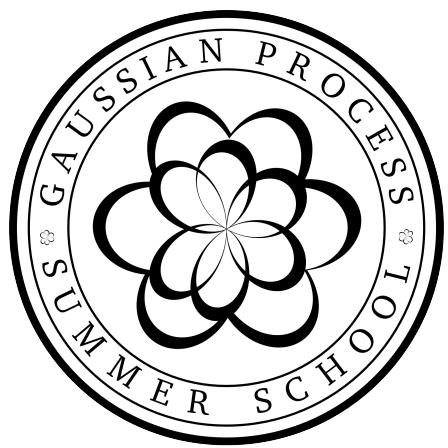
Gaussian Processes for Global Optimization

Javier González

GPSS - Sheffield, September 2015



Gaussian Processes

So far in the summer school...

GPs are scalable and flexible probabilistic models useful for regression, classification, etc.

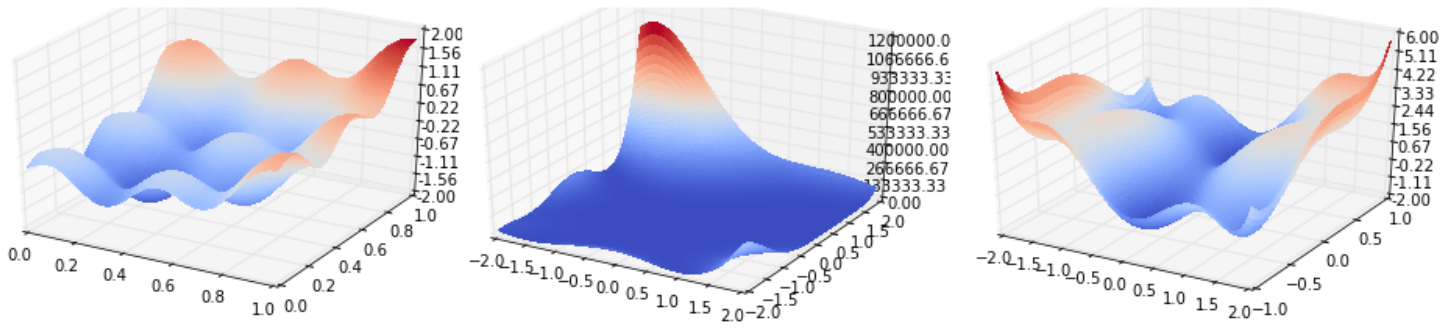
In this tutorial...

How can we use GPs to solve global optimization problems?

Global optimization

Consider a 'well behaved' function $f : \mathcal{X} \to \mathbb{R}$ where $\mathcal{X} \subseteq \mathbb{R}^D$ is a compact set.

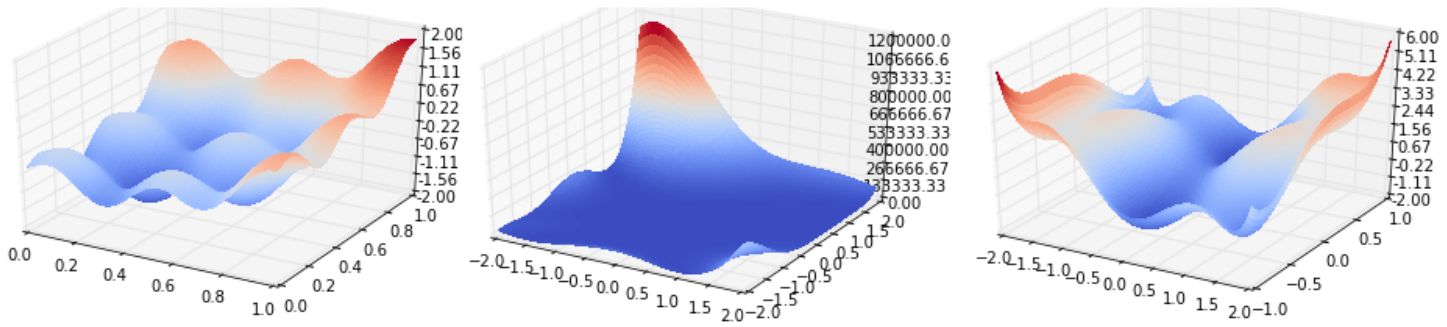
$$x_M = \arg\min_{x \in \mathcal{X}} f(x).$$



Global optimization

Consider a 'well behaved' function $f : \mathcal{X} \to \mathbb{R}$ where $\mathcal{X} \subseteq \mathbb{R}^D$ is a compact set.

$$x_M = \arg\min_{x \in \mathcal{X}} f(x).$$



- f is explicitly unknown and multimodal.
- Evaluations of f may be perturbed.
- Evaluations of f are expensive.

What to do?

- Gradient and Hessian are not computable.

Grid search?

If f is L-Lipschitz continuous and we are in a noise-free domain to guarantee that we propose some $\mathbf{x}_{M,n}$ such that

$$f(\mathbf{x}_M) - f(\mathbf{x}_{M,n}) \leq \epsilon$$

we need to evaluate f on a D-dimensional unit hypercube:

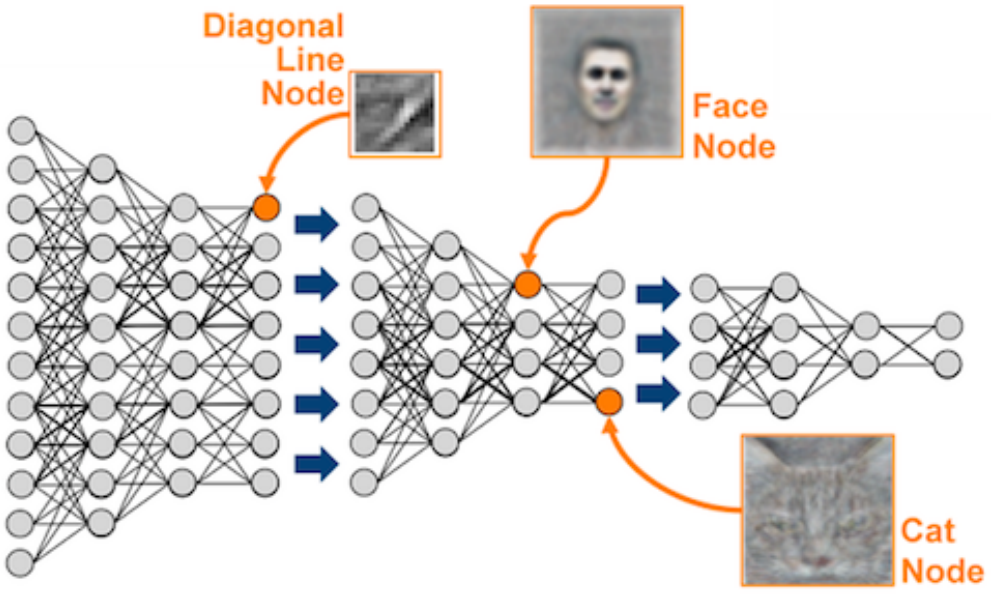
$$(L/\epsilon)^D \textit{evaluations}!$$

Example: $(10/0.01)^5 = 10e14...$

... but function evaluations are very expensive!

Expensive functions, who doesn't have one?

Parameter tuning in ML algorithms.

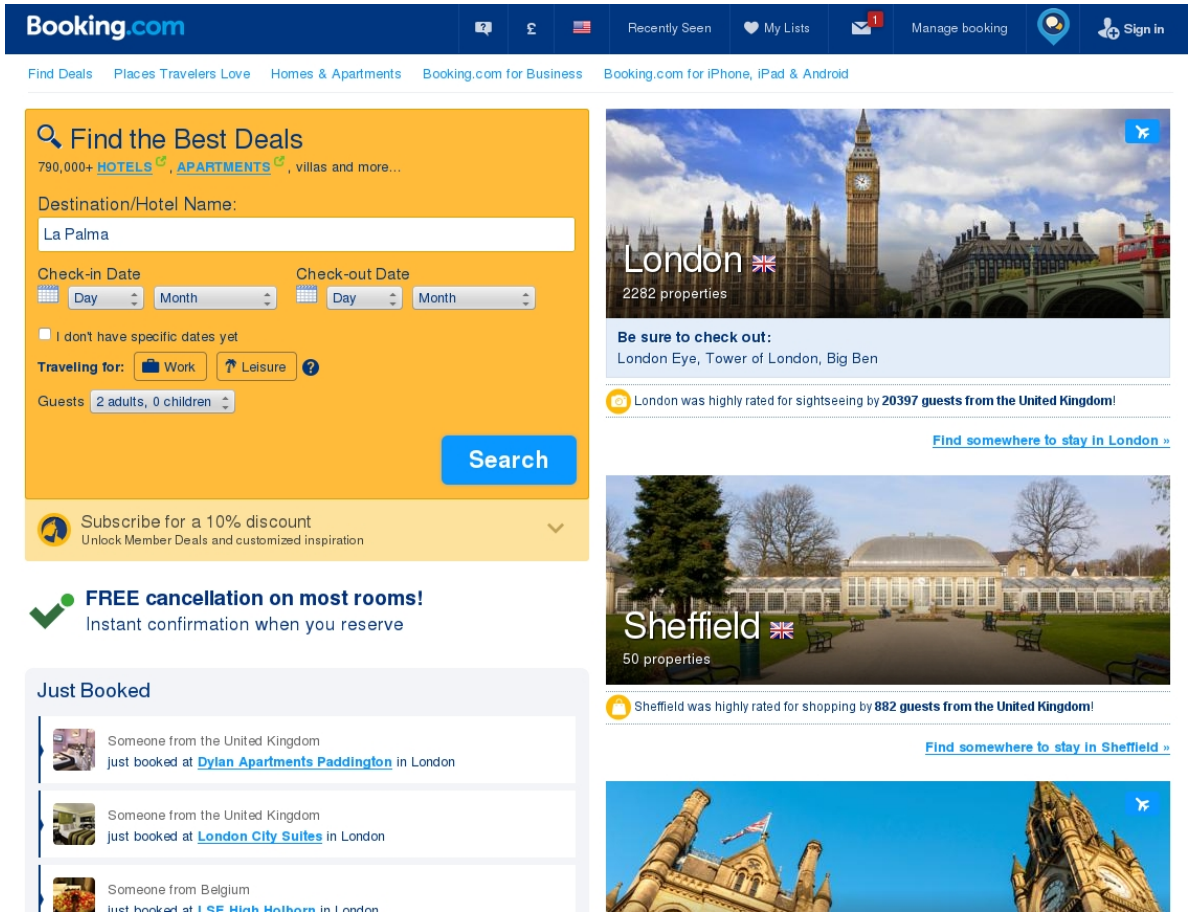


- Number of layers/units per layer
- Weight penalties
- Learning rates, etc.

Figure source: http://theanalyticsstore.com/deep-learning

Expensive functions, who doesn't have one?

