Second



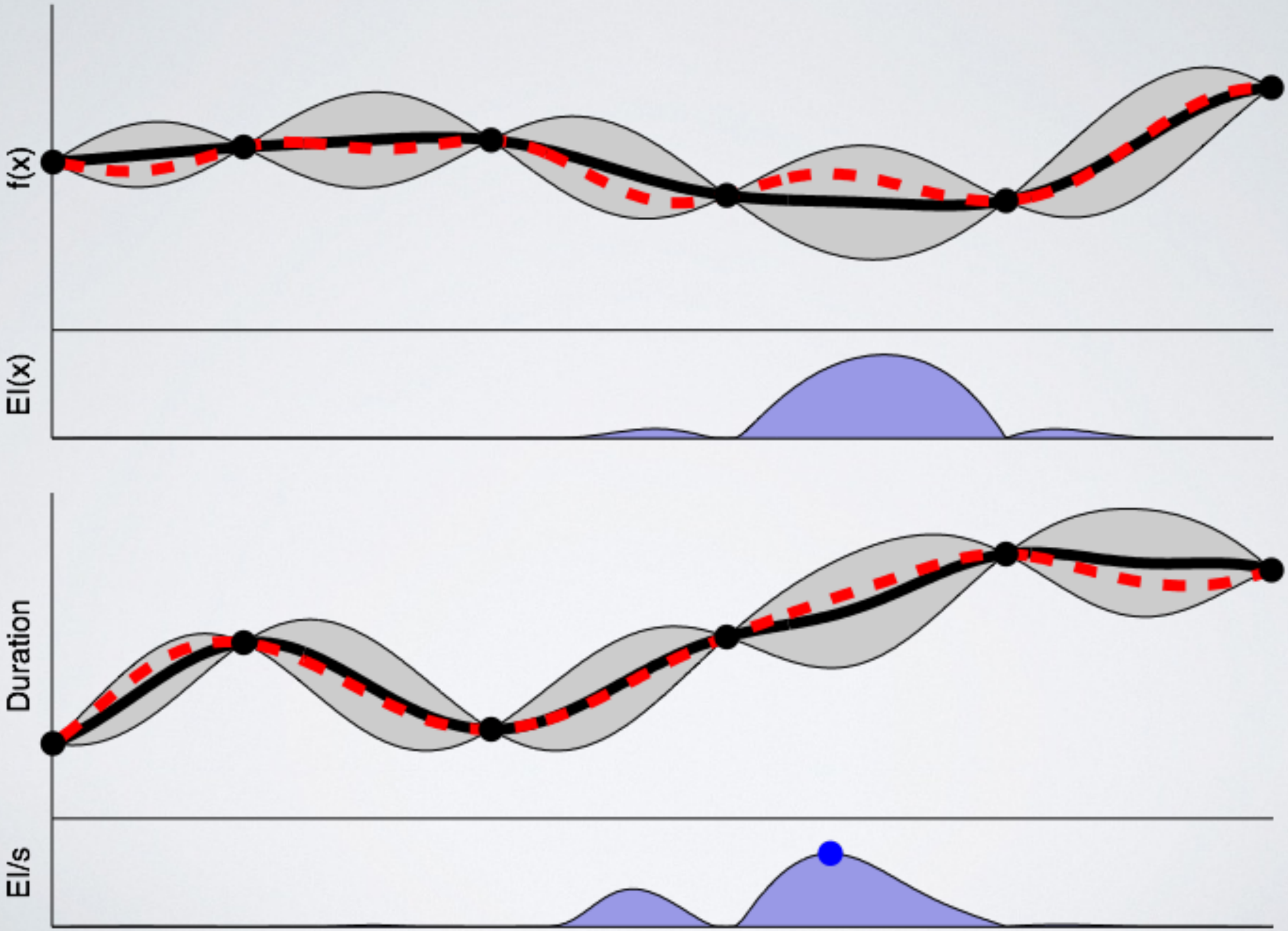
Expected Improvement per Second



Expected Improvement per Second

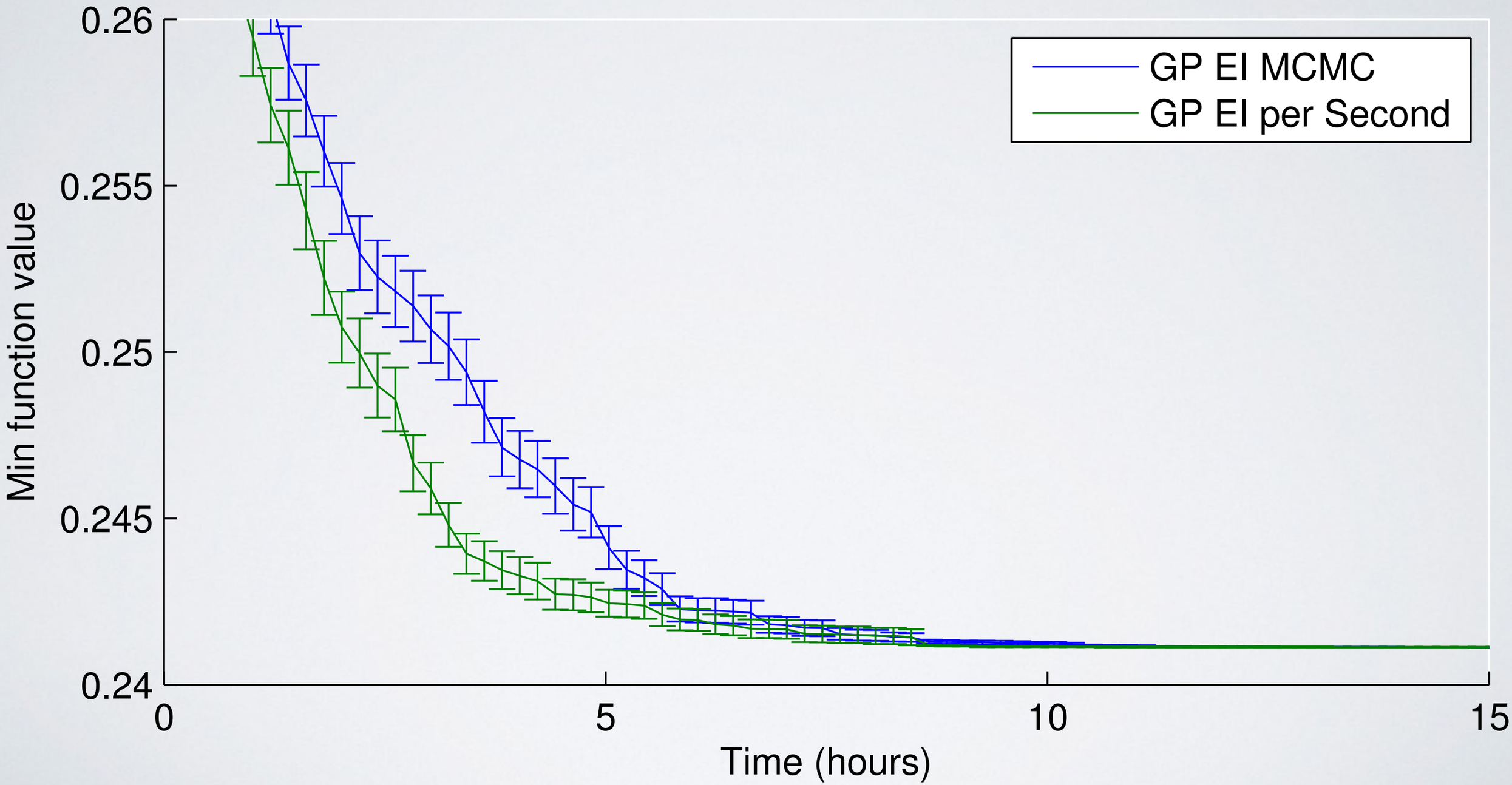


Expected Improvement per Second



Expected Improvement per Second

Structural SVM for Protein Motif Finding
Miler et al. (2012)

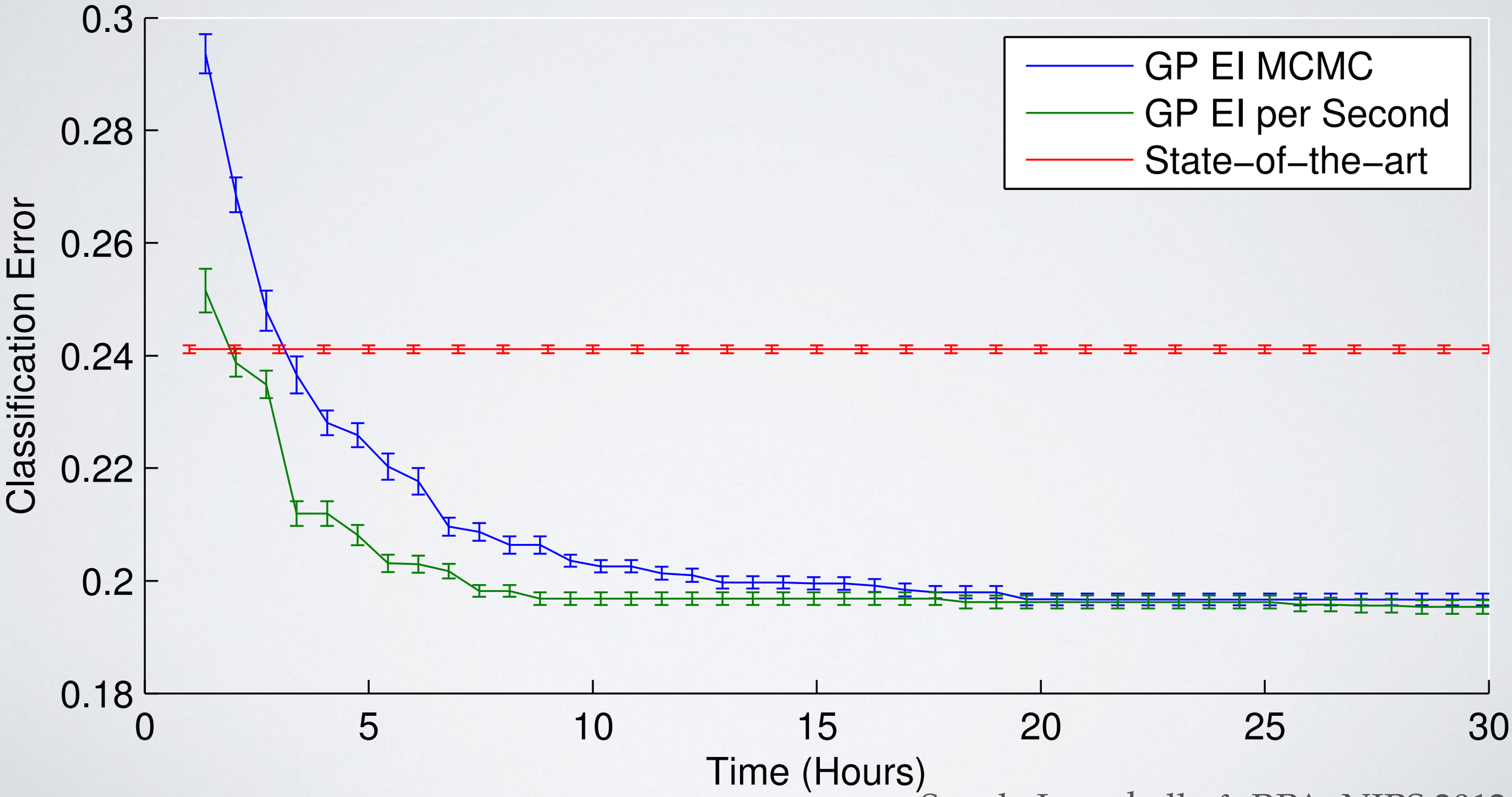


Snoek, Larochelle & RPA, NIPS 2012

Expected Improvement per Second

CIFAR10: Deep convolutional neural net (Krizhevsky)

Achieves 9.5% test error vs. 11% with hand tuning.



Snoek, Larochelle & RPA, NIPS 2012

Parallelizing Bayesian Optimization

- **Classical Bayesian optimization is sequential.**
  Do an experiment. Wait until it finishes. Repeat.

- **Compute clusters let us do many things at once.**
  How do we efficiently pick multiple experiments?

- **Fantasize outcomes from the Gaussian process!**
  The probabilistic model gives us coherent predictions.
  Use the predictions to imagine possible futures.
  Evaluate points that are good under the *average future*.