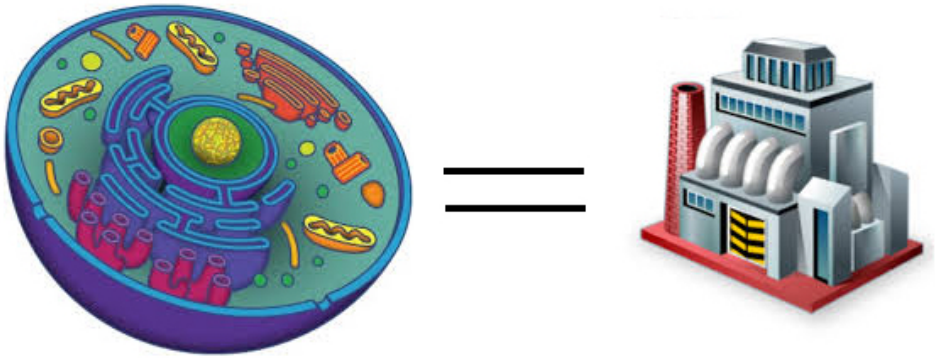
Tuning websites with A/B testing



Optimize the web design to maximize sign-ups, downloads, purchases, etc.

Expensive functions, who doesn't have one?

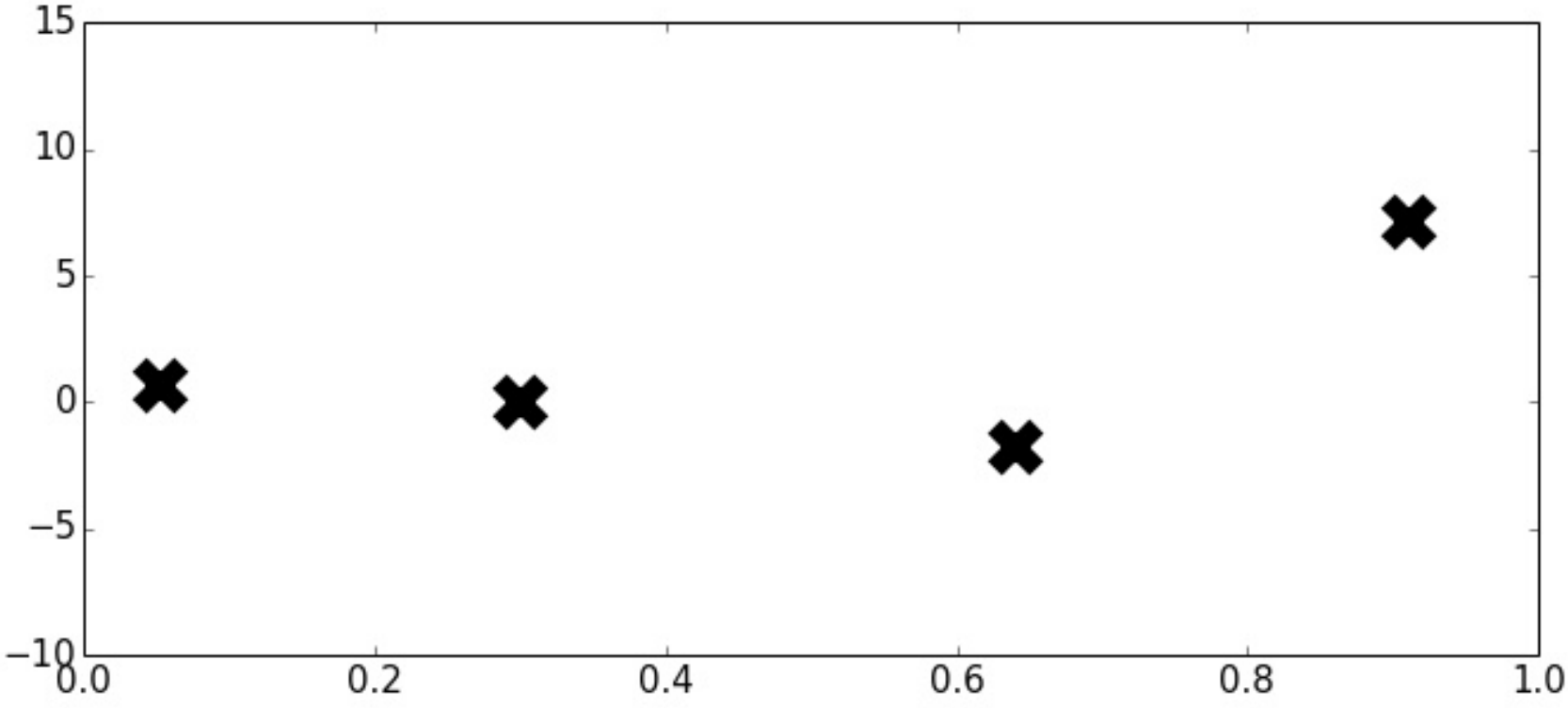
Synthetic gene design: Use mammalian cells to make protein products.



Optimize genes (ATTGGTUGA...) to best enable the cell-factory to operate most efficiently.

Typical situation

We have a few function evaluations

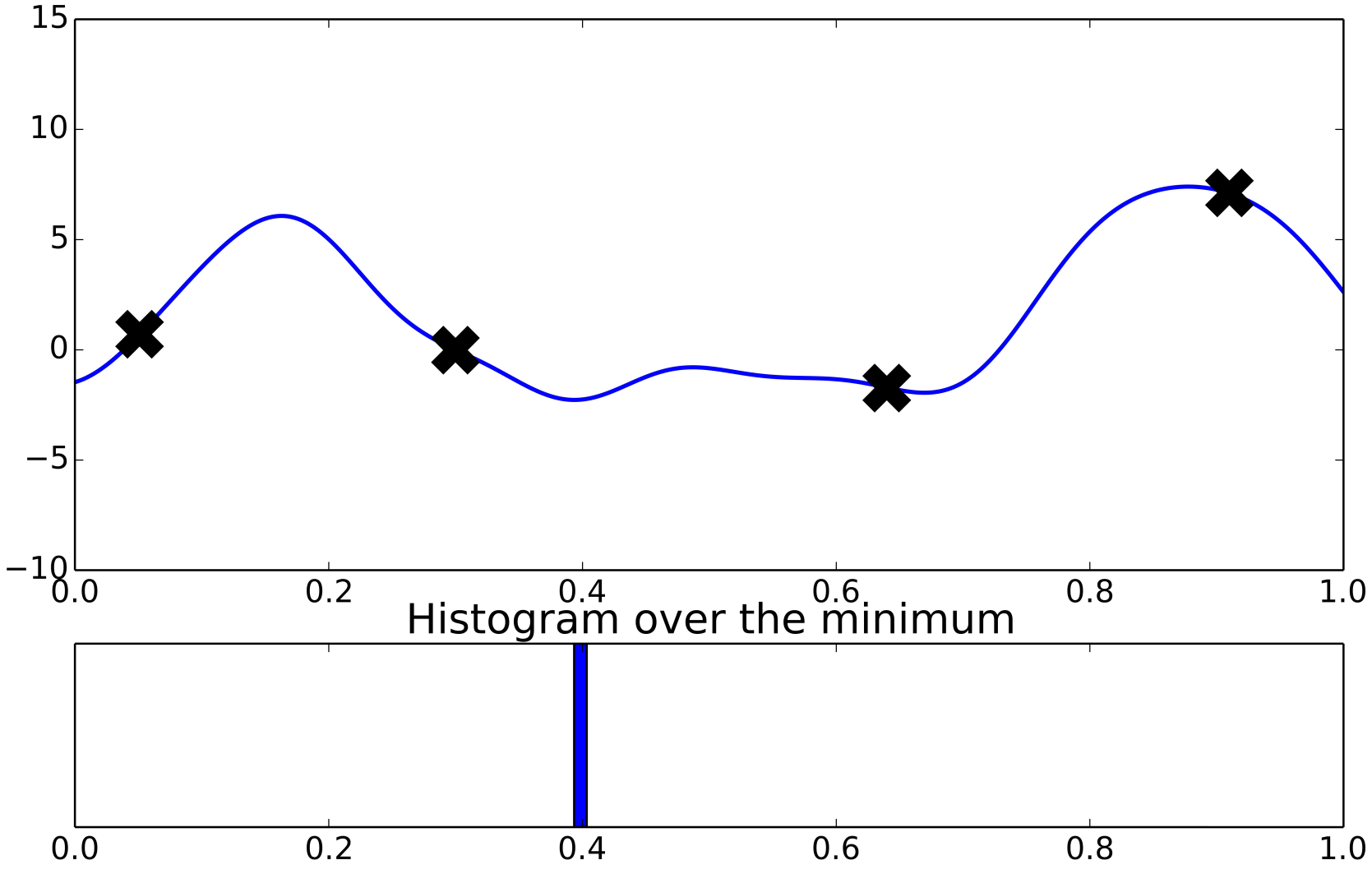


Where is the minimum of f?

Where should the take the next evaluation?

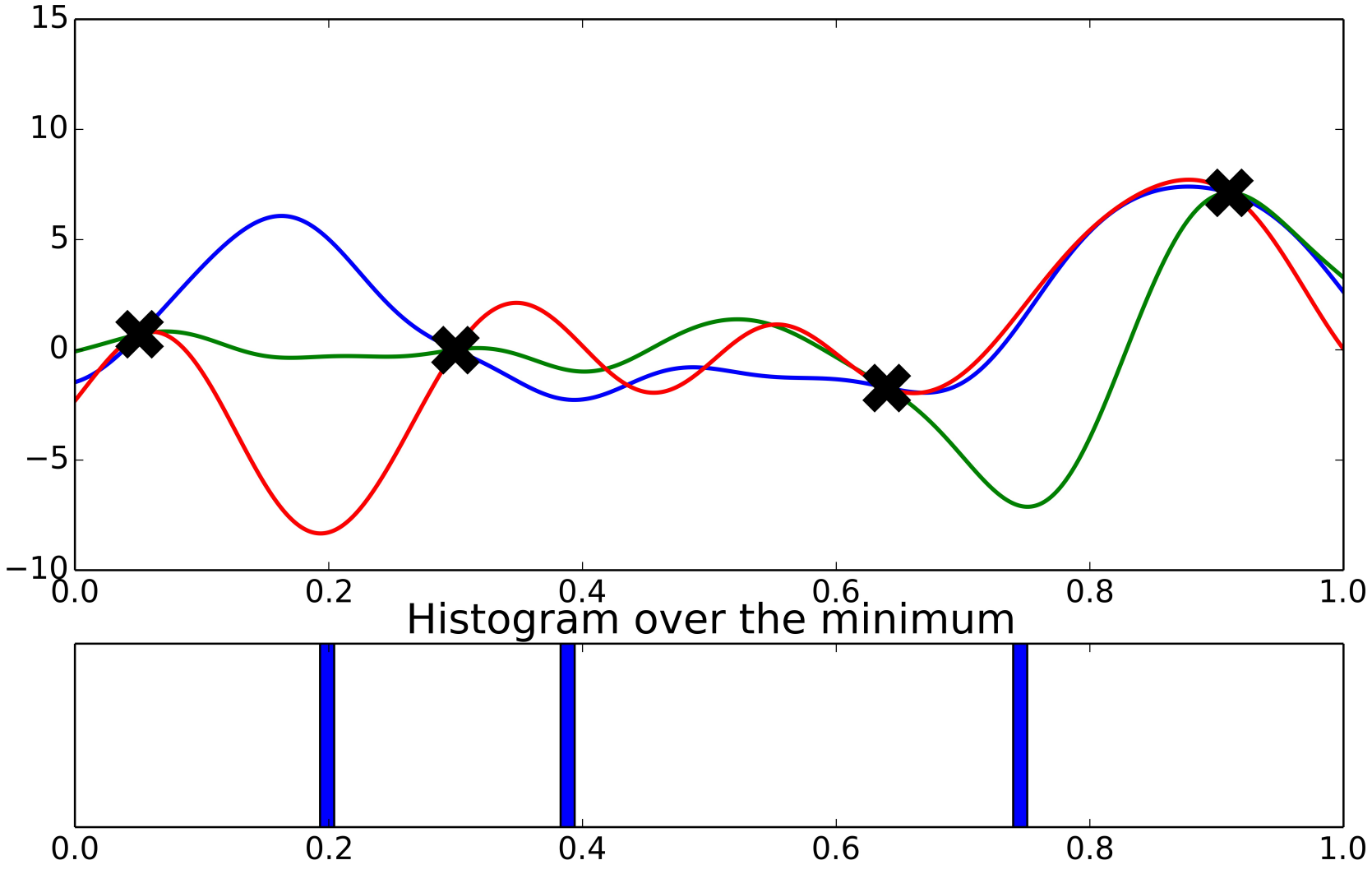
Intuitive solution

One curve



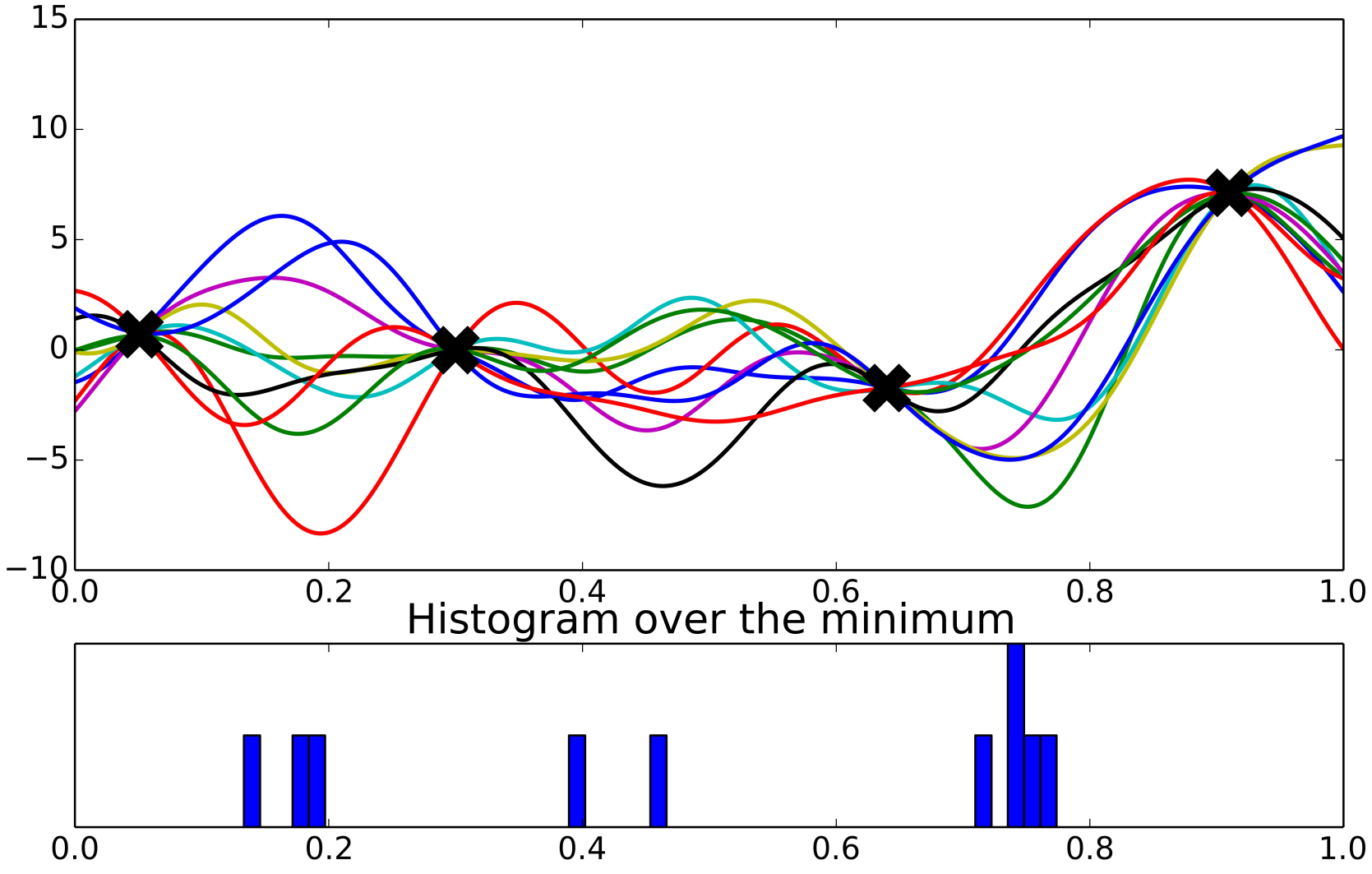
Intuitive solution

Three curves



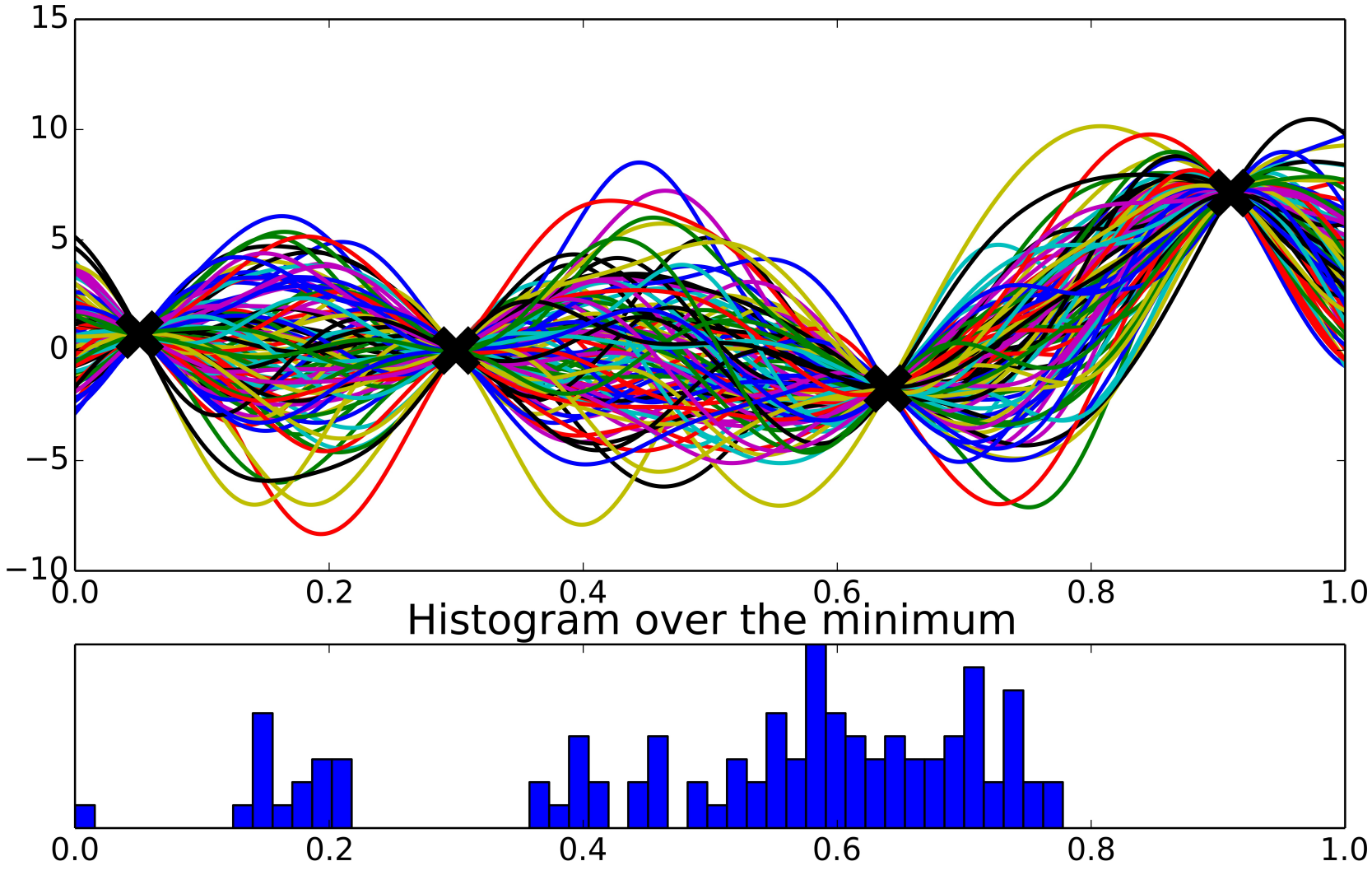
Intuitive solution

Ten curves



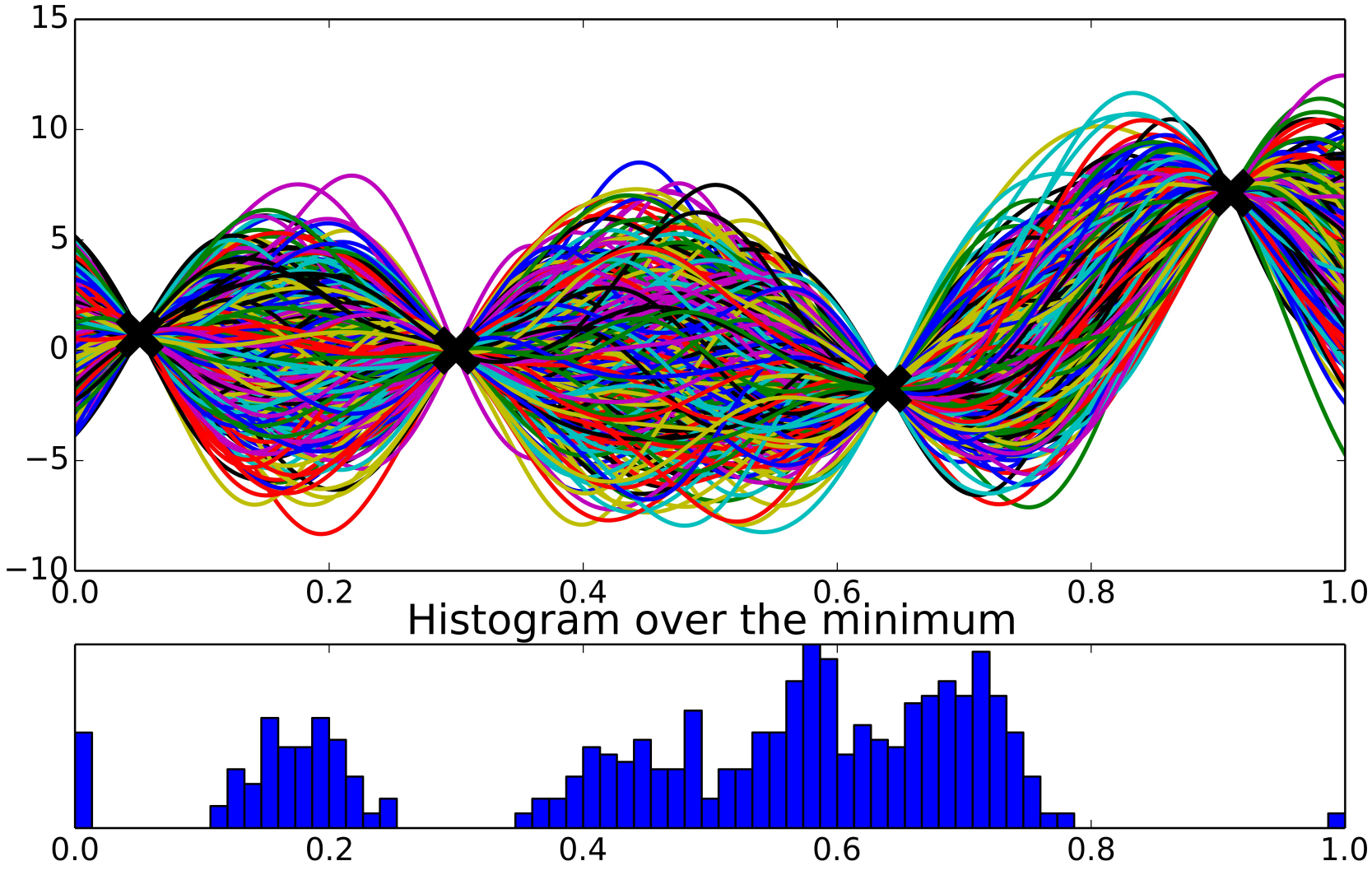
Intuitive solution

Hundred curves



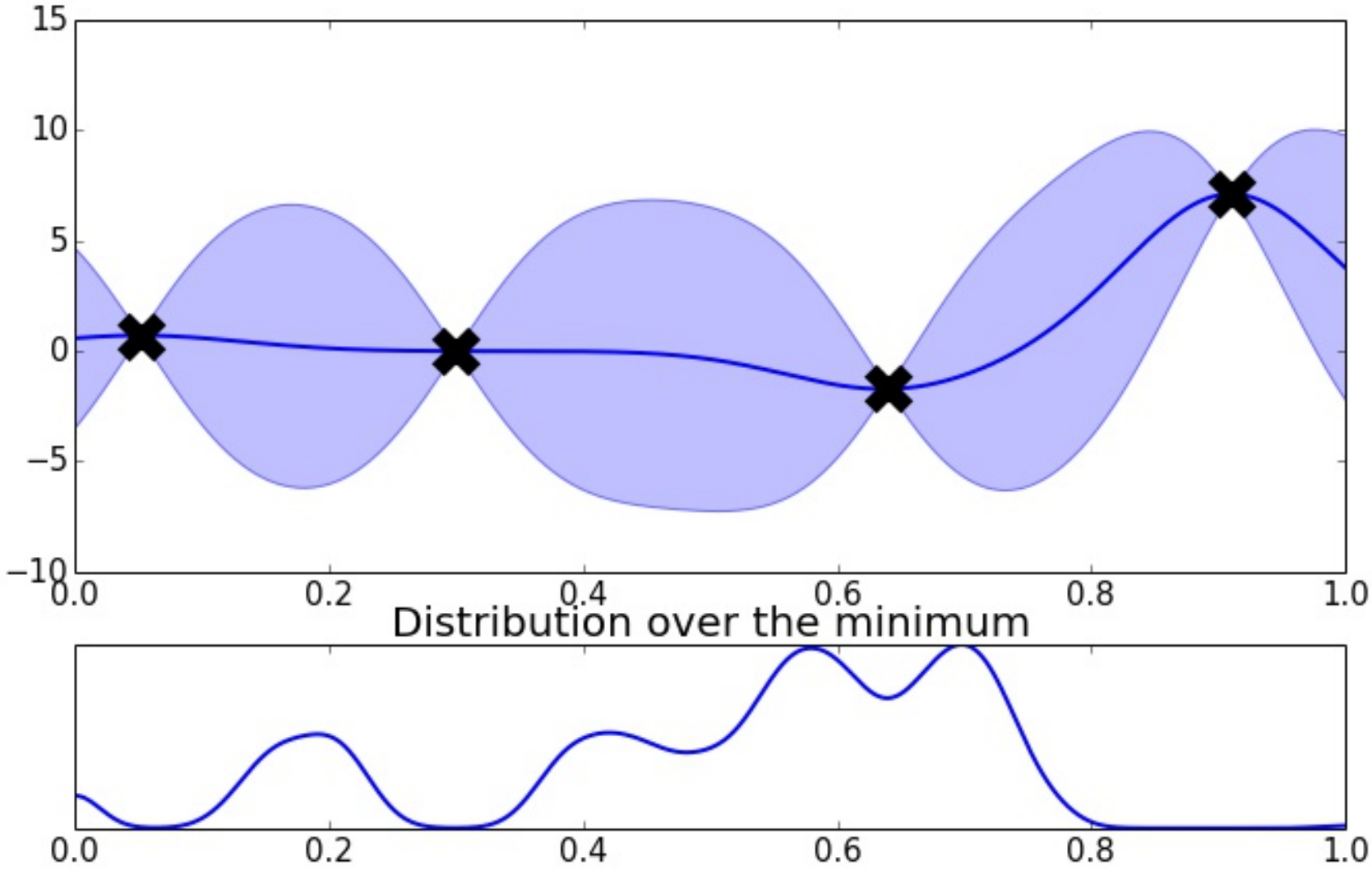
Intuitive solution

Many curves



Intuitive solution

Infinite curves



What just happened?

- We made some prior assumptions about our function.
- Information about the minimum is now encoded in a new function (the probability distribution p_{min} in this case).
- We can use p_{min} (or a functional of it: entropy search) to decide where to sample next.
- Other functions to encode relevant information about the minimum are possible, e. g. the 'marginal expected gain' at each location.

Bayesian Optimization

Methodology to perform global optimization of multimodal black-box functions [Mockus, 1978].

1. Choose some *prior measure* over the space of possible objectives f.

2. Combine prior and the likelihood to get a *posterior* over the objective given some observations.

3. Use the posterior to decide where to take the next evaluation according to some *acquisition function*.