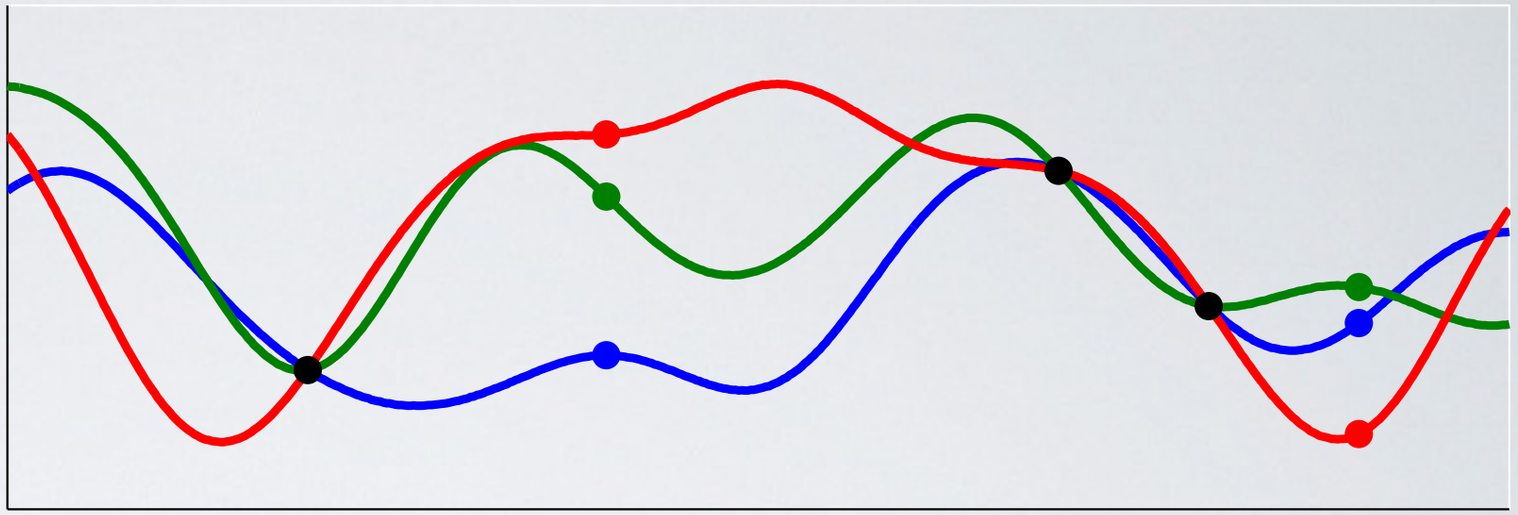
$$\hat{a}(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta, \{\mathbf{x}_j\}) = \int_{\mathbb{R}^J} a(\mathbf{x}\,;\,\{\mathbf{x}_n, y_n\}, \theta, \{\mathbf{x}_j, y_j\})\, p(\{y_j\}_{j=1}^J \mid \{\mathbf{x}_j\}_{j=1}^J, \{\mathbf{x}_n, y_n\}_{n=1}^N)\, \mathrm{d}y_1 \cdots \mathrm{d}y_J$$

- **Alternative: shrink the variance and integrate out the mean**, as in Desautels, Krause & Burdick, JMLR 2014.

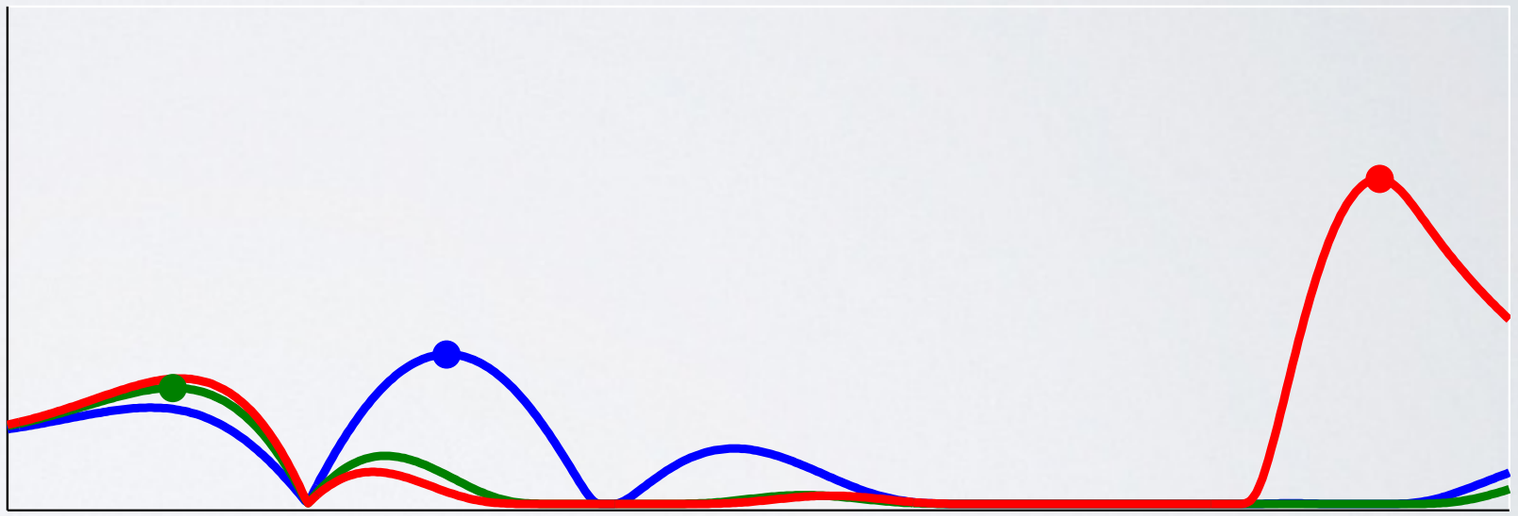
Snoek, Larochelle & RPA, NIPS 2012

Parallelizing Bayesian Optimization

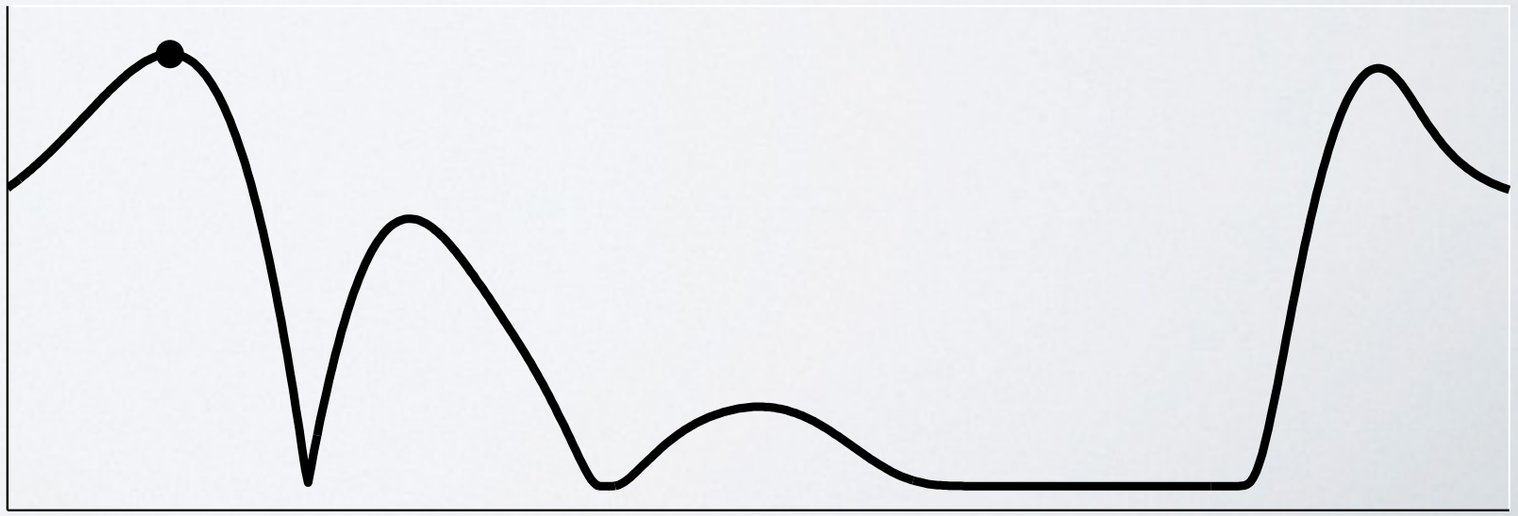
Two pending experiments, with three joint fantasies



Three expected improvement functions, one for each fantasy

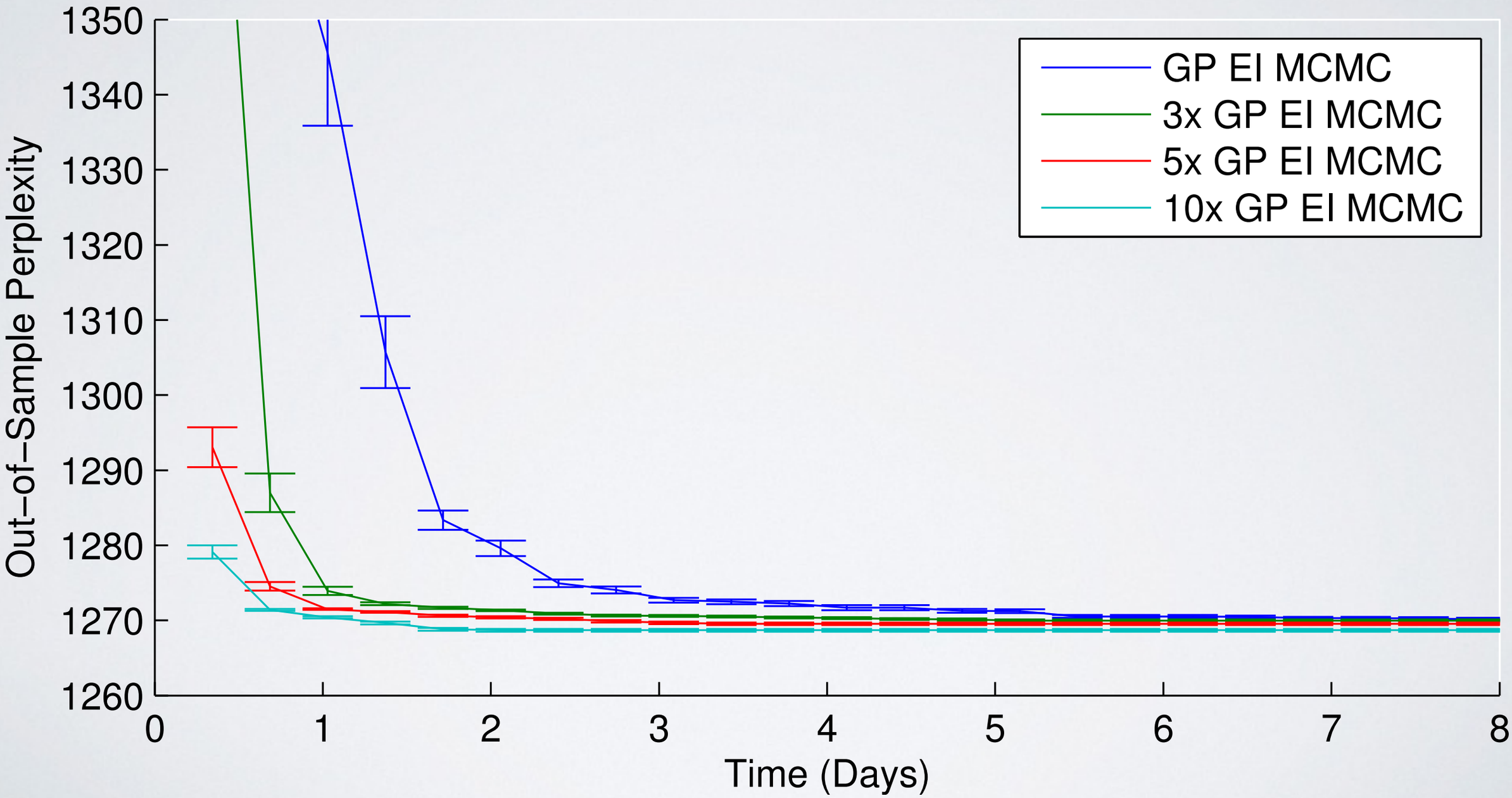


Integrating out the fantasies yields an overall expected improvement



Parallelized Bayesian Optimization

Online Latent Dirichlet Allocation (250K documents)
Hoffman, Blei & Bach (2010)



Snoek, Larochelle & RPA, NIPS 2012

Multi-Task Bayesian Optimization

- **We usually don't just have one problem to solve.**
  Probably, we have a set of related objectives.

- **Examples:**
  Object classification for different image types.
  Speech recognition for different languages/speakers.
  Different folds of the same cross-validation setup.

- **How can we share information between problems?**
  Use a prior on vector-valued functions.
  Reframe acquisitions to leverage output covariance.
  Make decisions that look for *bits* about the minimum.