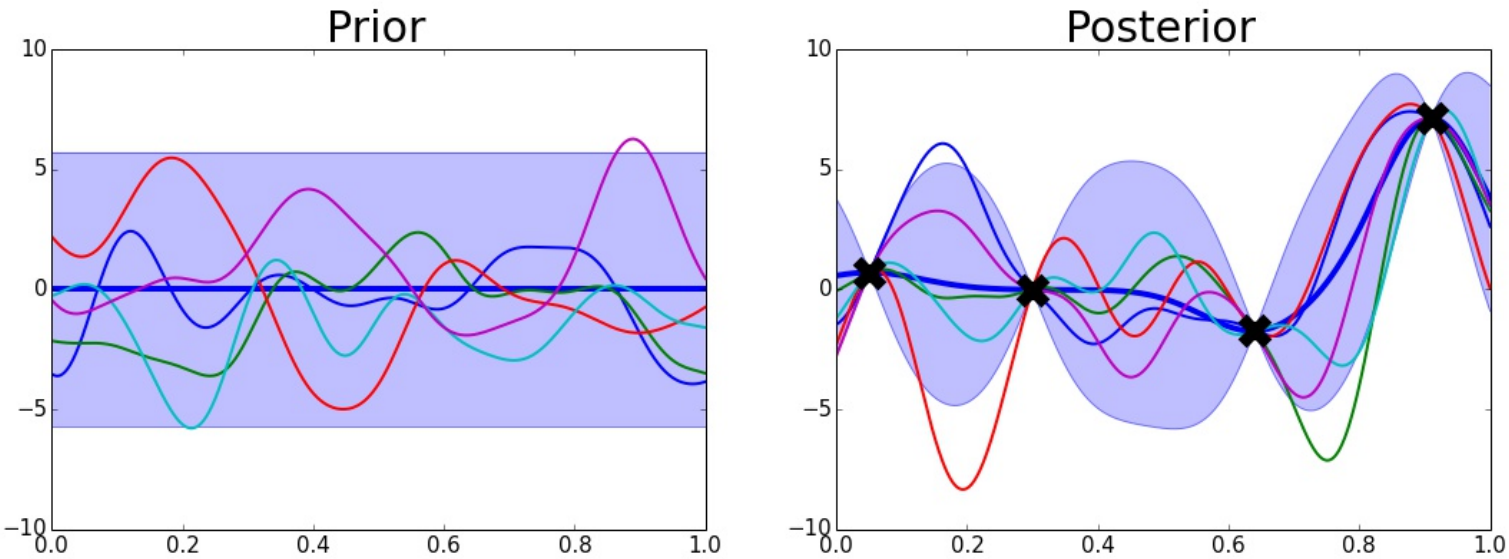
4. Augment the data.

Iterate between 2 and 4 until the evaluation budget is over.

Probability measure over functions: Gaussian Processes

Other choices: t-Student processes [Shah et al. 2013], Deep NN [Snoek et al., 2015].

Infinite-dimensional probability density, such that each linear finite-dimensional restriction is multivariate Gaussian.



- Model $f(x) \sim \mathcal{GP}(\mu(x), k(x, x'))$ is determined by the mean function $m(x)$ and covariance function $k(x, x'; \theta)$.
- Posterior mean $\mu(x; \theta, \mathcal{D})$ and variance $\sigma(x; \theta, \mathcal{D})$ can be computed explicitly given a dataset $\mathcal{D}$.

Acquisition functions

Making use of the model uncertainty

GPs has marginal closed-form for the posterior mean $\mu(x)$ and variance $\sigma^2(x)$.

- **Exploration**: Evaluate in places where the variance is large.
- **Exploitation**: Evaluate in places where the mean is low.

Acquisition functions balance these two factors to determine where to evaluate next.

Exploration vs. exploitation

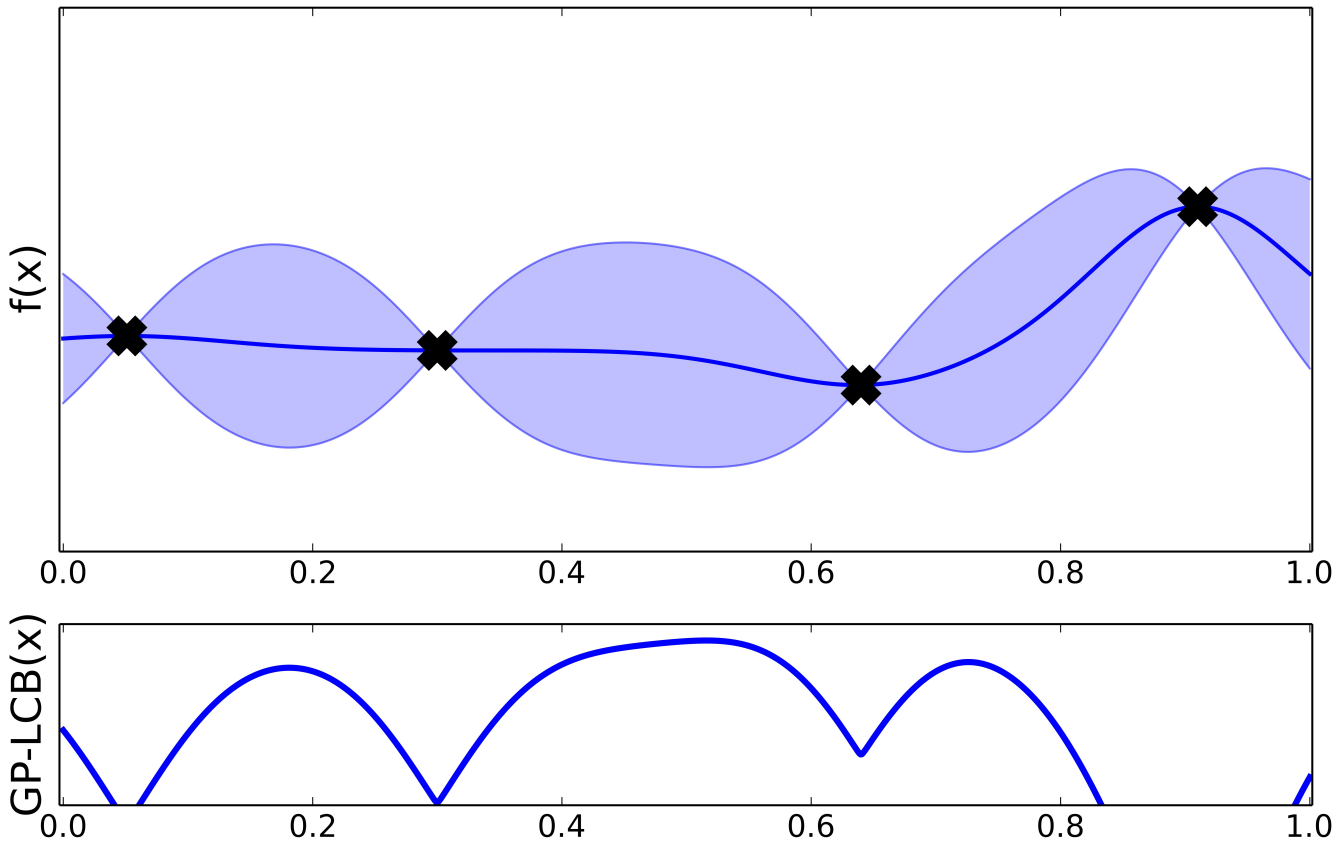
Bayesian optimization explains human active search [Borji and Itti, 2013]