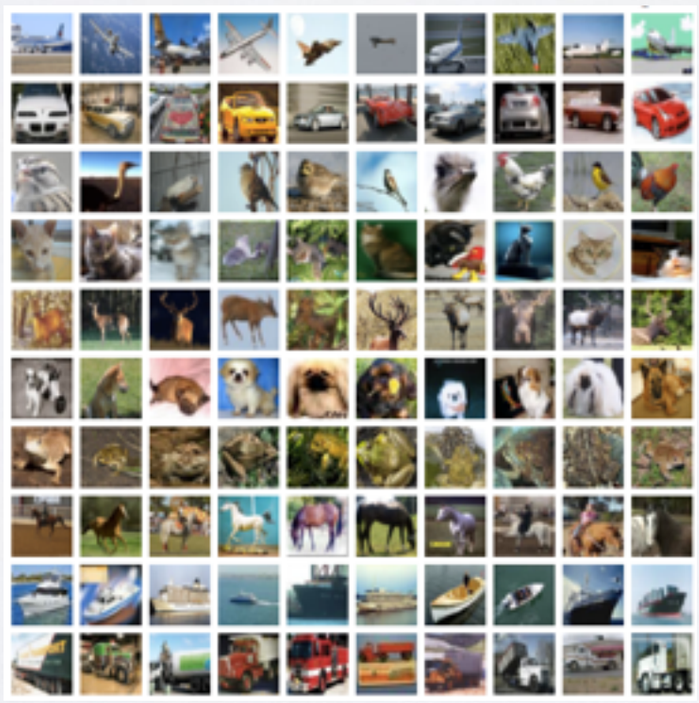
Swersky, Snoek & RPA, NIPS 2013

Parallel-Task Bayesian Optimization

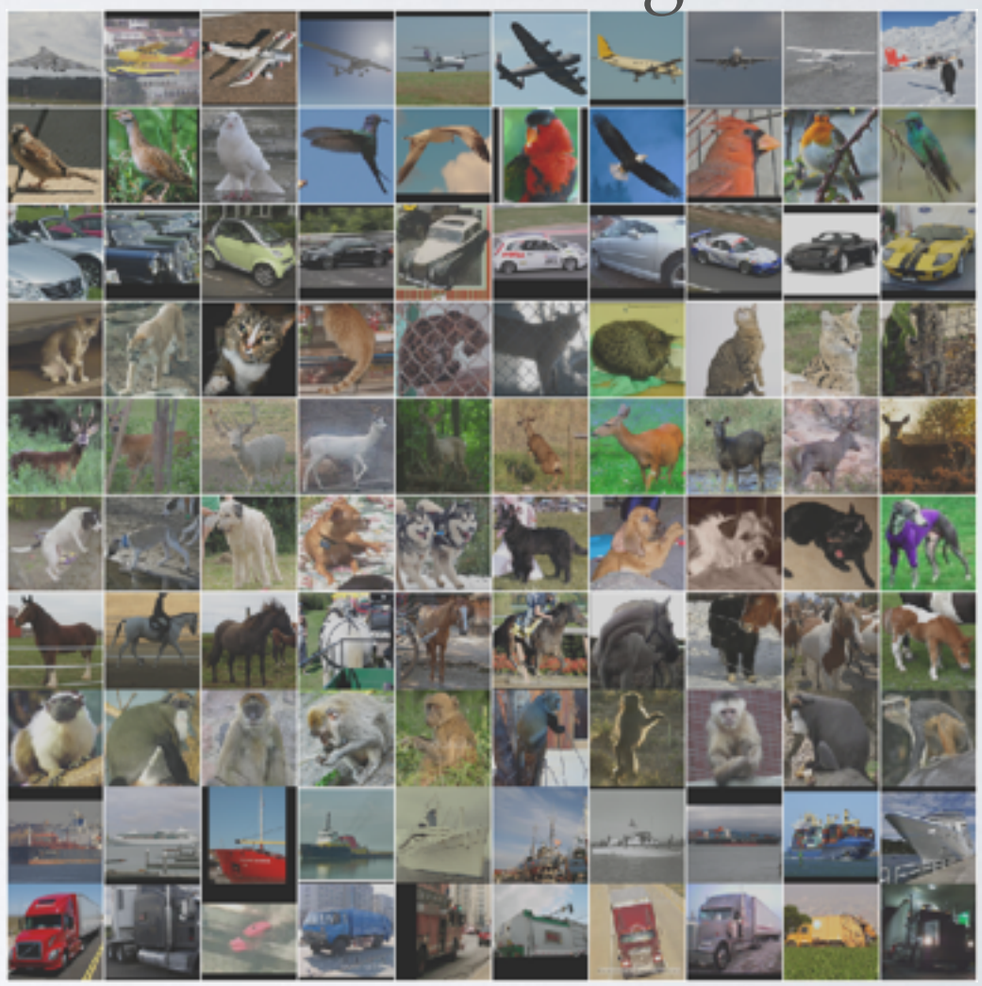
Convolutional Neural Network with Identical Architectures