GP Upper (lower) Confidence Band

[Srinivas et al., 2010]

Direct balance between exploration and exploitation:

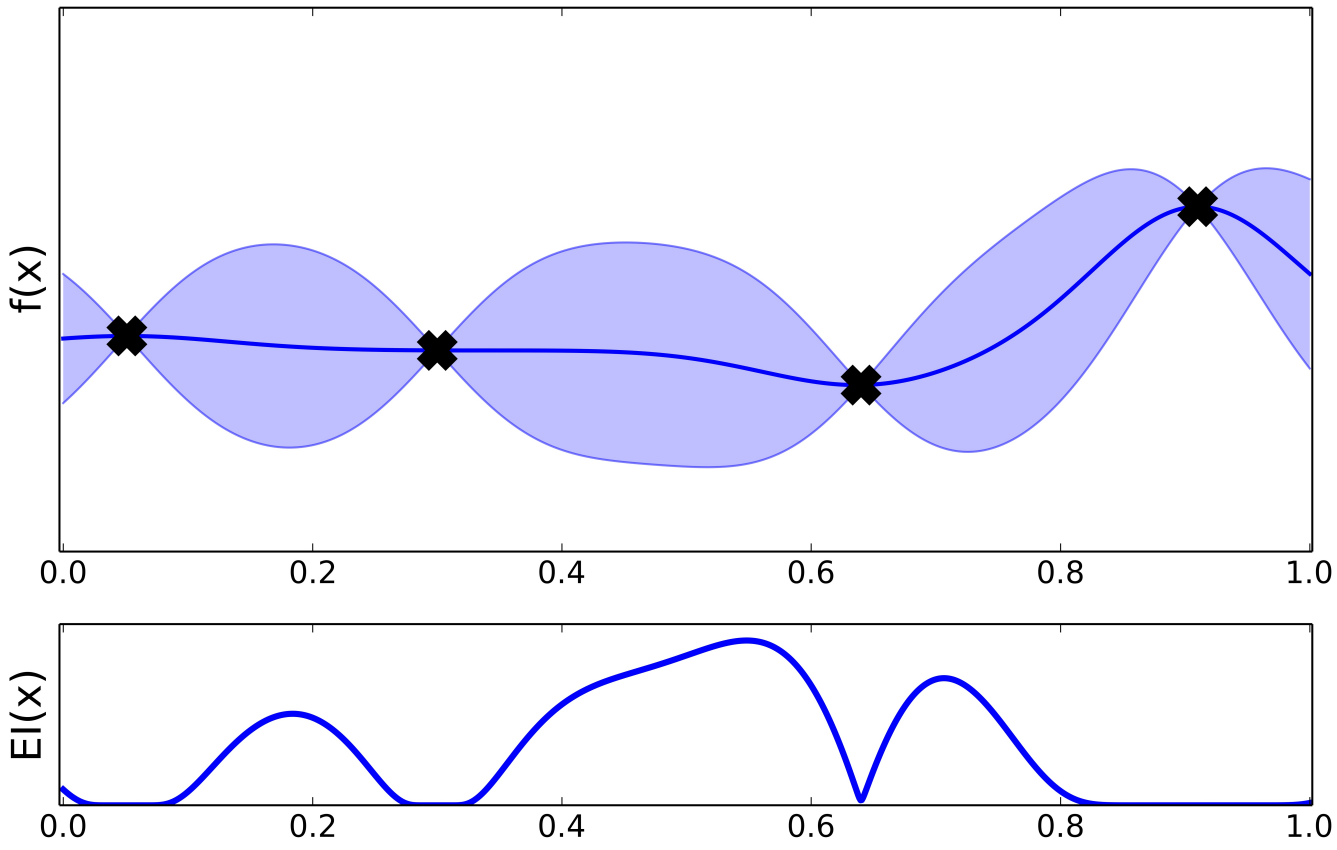
$$\alpha_{LCB}(\mathbf{x}; \theta, \mathcal{D}) = -\mu(\mathbf{x}; \theta, \mathcal{D}) + \kappa\sigma(\mathbf{x}; \theta, \mathcal{D})$$



Expected Improvement

[Jones et al., 1998]

$$\alpha_{EI}(\mathbf{x}; \theta, \mathcal{D}) = \int_y \max(0, y_{best} - y) p(y|\mathbf{x}; \theta, \mathcal{D}) dy$$

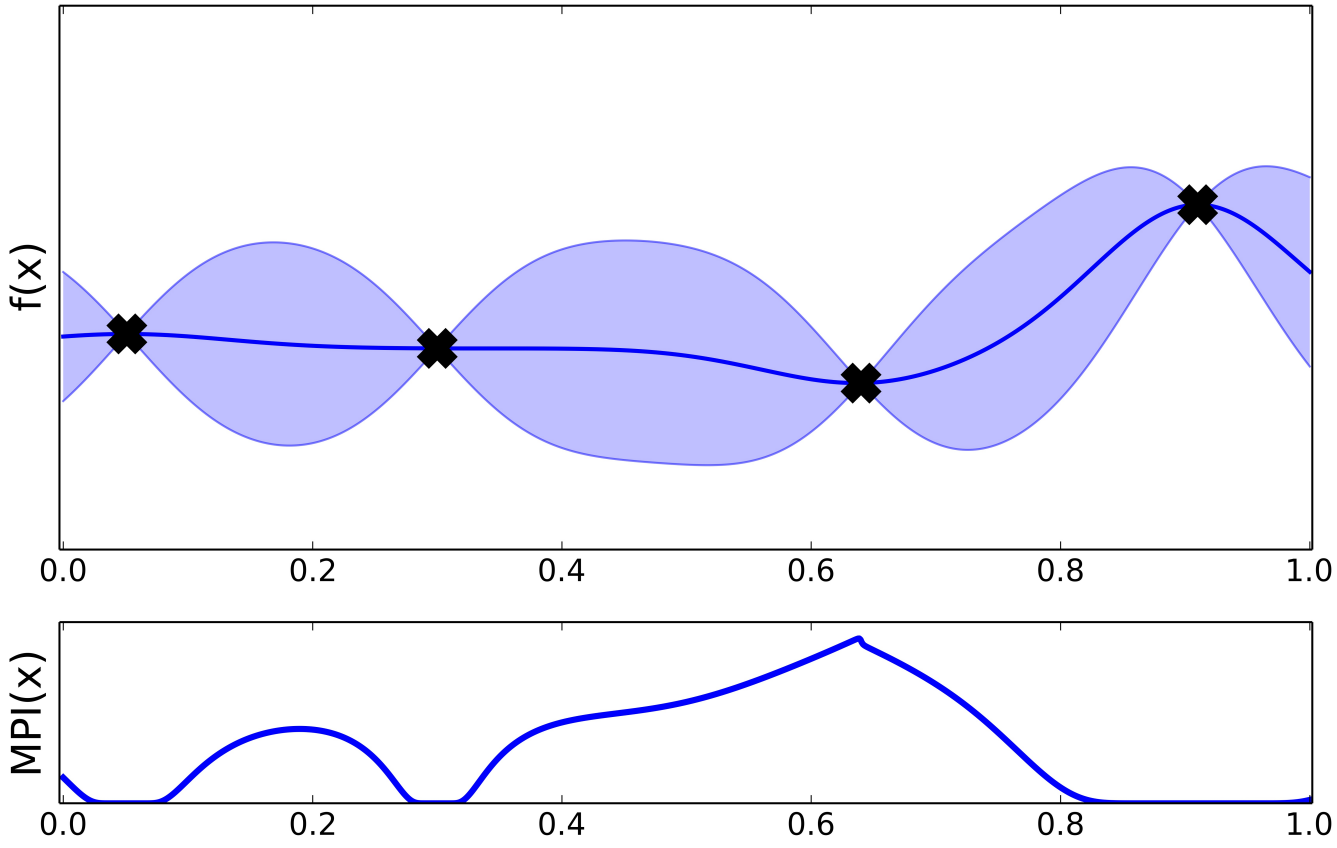


Maximum Probability of Improvement

[Hushner, 1964]

$$\gamma(\mathbf{x}) = \sigma(\mathbf{x}; \theta, \mathcal{D})^{-1}(\mu(\mathbf{x}; \theta, \mathcal{D}) - y_{best})$$