Google Street View
House Numbers:
73K 32x32 Digits

CIFAR-10
50K 32x32
Natural Images



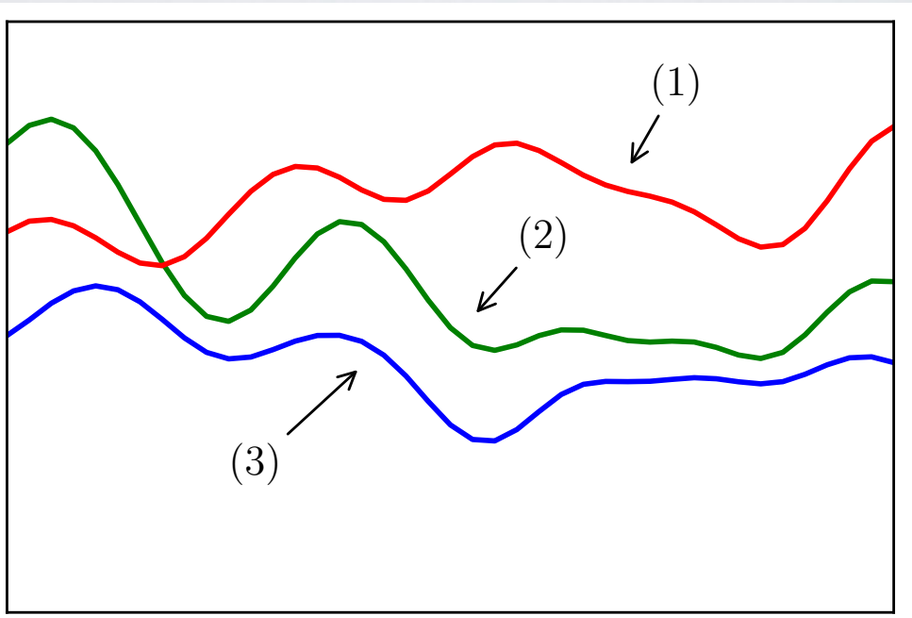
STL-10
5K 96x96
Natural Images



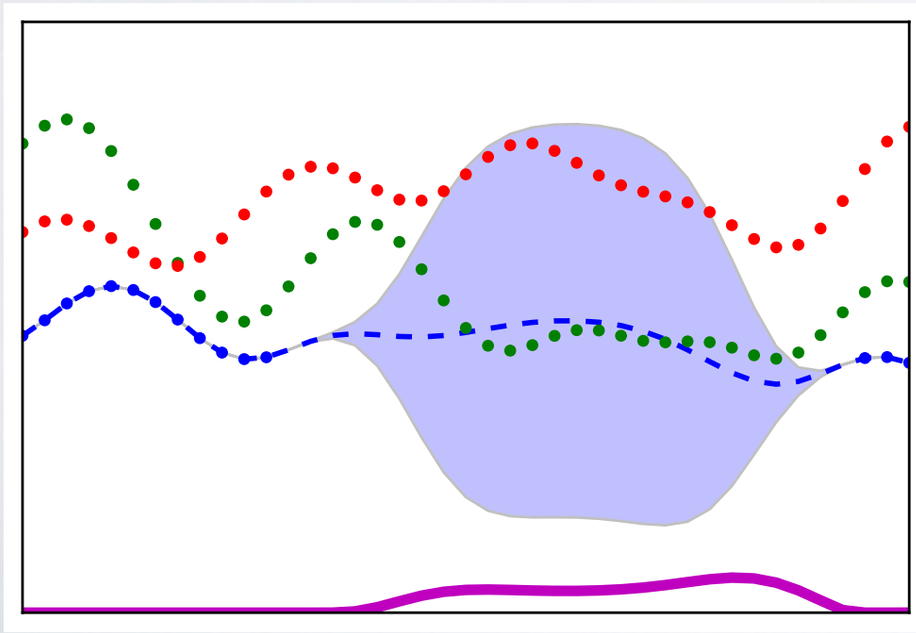
Swersky, Snoek & RPA, NIPS 2013

Correlated Objective Functions

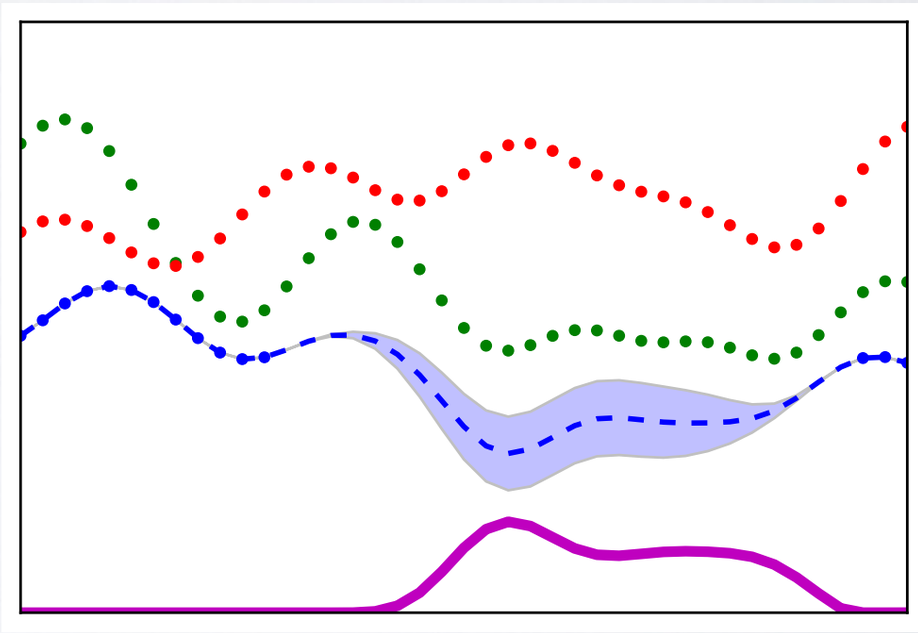
dependent functions



independent predictions



dependent predictions

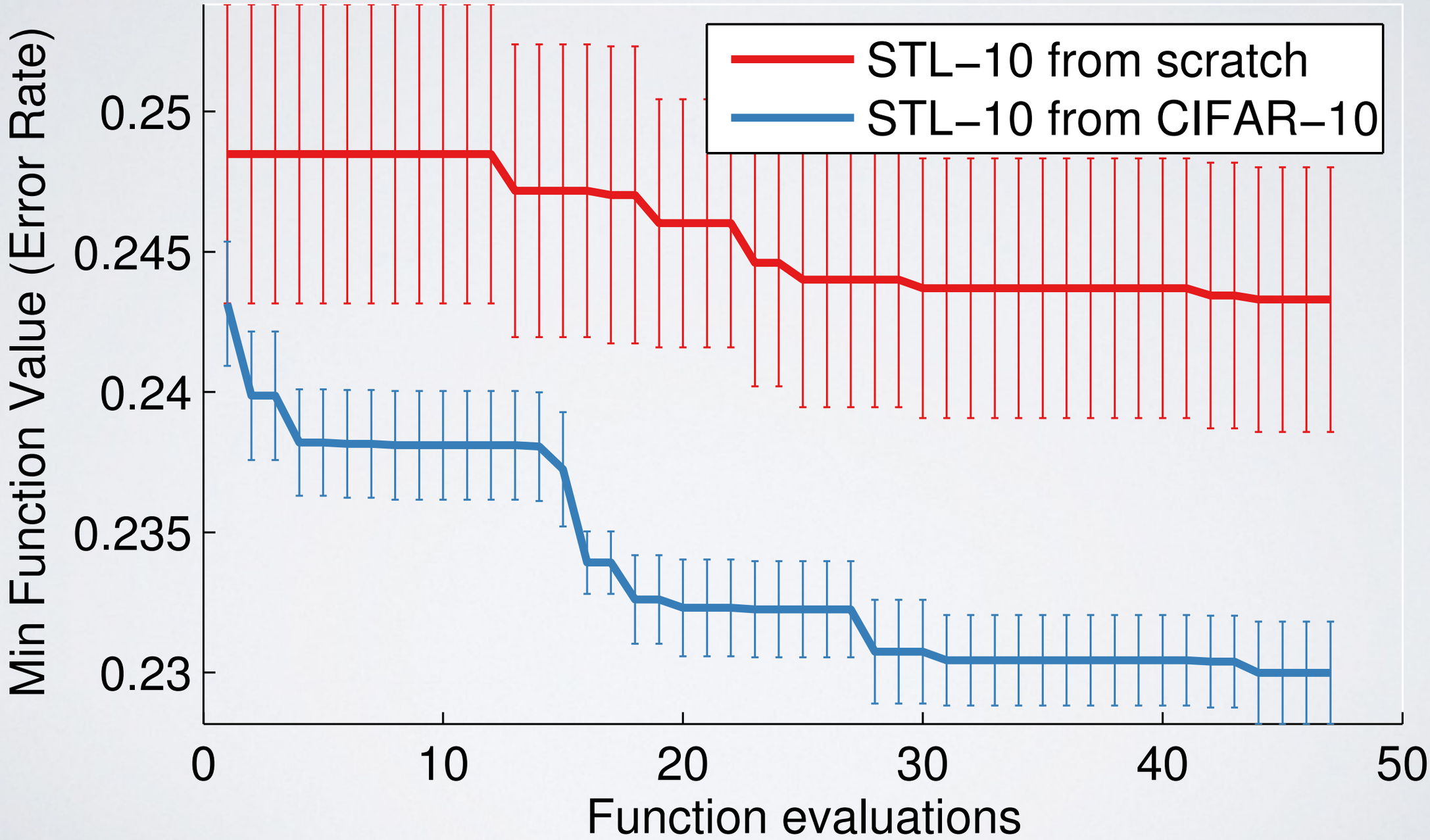


Swersky, Snoek & RPA, NIPS 2013

Parallel-Task Bayesian Optimization

Visual object recognition: STL-10 and CIFAR-10

Achieves 70.7% test -- best previous was 64.5%

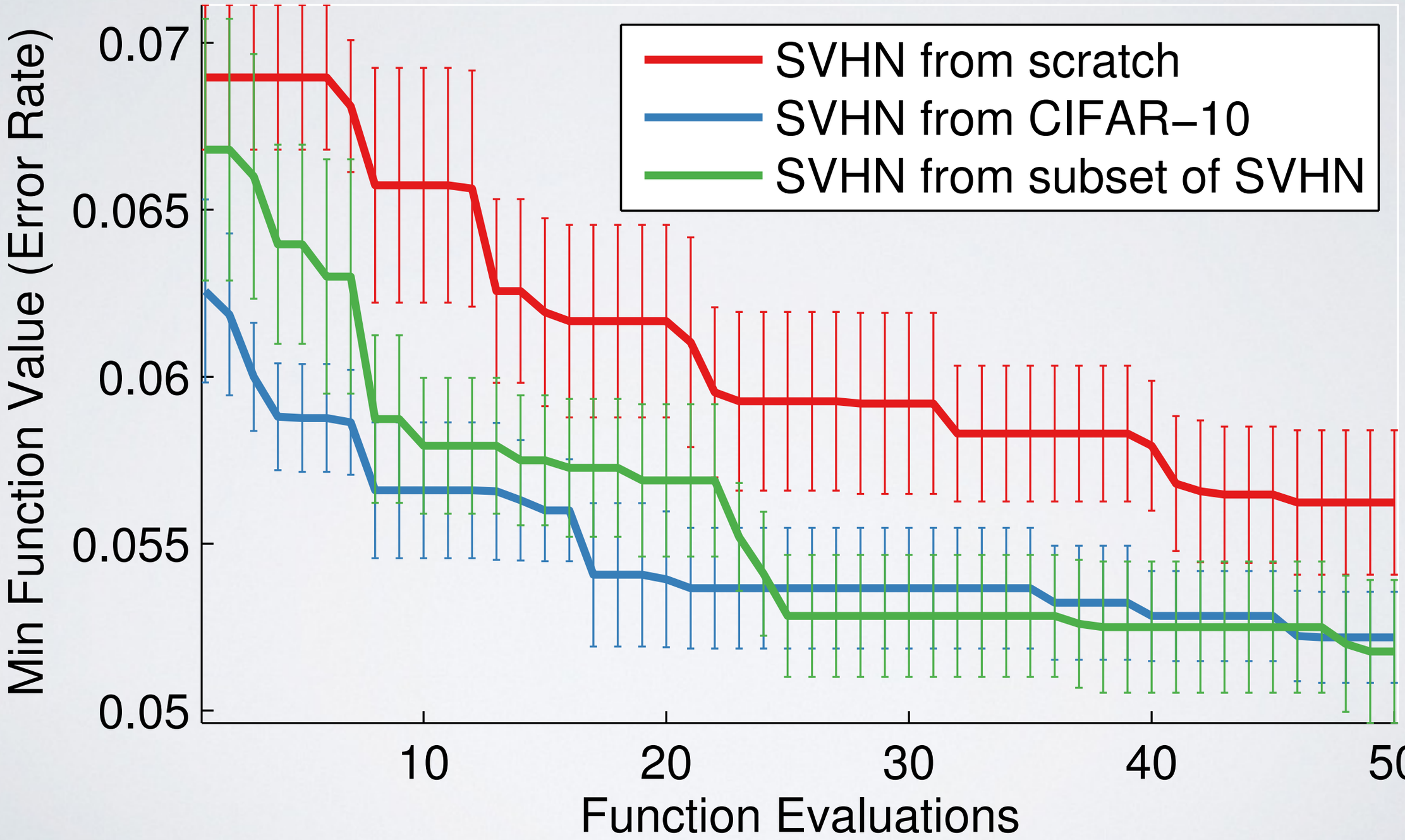


Swersky, Snoek & RPA, NIPS 2013

Parallel-Task Bayesian Optimization

Recognizing house numbers, leveraging object recognition

Achieves 4.77% test error -- best non-dropout result.

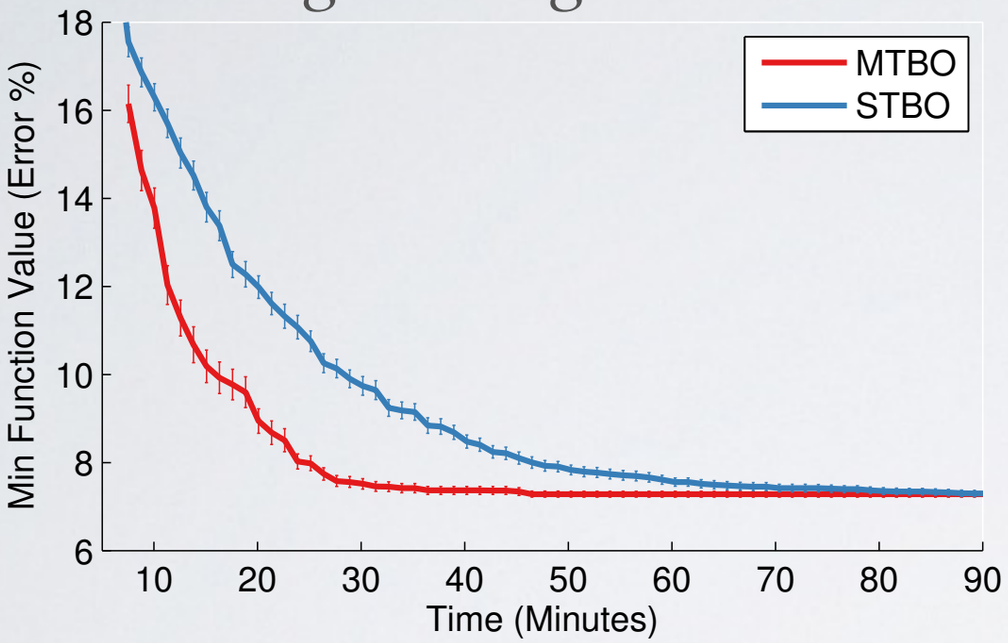


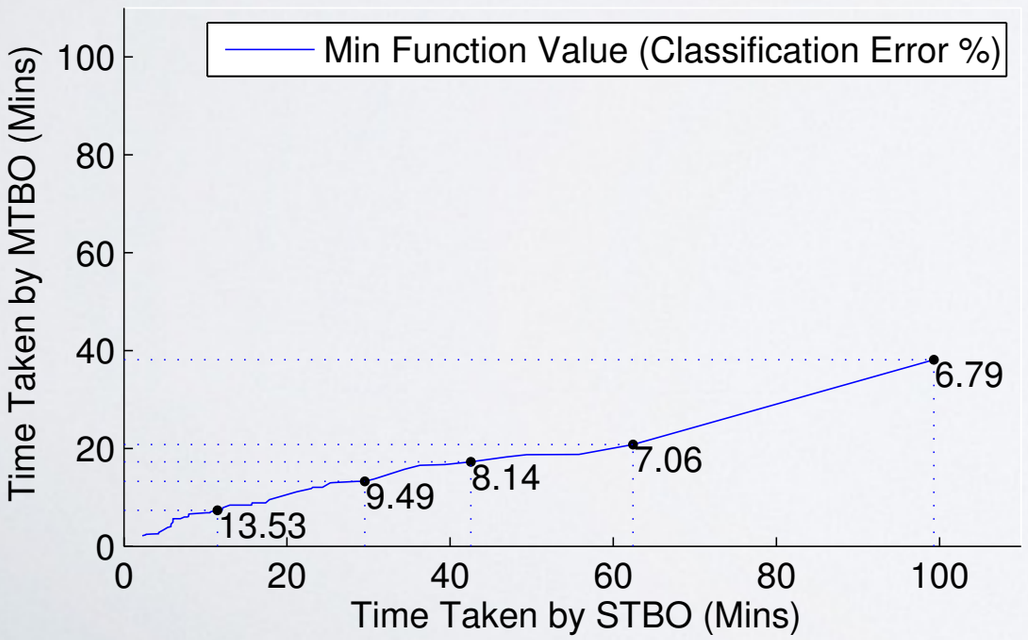
Swersky, Snoek & RPA, NIPS 2013

Auxiliary-Task Bayesian Optimization

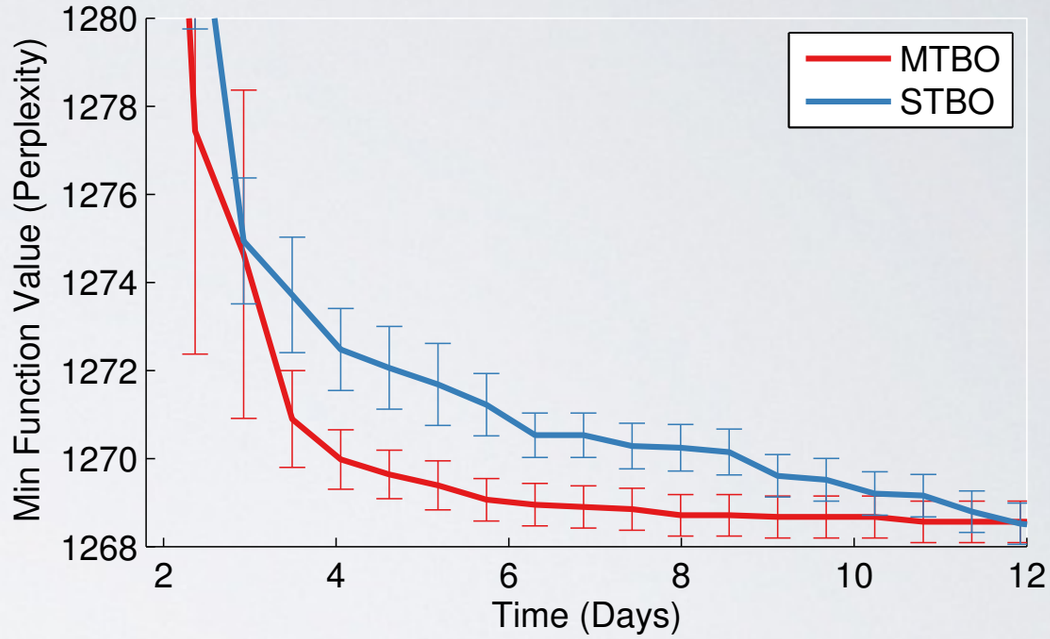
Using a smaller problem to go fast on a bigger problem.

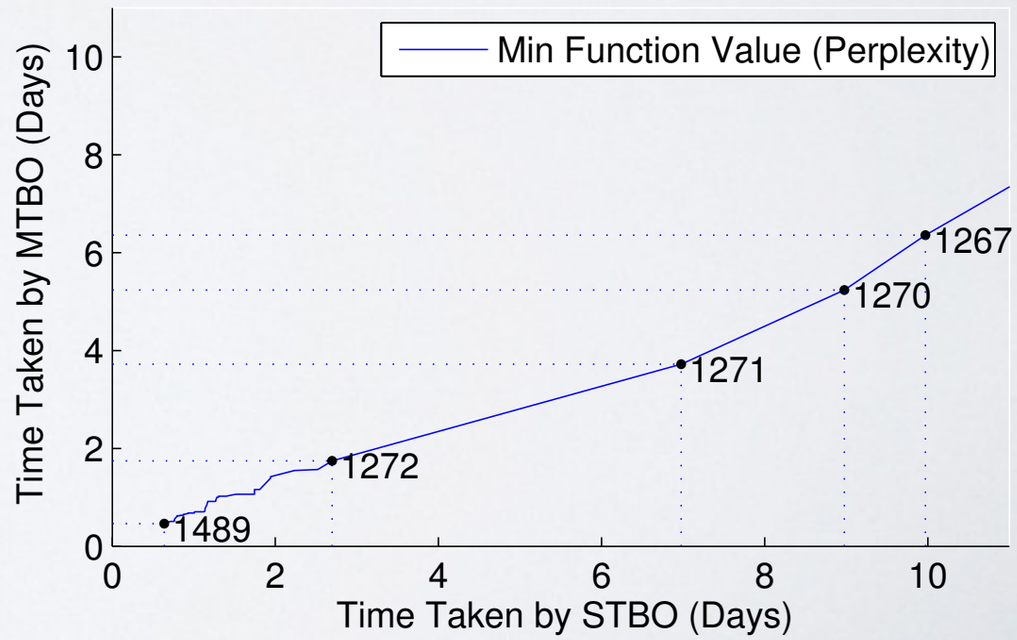
Digit Recognition





Online Latent Dirichlet Allocation





Swersky, Snoek & RPA, NIPS 2013

Bayesian Optimization with Unknown Constraints

- In general, you need to specify a range for each parameter you wish to optimize.
- However, there may be regimes in which things break, that are not easily pre-specified.
- Alternatively, you may have expensive constraints that you want to make optimal decisions about.

$$\min_{\mathbf{x}} f(\mathbf{x}) \text{ s.t. } \forall k \; g_k(\mathbf{x}) \geq 0$$

objective function

constraint functions

Gelbart, Snoek & RPA, UAI 2014

Noisy Constraints

- Realistically, lots of problems we care about have noisy constraints just like they have noisy objectives.
- This is a problem because we can never be completely sure that we have a valid solution.
- This can be addressed via probabilistic constraints:

$$\min_{\mathbf{x}} \mathbb{E}[f(\mathbf{x})] \text{ s.t. } \forall k \ \Pr[g_k(\mathbf{x}) \geq 0] \geq 1 - \delta_k$$

objective function

constraint functions

confidence threshold

Gelbart, Snoek & RPA, UAI 2014

Decoupled Constraints

- Just like in the multi-task case, it may be possible to evaluate the constraints separately from the objective.