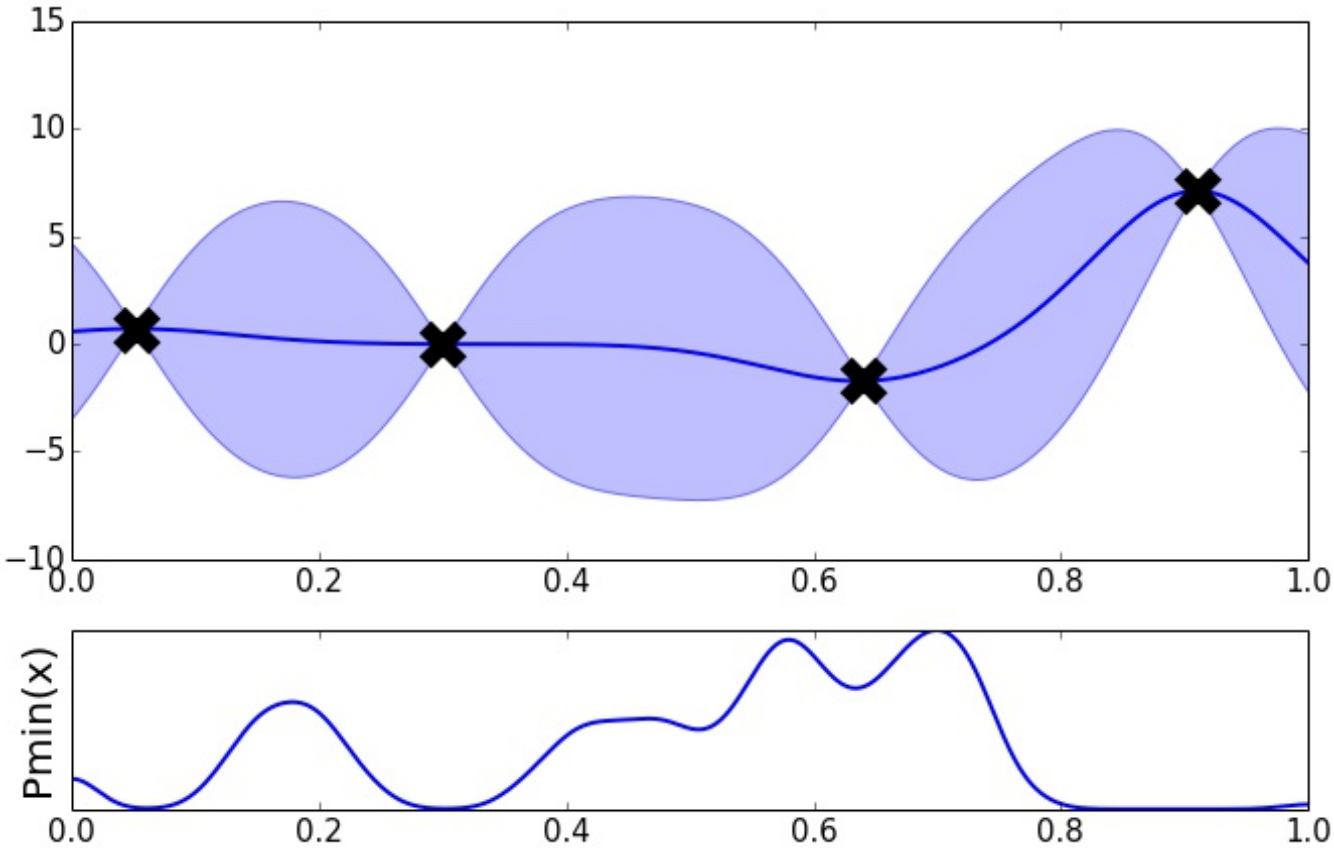
$$\alpha_{MPI}(\mathbf{x}; \theta, \mathcal{D}) = p(f(\mathbf{x}) < y_{best}) = \Phi(\gamma(\mathbf{x}))$$



Information-theoretic approaches

[Hennig and Schuler, 2013; Hernández-Lobato et al., 2014]

$$\alpha_{ES}(\mathbf{x}; \theta, \mathcal{D}) = H[p(x_{min}|\mathcal{D})] - \mathbb{E}_{p(y|\mathcal{D},\mathbf{x})}[H[p(x_{min}|\mathcal{D} \cup \{\mathbf{x}, y\})]]$$

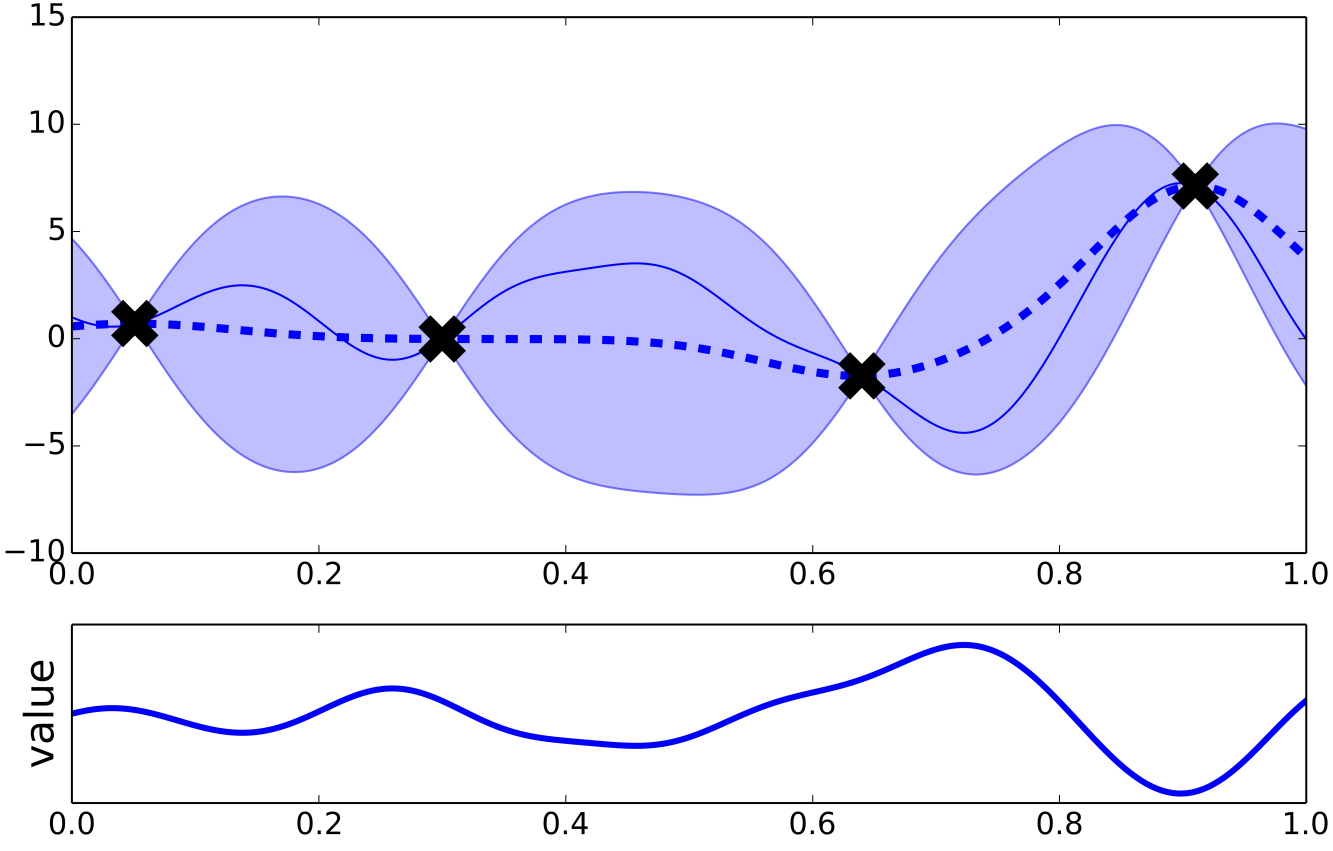


Thomson sampling

Probability matching

$$\alpha_{THOMSON}(\mathbf{x}; \theta, \mathcal{D}) = g(\mathbf{x})$$

$g(\mathbf{x})$ is sampled form $\mathcal{GP}(\mu(x), k(x, x'))$



Bayesian Optimization

As a 'mapping' between two problems

BO is an strategy to transform the problem

$$x_M = \arg\min_{x \in \mathcal{X}} f(x)$$

unsolvable!

into a series of problems:

$$x_{n+1} = \arg\max_{x \in \mathcal{X}} \alpha(x; \mathcal{D}_n, \mathcal{M}_n)$$

solvable!

where now:

- $\alpha(x)$ is inexpensive to evaluate.
- The gradients of $\alpha(x)$ are typically available.
- Still need to find x_{n+1}: DIRECT, cma, gradient methods.