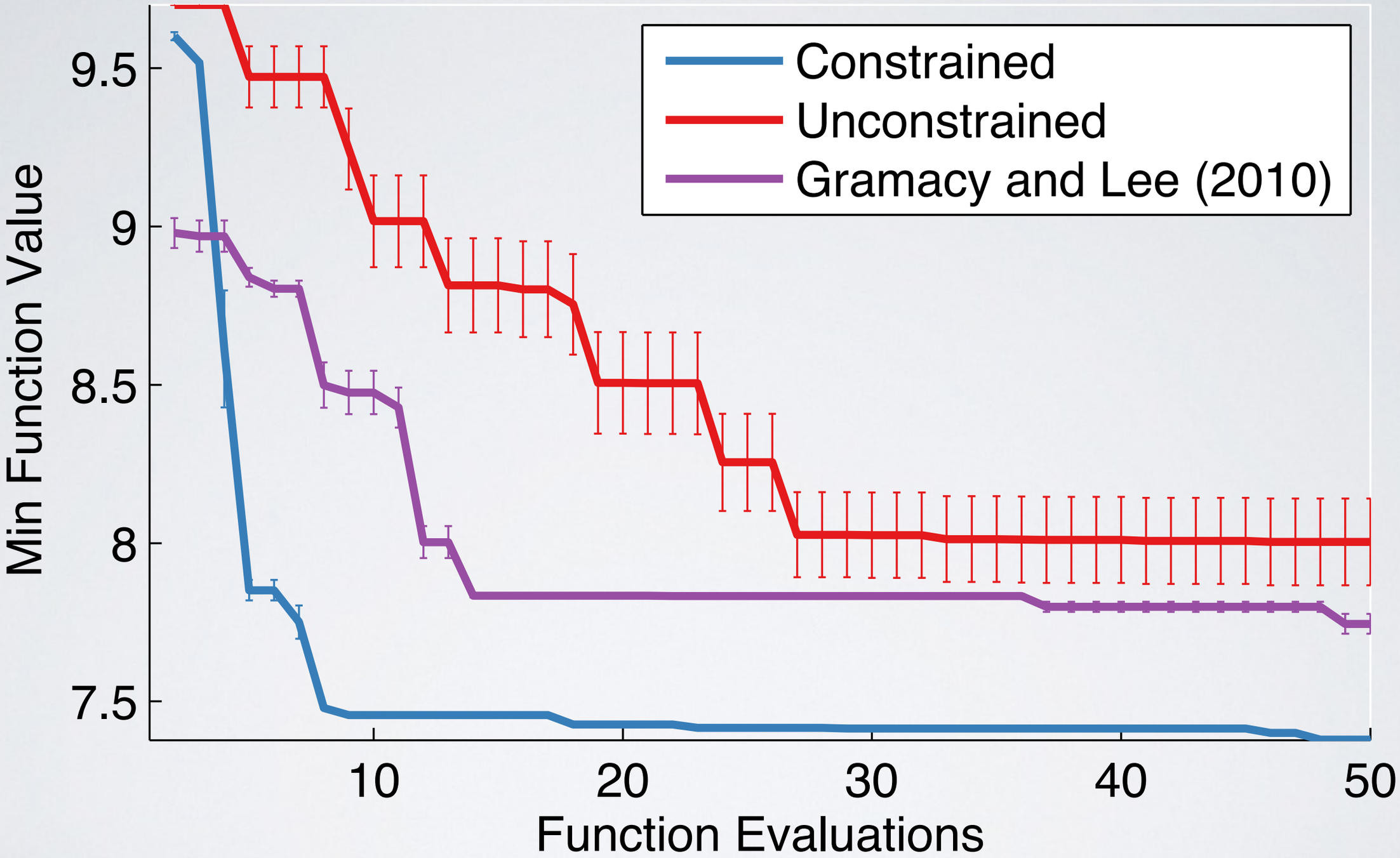
- The constraints may have vastly different costs.

- We call these *decoupled* constraints.

- We use entropy search to determine whether to evaluate the constraint or the objective.

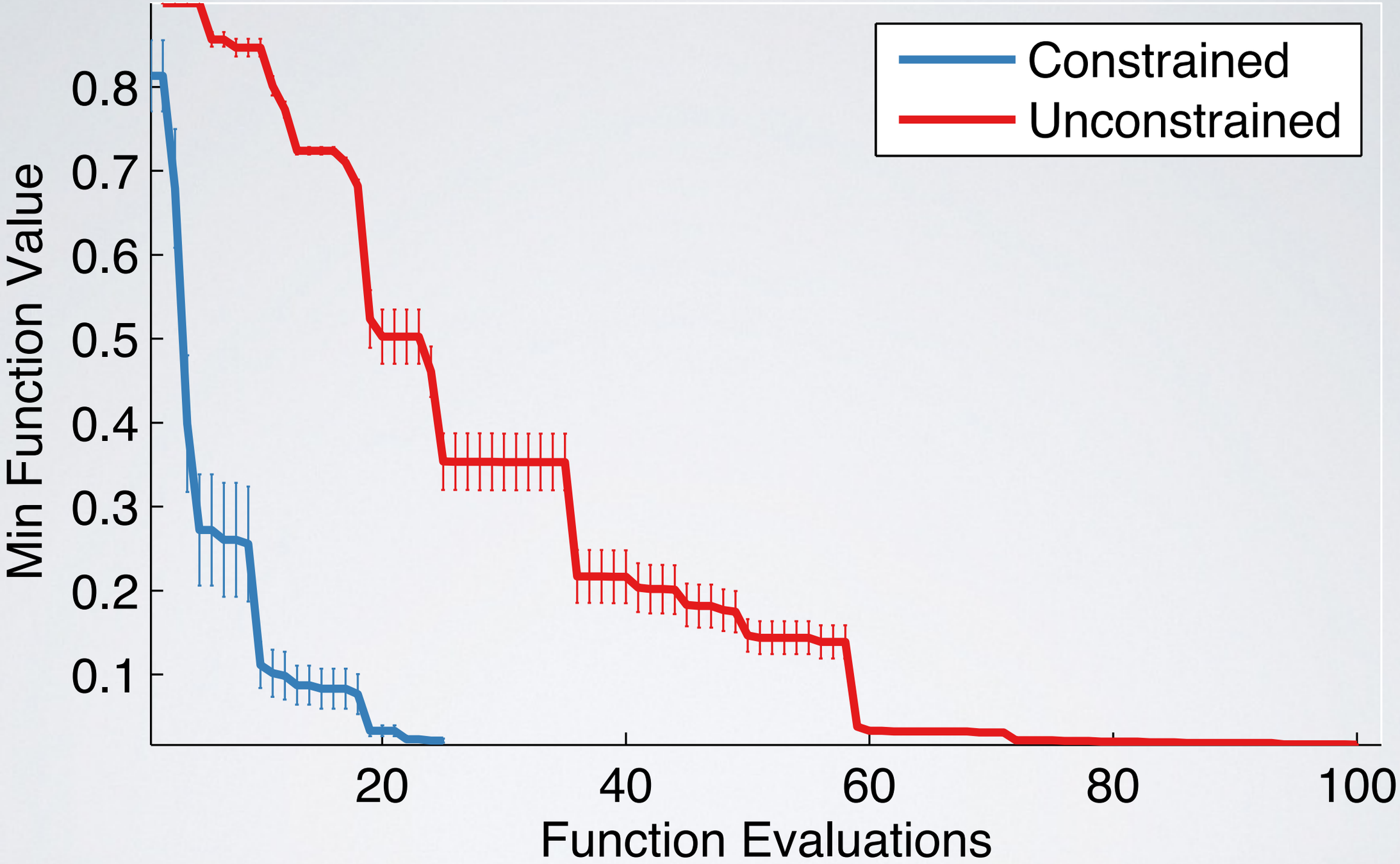
Gelbart, Snoek & RPA, UAI 2014

LDA with Sparsity Constraints



Require that online LDA result in low-entropy topics.

Neural Network with Memory Constraints



Decoupled: find a good neural network for MNIST, subject to stringent memory limitations.

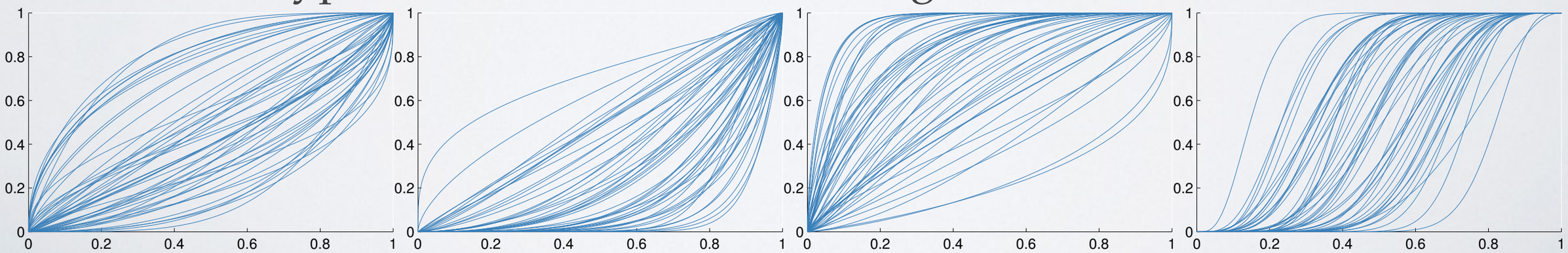
Bayesian Optimization with Random Projections

- Bayesian optimization relies on having a good prior on functions — challenging in high dimensions!
- Realistically, many problems have low effective dimensionality, i.e., the objective is primarily determined by a manifold in the input space.
- Wang, Zoghi, Hutter, Matheson & de Freitas, IJCAI 2013 suggest a way to deal with many dimensions.
- Approach: choose a random embedding of the input dimensions and then optimize in the subspace.
- Straightforward to implement and offers some progress for large and difficult BayesOpt problems.

Input Warping for Nonstationary Objectives

- Good priors are key to Bayesian optimization.
- Almost all widely-used kernels are stationary, i.e., covariance depends only on the distance.
- For machine learning hyperparameters, this is a big deal; learning rate changes may have big effects in one regime and small ones in another.
- Real problems are highly nonstationary, but it is difficult to strike a balance between flexibility and tractability.

- One approach: extend the kernel to include a parametric bijection of each input, and integrate it out with MCMC.
- The CDF of beta and Kumaraswamy distributions are suitable bijections on the unit hypercube and allow interesting nonlinear effects.