Illustration of BO

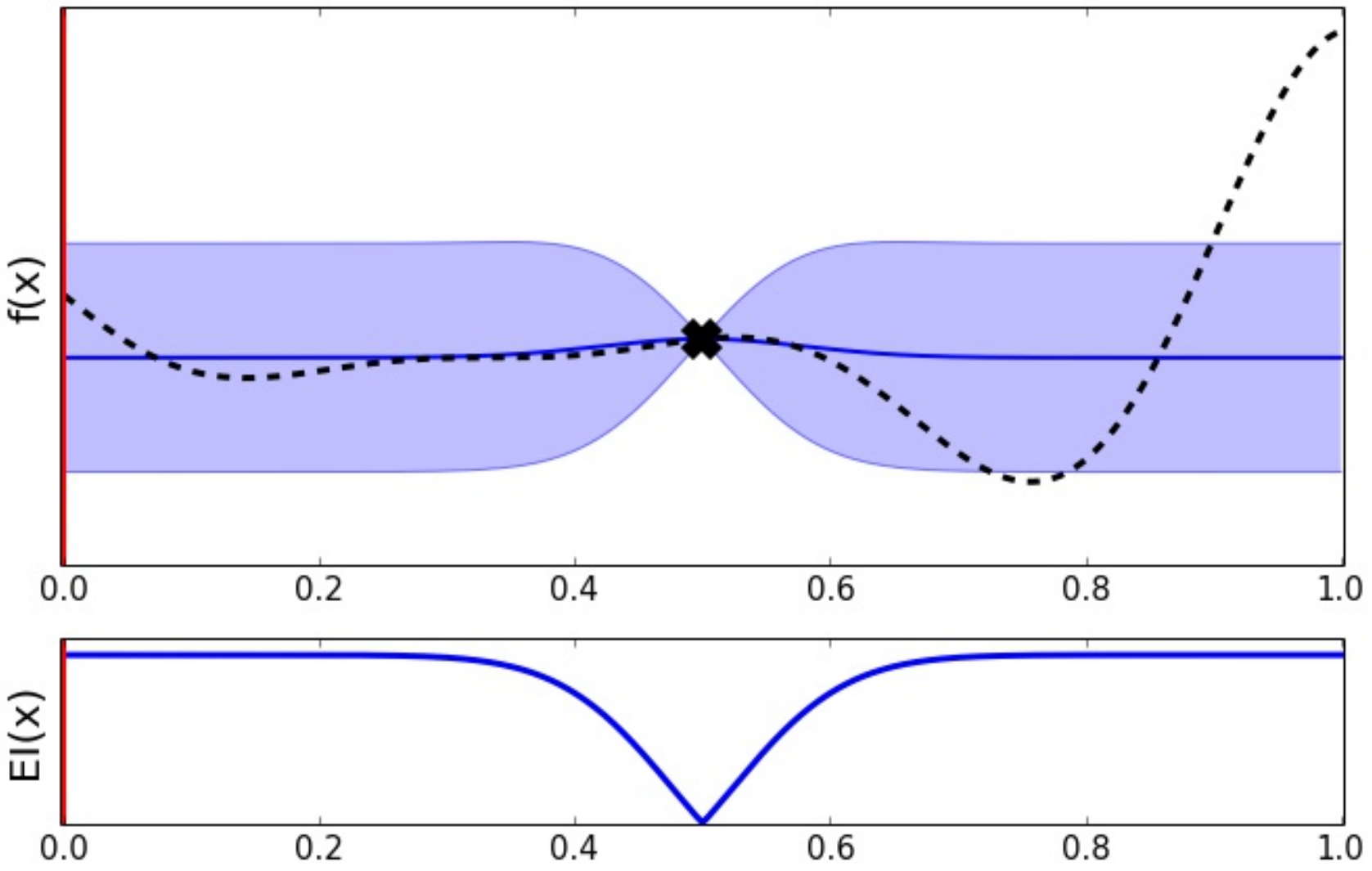


Illustration of BO

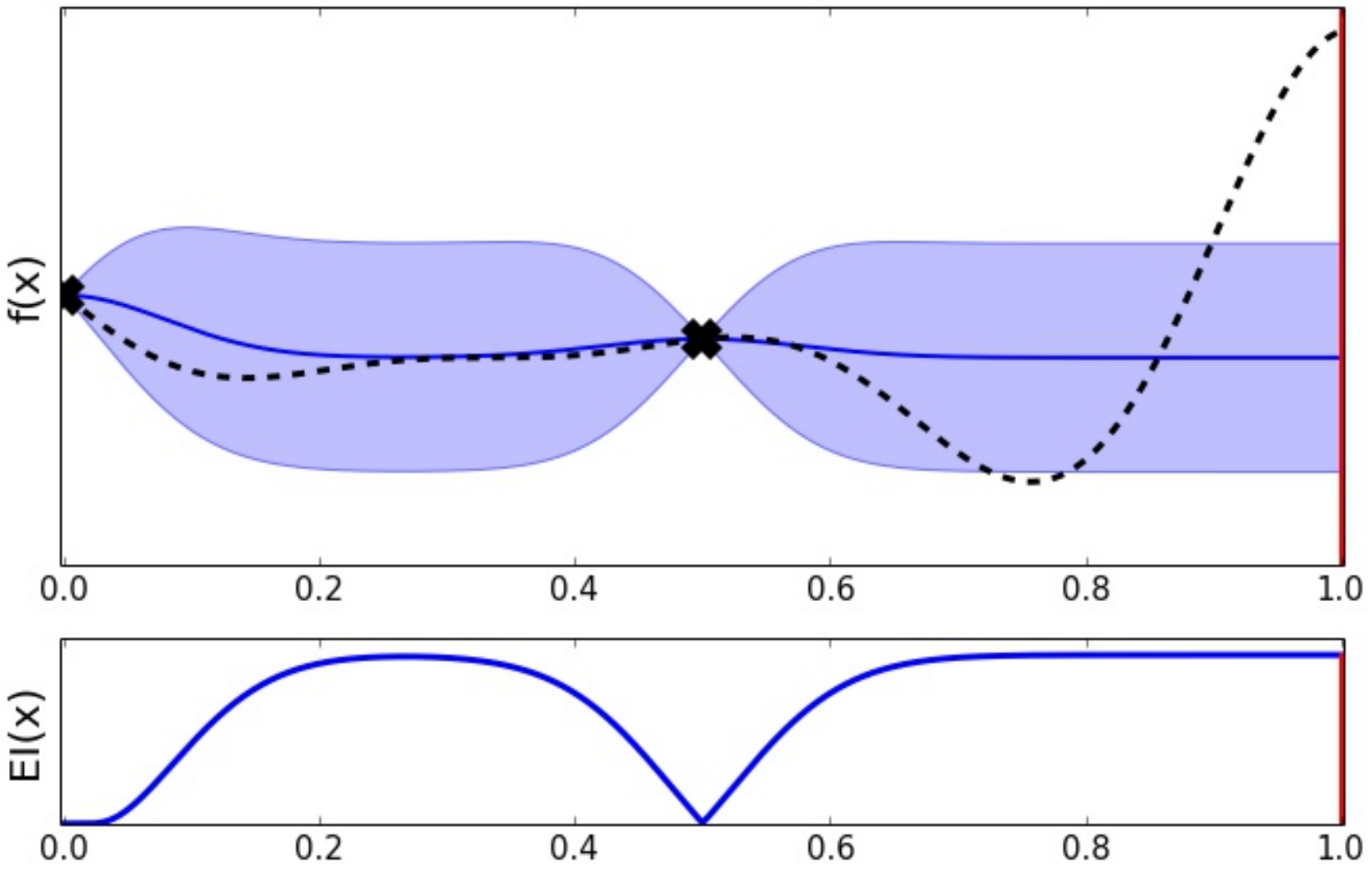


Illustration of BO

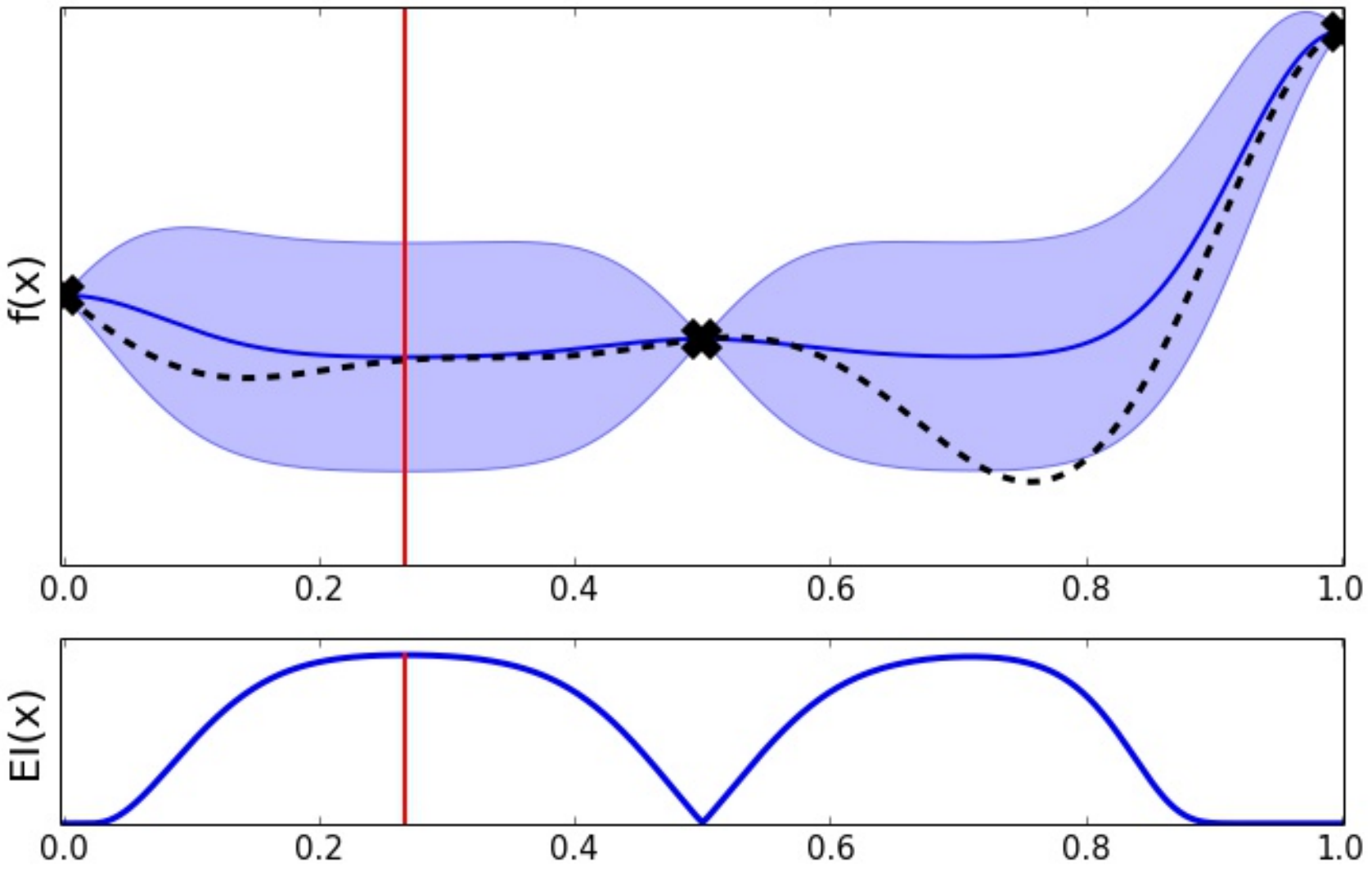


Illustration of BO