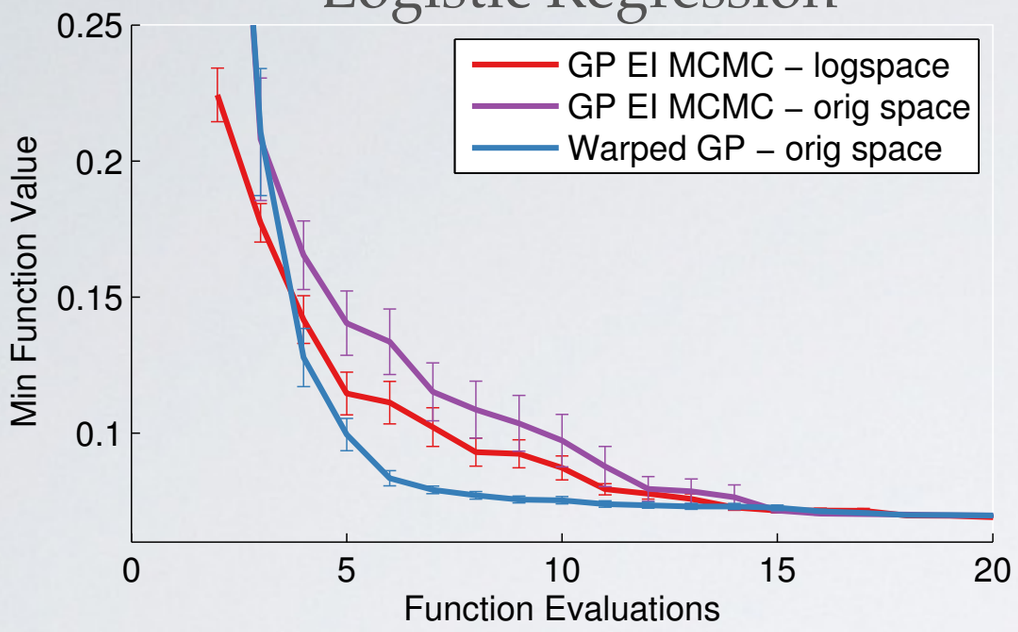


(a) Linear (b) Exponential (c) Logarithmic (d) Sigmoidal

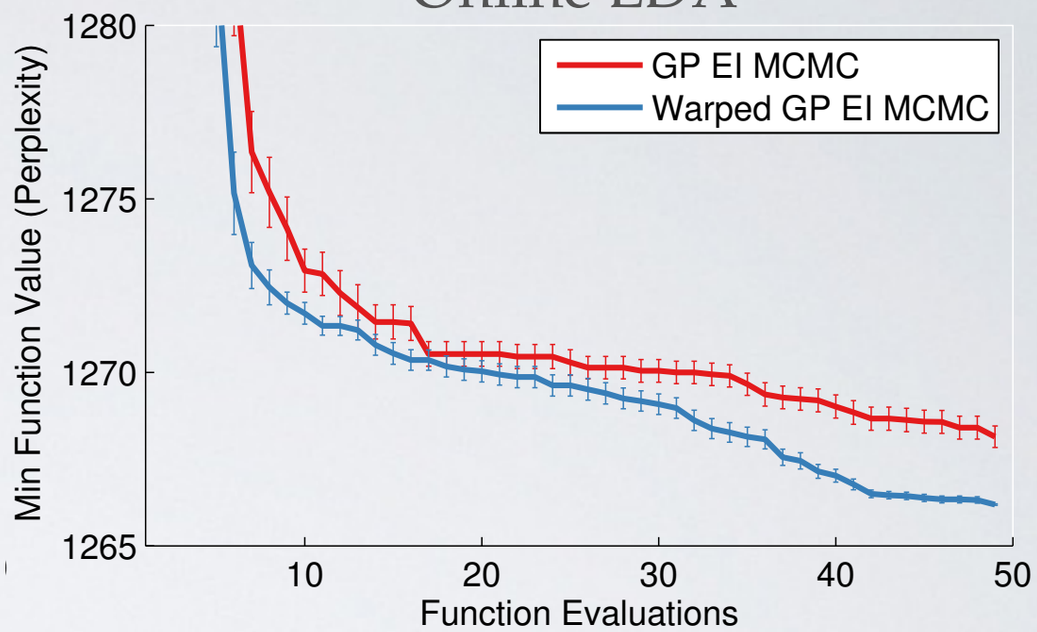
Snoek, Swersky, Zemel & RPA, ICML 2014

Input Warping for Nonstationary Objectives

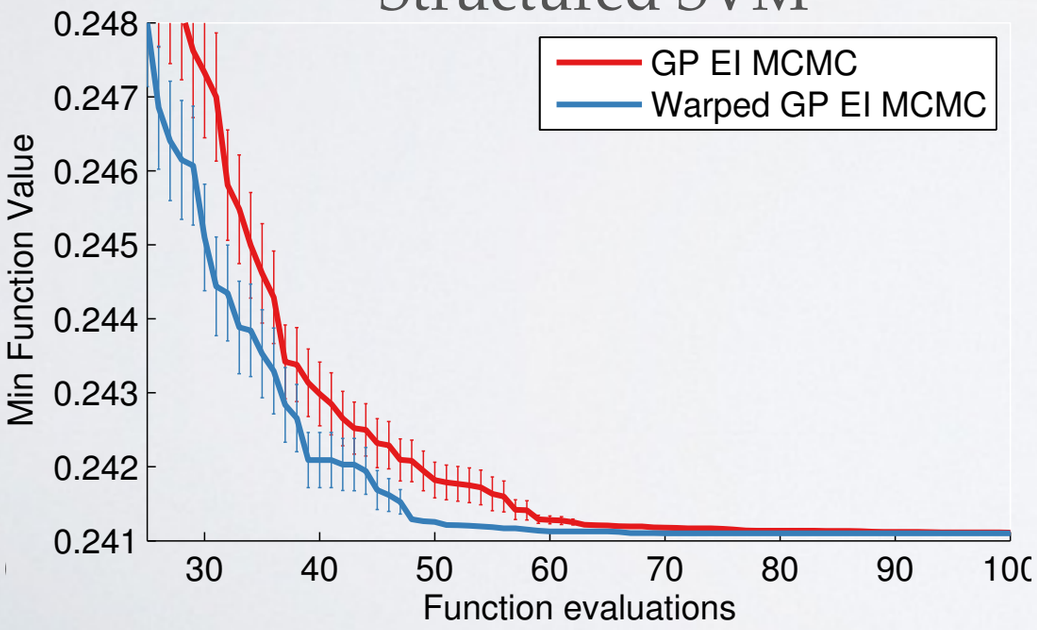
Logistic Regression



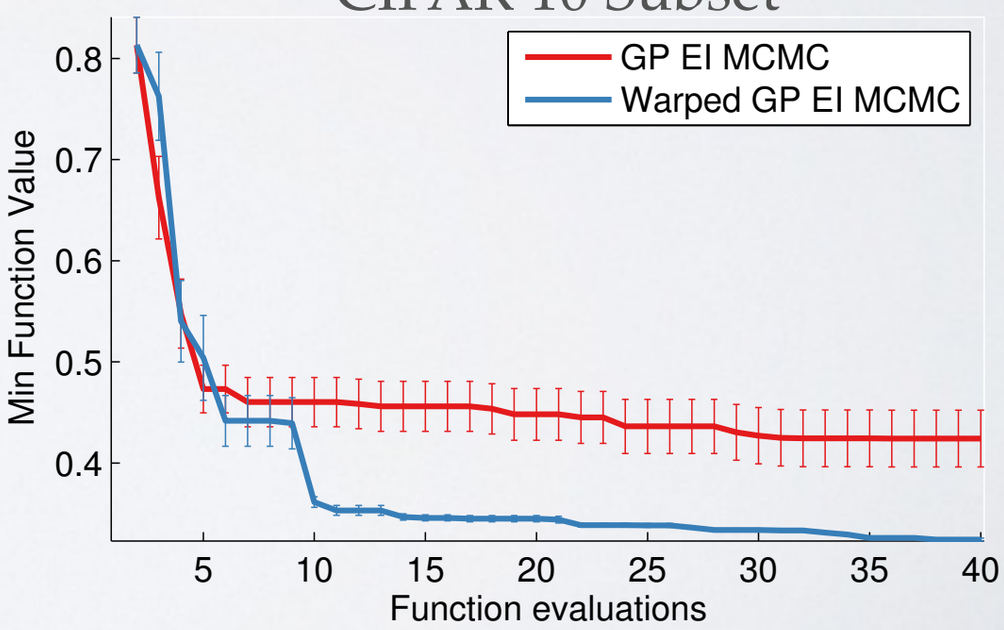
Online LDA



Structured SVM

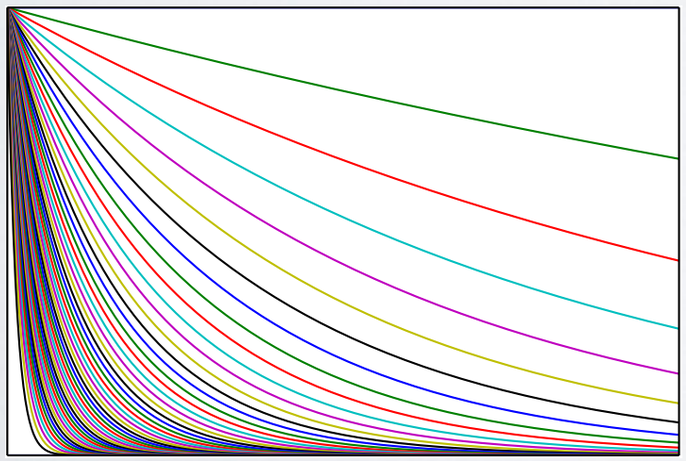


CIFAR 10 Subset

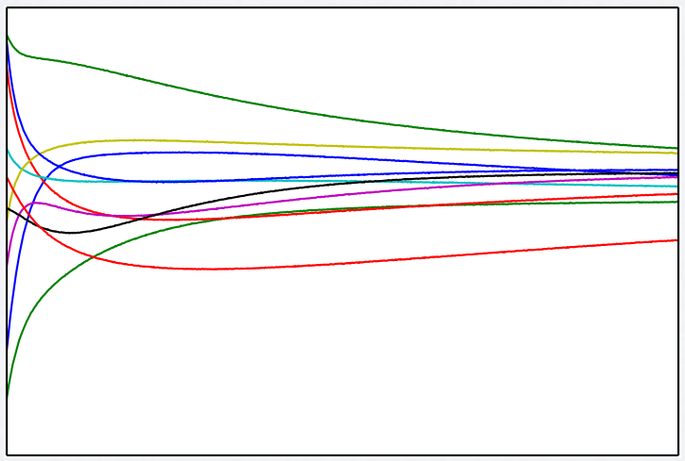


Freeze-Thaw Bayesian Optimization

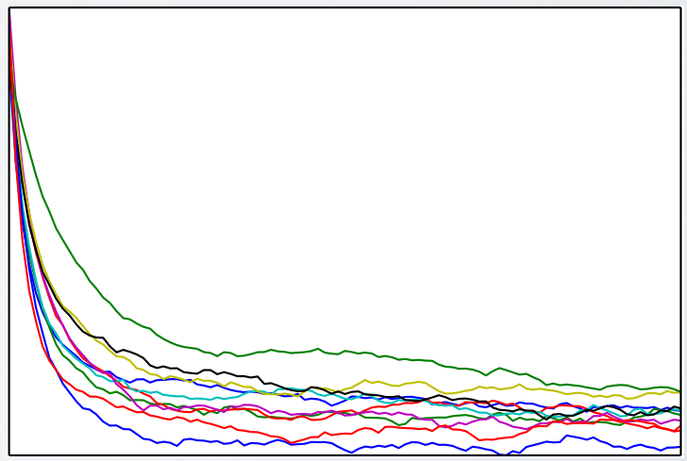
- For ML BayesOpt, the objective itself is often the result of an incremental training procedure, e.g., SGD.
- Humans often abort jobs that aren't promising.
- *Freeze-Thaw* Bayesian optimization tries to do the same thing.
- It tries to project when a training procedure will give a good value, and only continues promising ones.
- Requires a prior on training curves as a function of input hyperparameters. Introduces a new kernel for this.



(a) Exponential Decay Basis



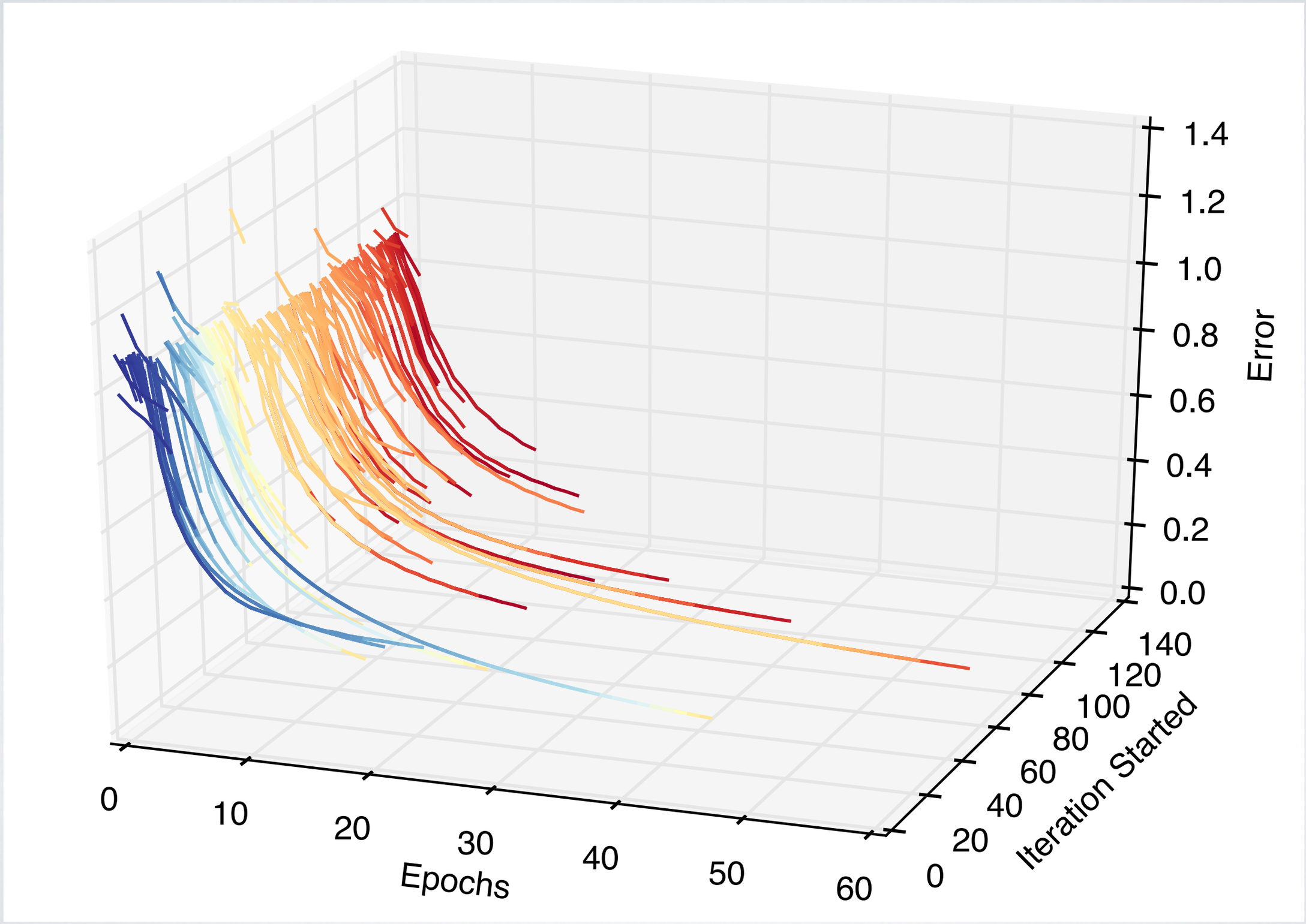
(b) Samples



(c) Training Curve Samples

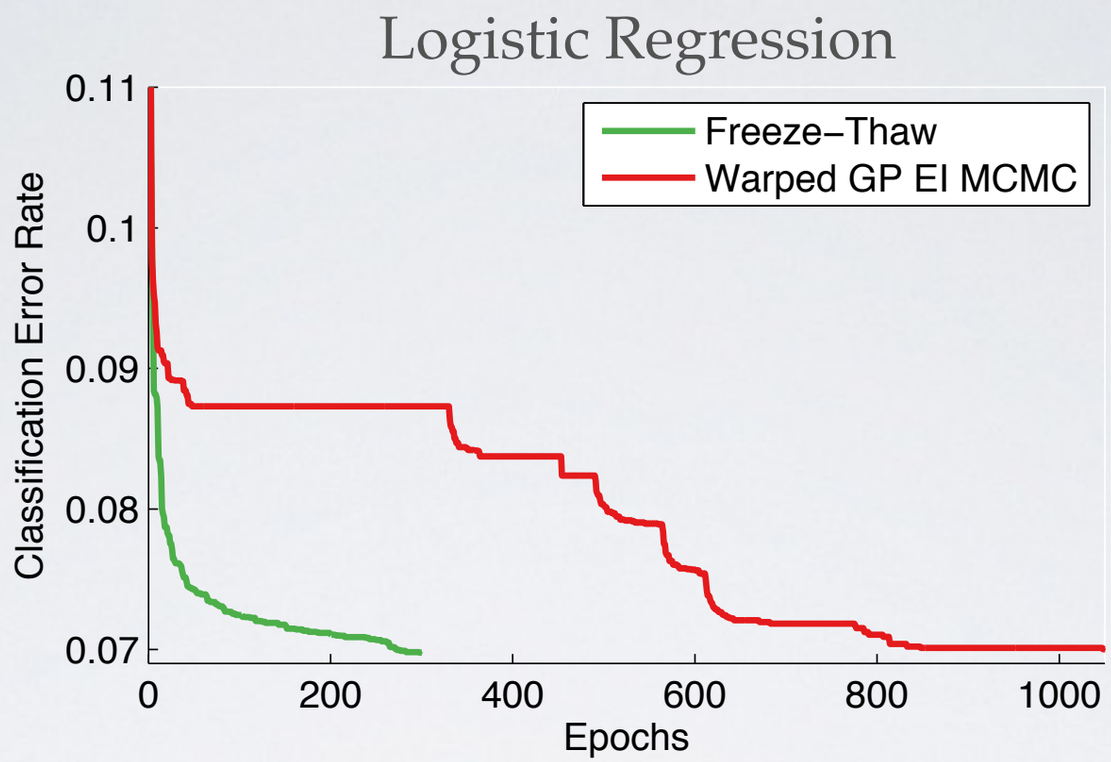
Swersky, Snoek & RPA, Under Review

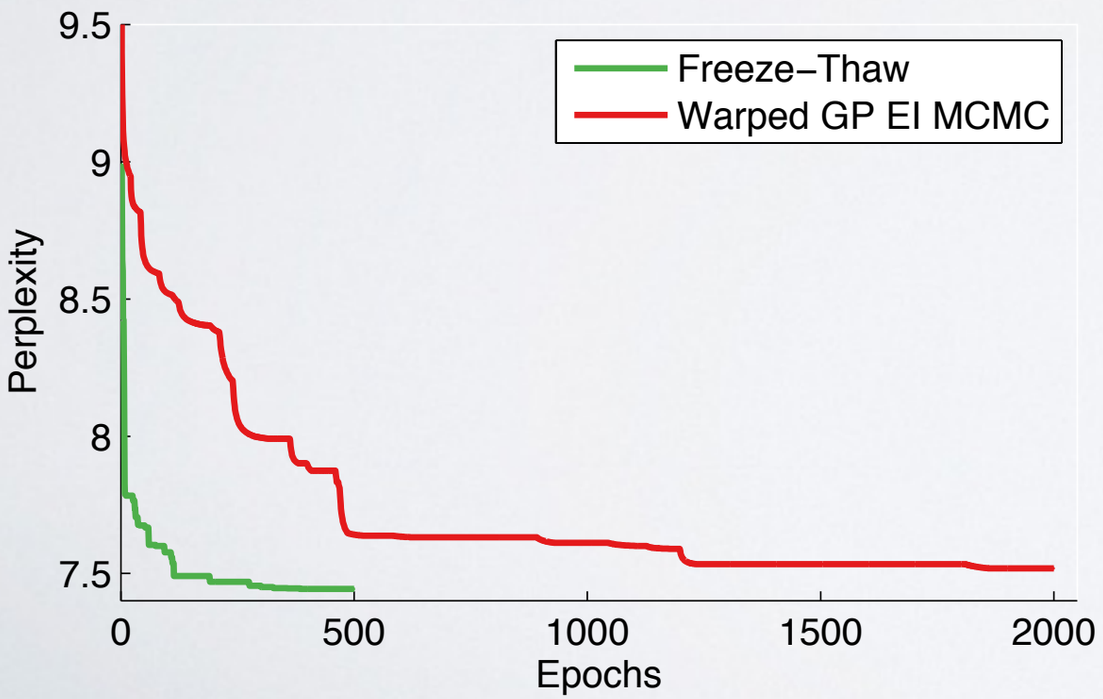
Freeze-Thaw Bayesian Optimization

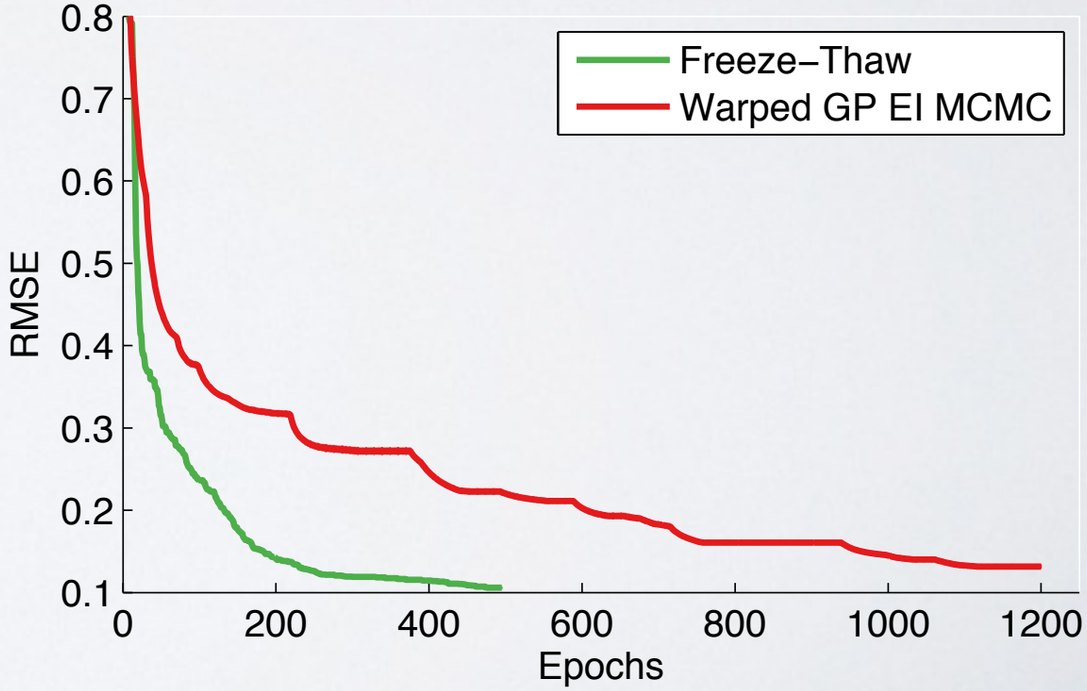


Swersky, Snoek & RPA, Under Review

Freeze-Thaw Bayesian Optimization



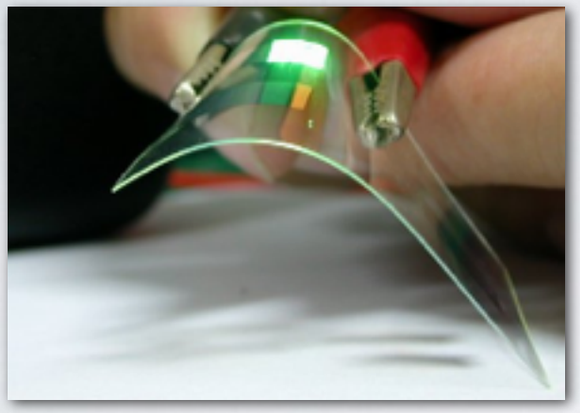




Swersky, Snoek & RPA, Under Review

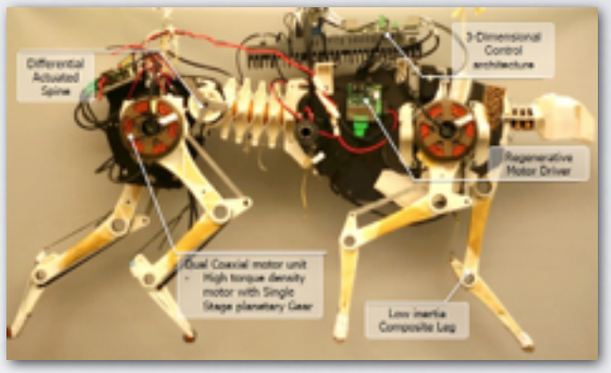
New Directions for Bayesian Optimization

SAMSUNG



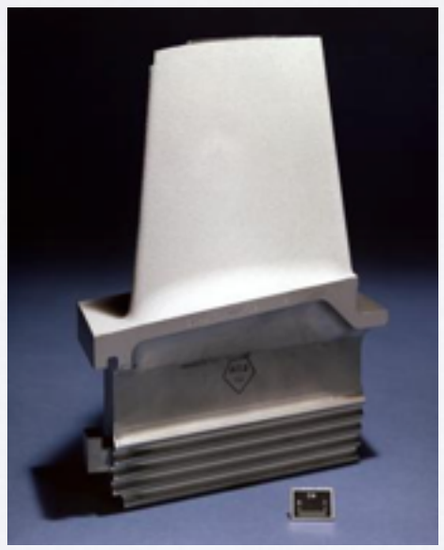
Finding new organic materials

Massachusetts Institute of Technology



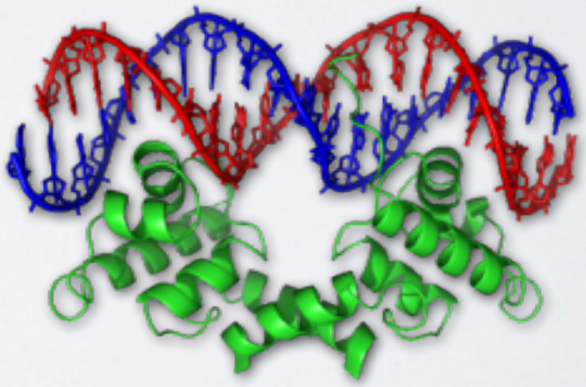
Optimizing robot control systems

Massachusetts Institute of Technology



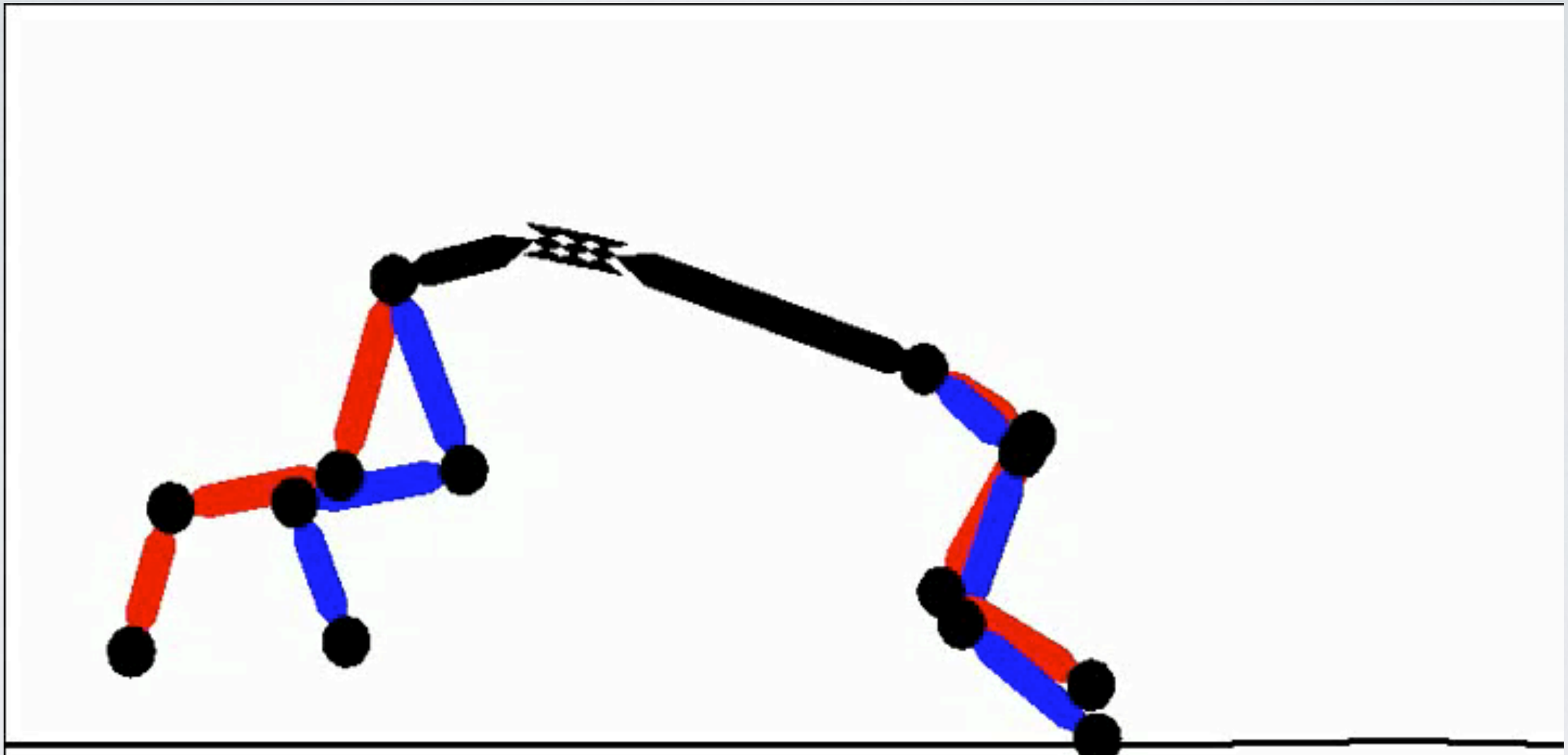
Improving turbine blade design

BROAD INSTITUTE



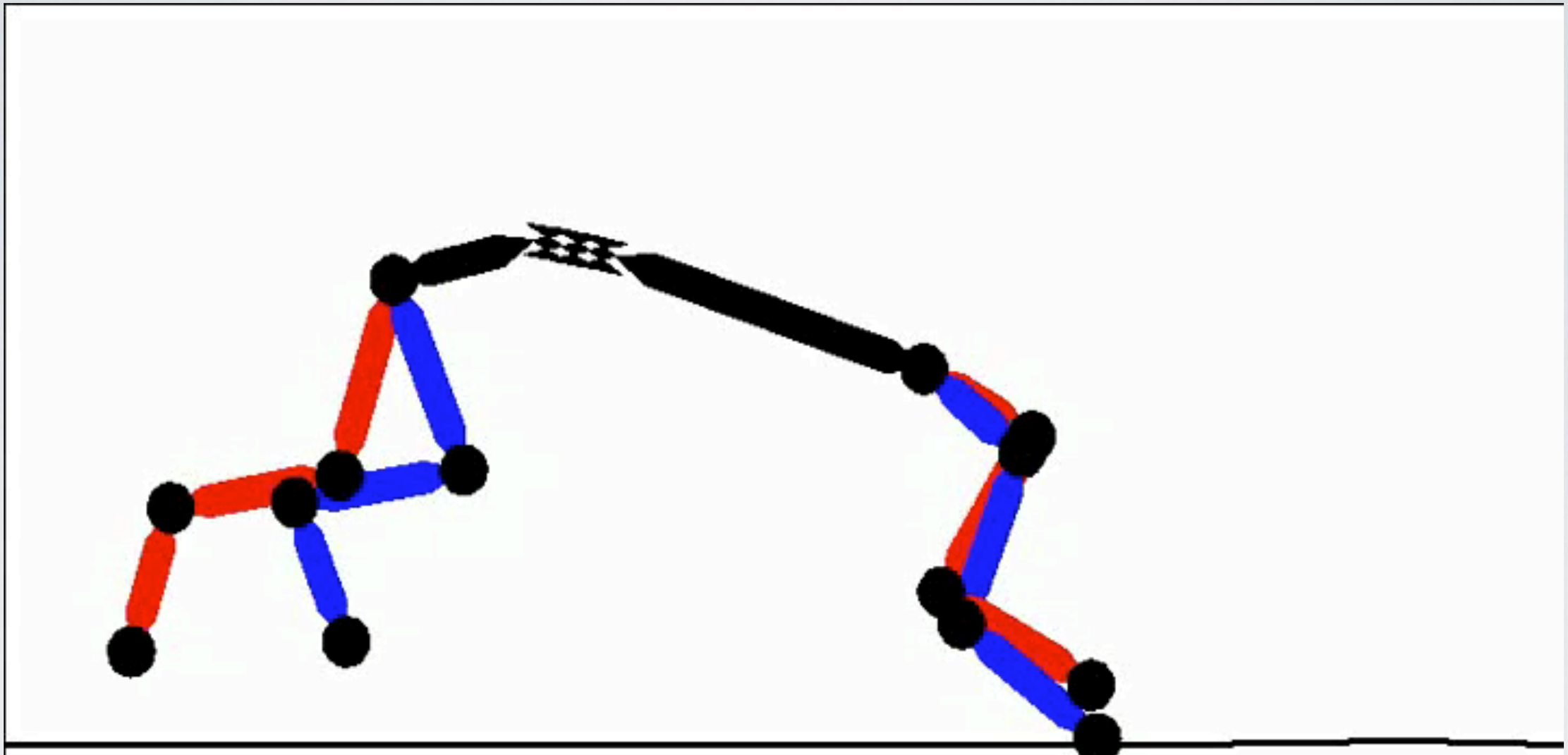
Designing DNA for protein affinities

Cheetah Gait - Sangbae Kim's Lab @ MIT



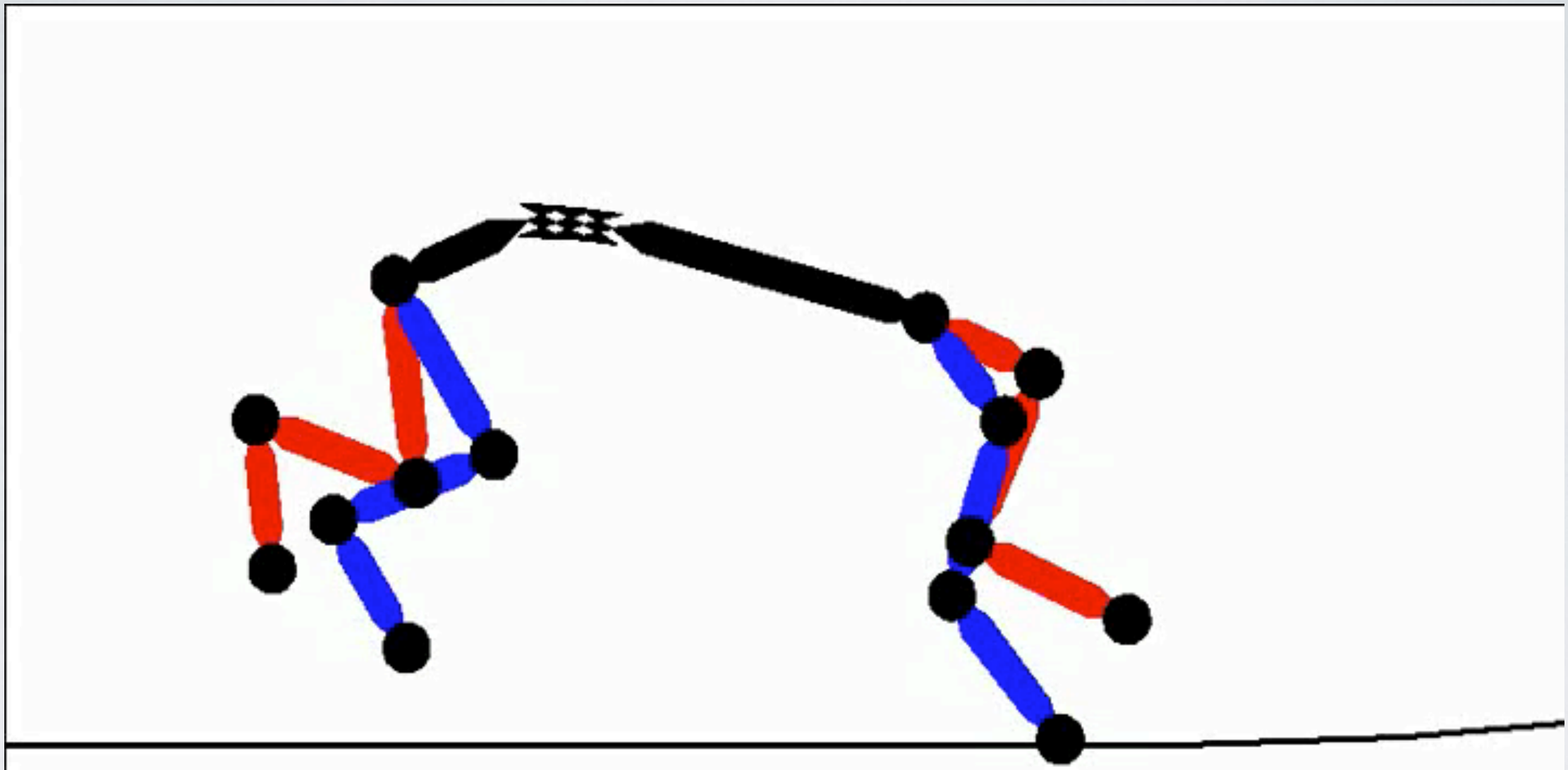
Human-Tuned Gait

Cheetah Gait - Sangbae Kim's Lab @ MIT



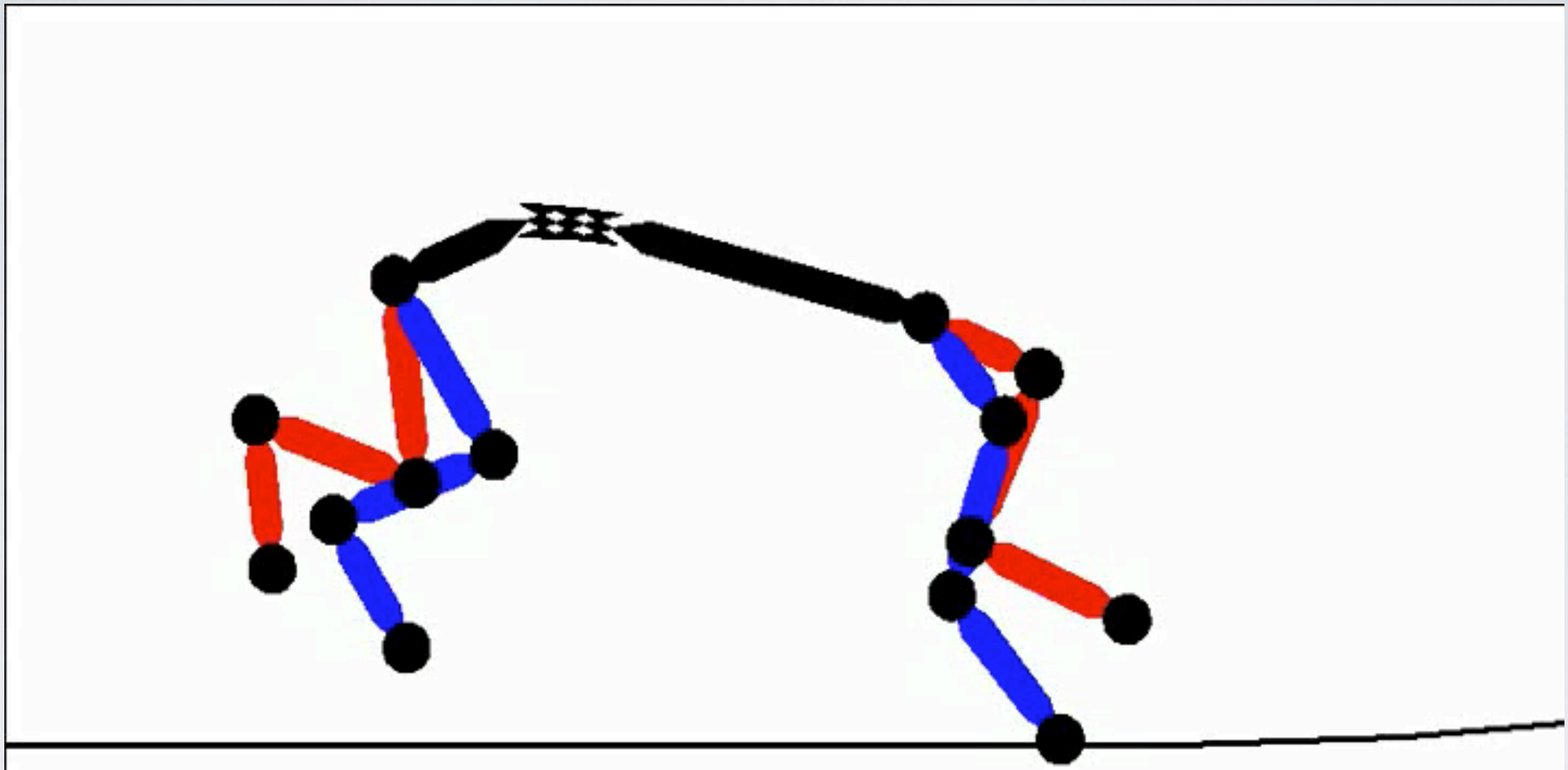
Human-Tuned Gait

Cheetah Gait - Sangbae Kim's Lab @ MIT



Bayesian Optimized Gait

Cheetah Gait - Sangbae Kim's Lab @ MIT



Bayesian Optimized Gait

"I just want to use Bayesian optimization."

- The tool my group maintains for performing Bayesian optimization is called Spearmint and is available at `https://github.com/HIPS/Spearmint`
- Spearmint is available for non-commercial use, and has most of the fancy research bits I've described today.
- For turn-key super-easy Bayesian optimization, we're launching a startup at `whetlab.com`. It provides a straightforward REST API and clients in Python, Matlab, R, etc.
- Our intention is to make Whetlab cheap/free for ML researchers.

Shameless Plug: Whetlab Python Client

```
import whetlab

# Define parameters to optimize
parameters = { ‘learn_rate'   : {'type':'float','min':0,'max':15,'size':1},
               ‘weight_decay’ : {‘type’:'float','min':-5,'max':10,'size':1},
                   . . .              . . .                  . . .
               ‘layer1_size'  : {'type':'integer', 'min':1, 'max':1000, ‘size':1}}

name = ‘My Fancy Deep Network'
description = 'Optimize the hyperparams of my network.’
outcome = {‘name’:’Cross-Validation Accuracy', 'type':'float'}
scientist = whetlab.Experiment(name=name, description=description,
                               parameters=parameters, outcome=outcome)

# Loop until you’re happy.
# You can have multiple of these in parallel, too.
for ii in xrange(1000):

    # Ask Whetlab scientist for a new experiment to try.
    job = scientist.suggest()

    # Perform experiment with your own code.
    . . .  do training and cross-validation here . . .

    # Inform Whetlab scientist about the outcome.
    scientist.update(job, outcome)
```

Another Shameless Plug: Kayak

- Simple open source Python AutoDiff library.
- Meant to be lightweight alternative to, e.g,. Theano.
- Available at https://github.com/HIPS/Kayak
- Not yet all that featureful.

```
import kayak
import numpy.random as np

# Create a batcher object.
batcher = kayak.Batcher(batch_size, inputs.shape[0])

# Specify inputs and targets.
X = kayak.Inputs(inputs, batcher)
T = kayak.Targets(targets, batcher)

# First-layer weights and biases, with random initializations.
W1 = kayak.Parameter( 0.1*npr.randn( inputs.shape[1], layer1_sz ))
B1 = kayak.Parameter( 0.1*npr.randn(1, layer1_sz) )

# First hidden layer: ReLU + Dropout
H1 = kayak.Dropout(kayak.HardReLU(kayak.ElemAdd(kayak.MatMult(X, W1), B1)), layer1_dropout)

 . . . various other bits in your architecture . . .

# Specify your loss function.
loss = kayak.MatSum(kayak.LogMultinomialLoss(Y, T))

# Gradients are so easy!
grad_W1 = loss.grad(W1)
grad_B1 = loss.grad(B1)
```

Thanks

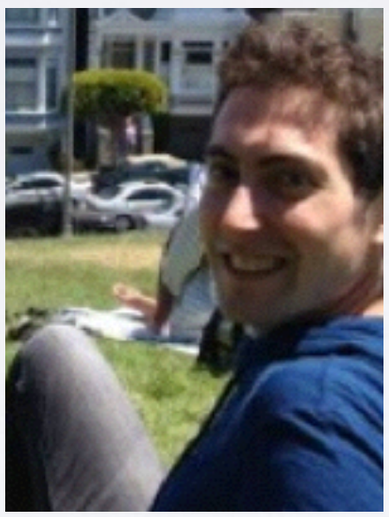
Coming soon at whetlab.com

Whetlab

DARPA

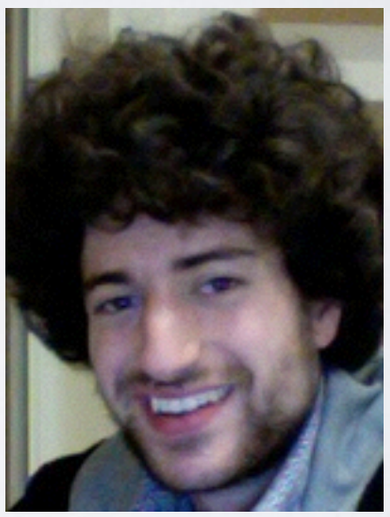
amazon.com

Jasper Snoek



Kevin Swersky

Hugo Larochelle



Michael Gelbart

Code: https://github.com/HIPS/Spearmint