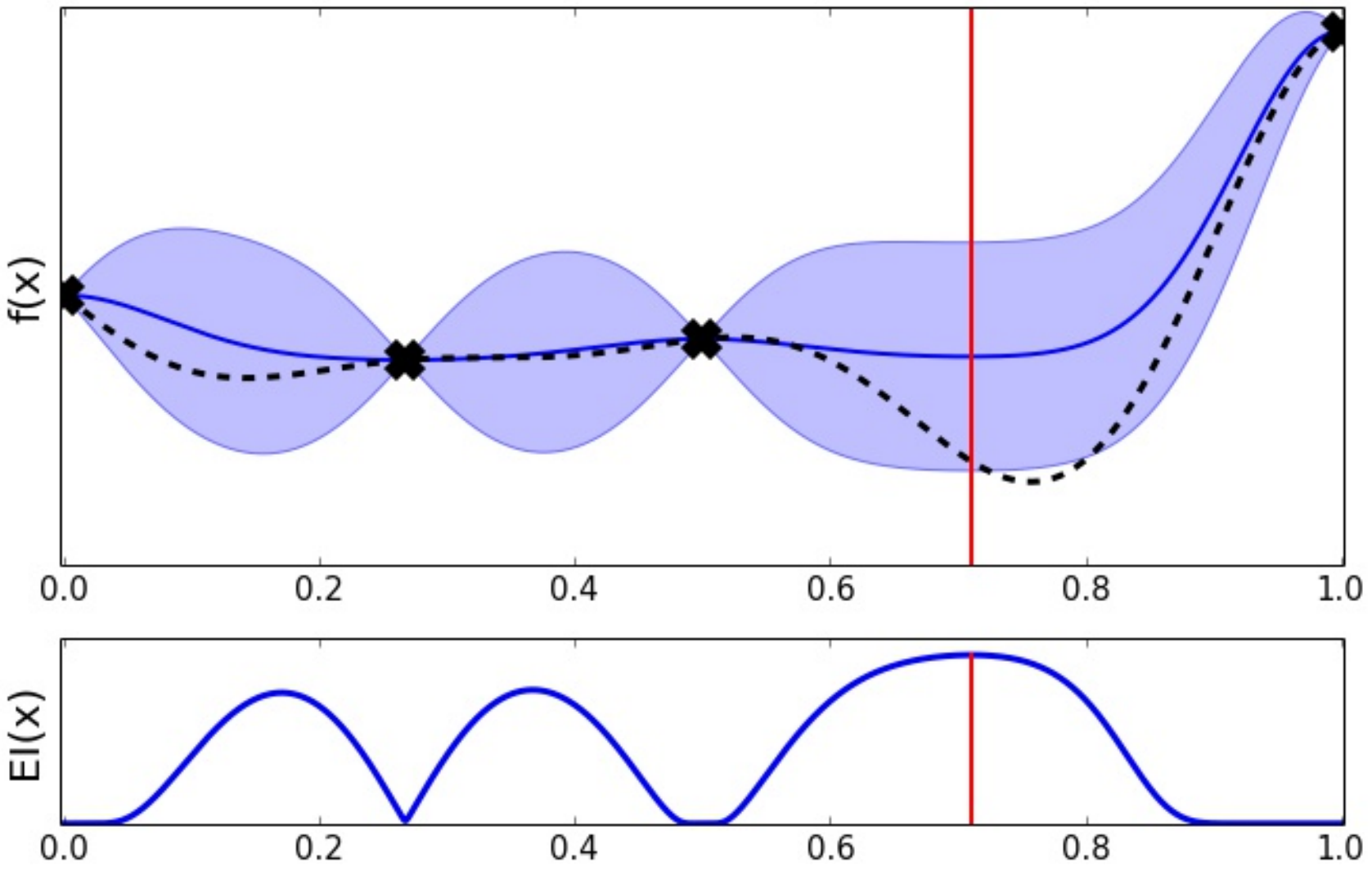


Illustration of BO

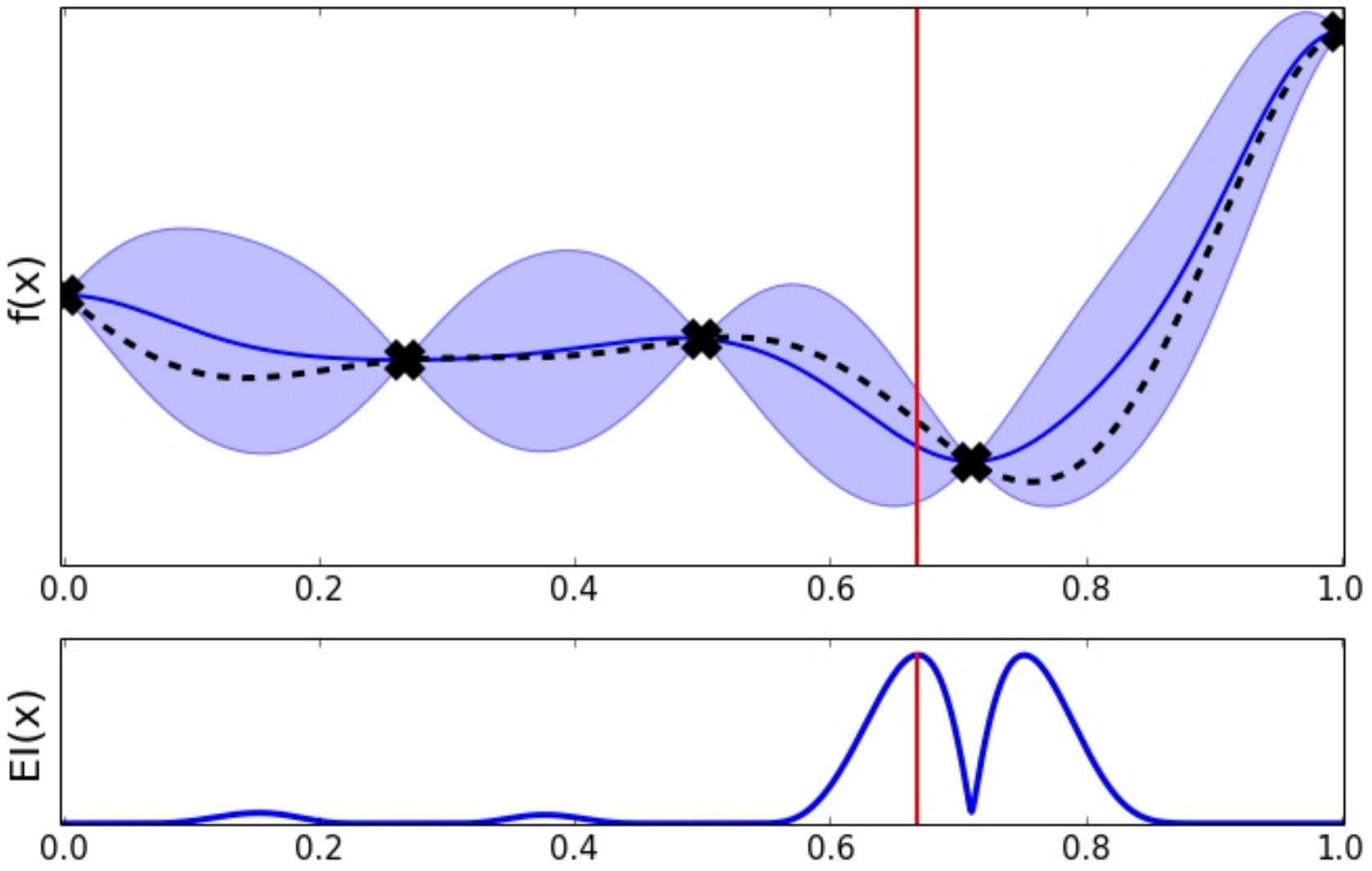


Illustration of BO

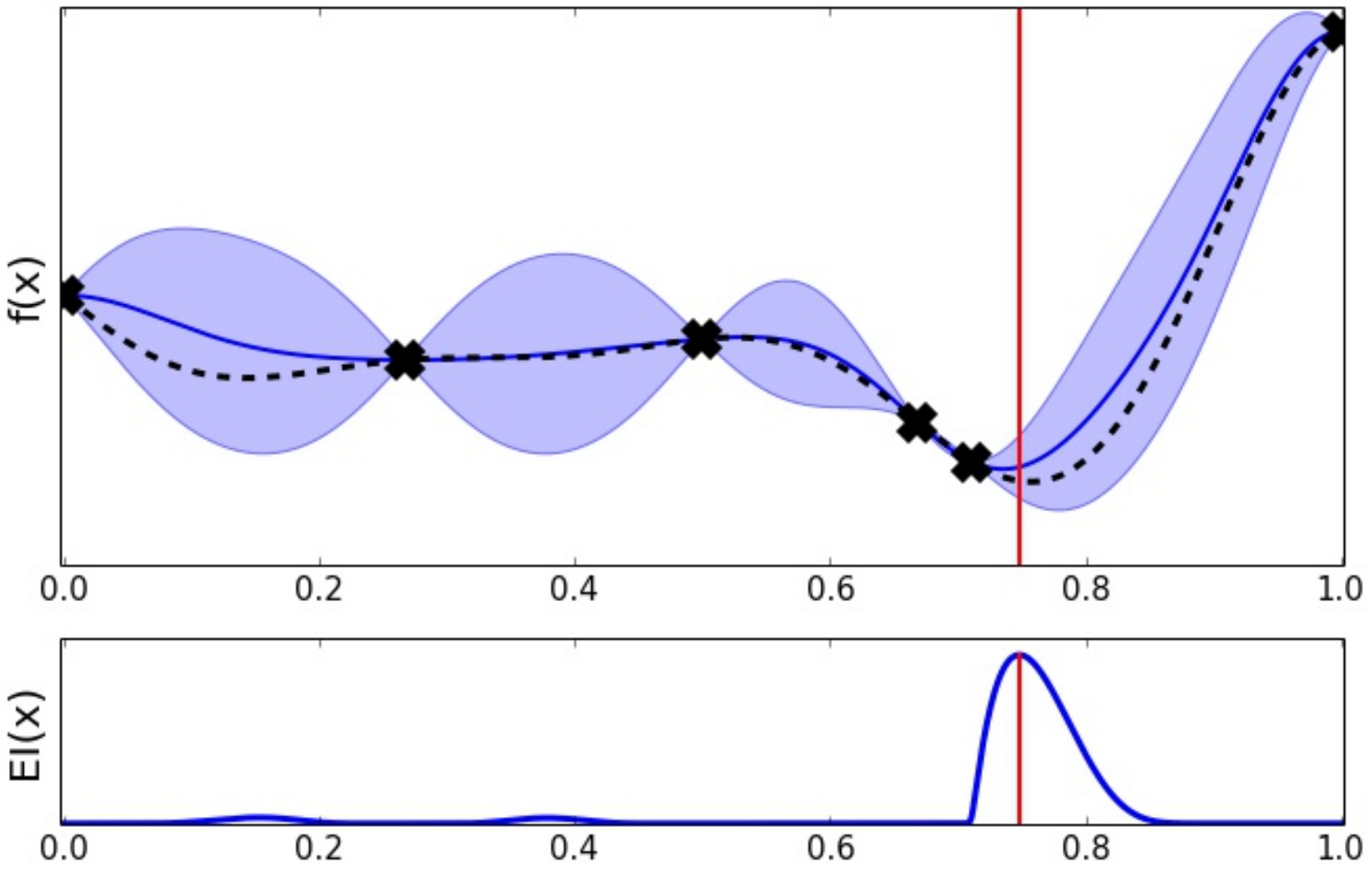


Illustration of BO

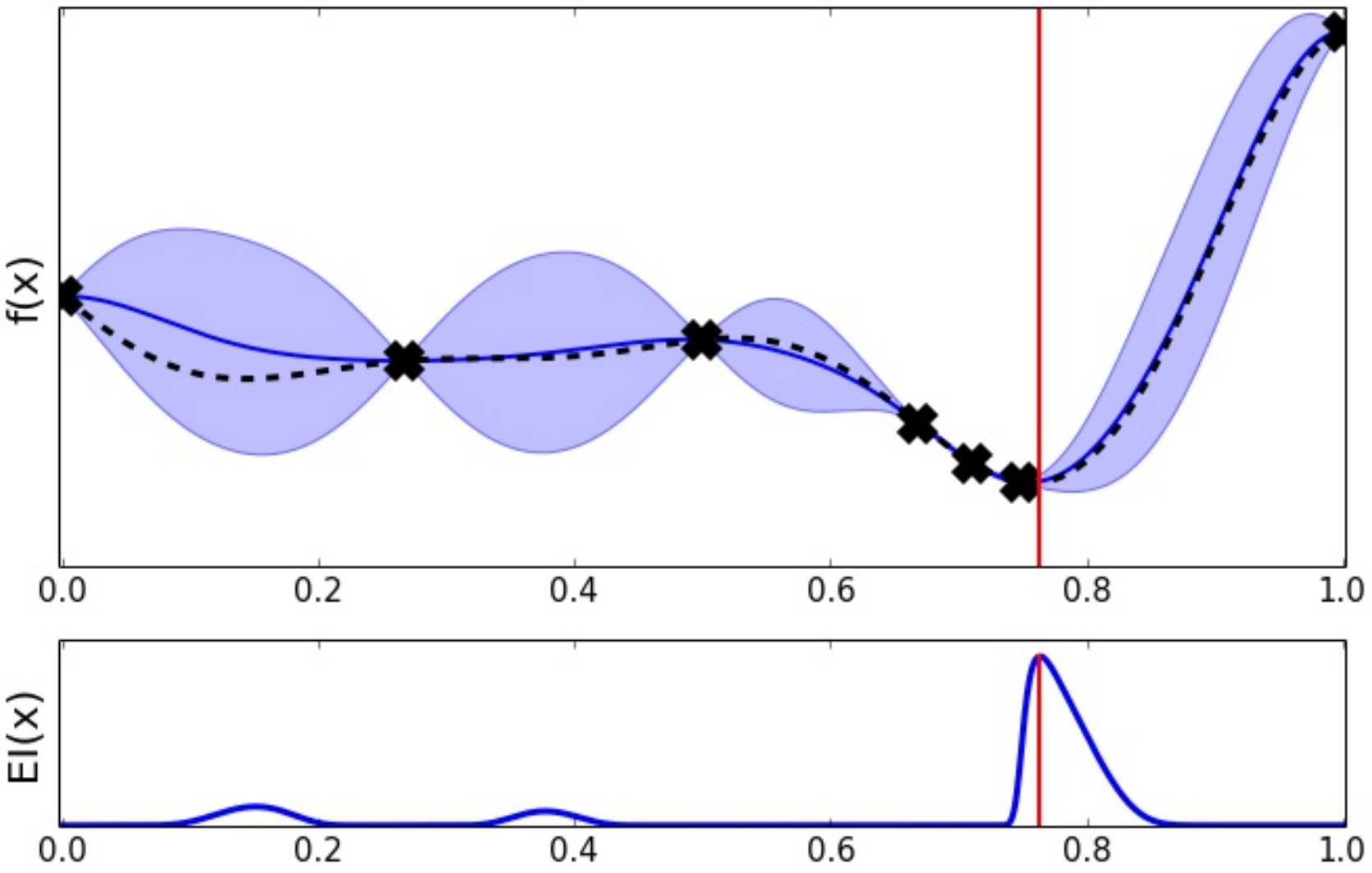


Illustration of BO

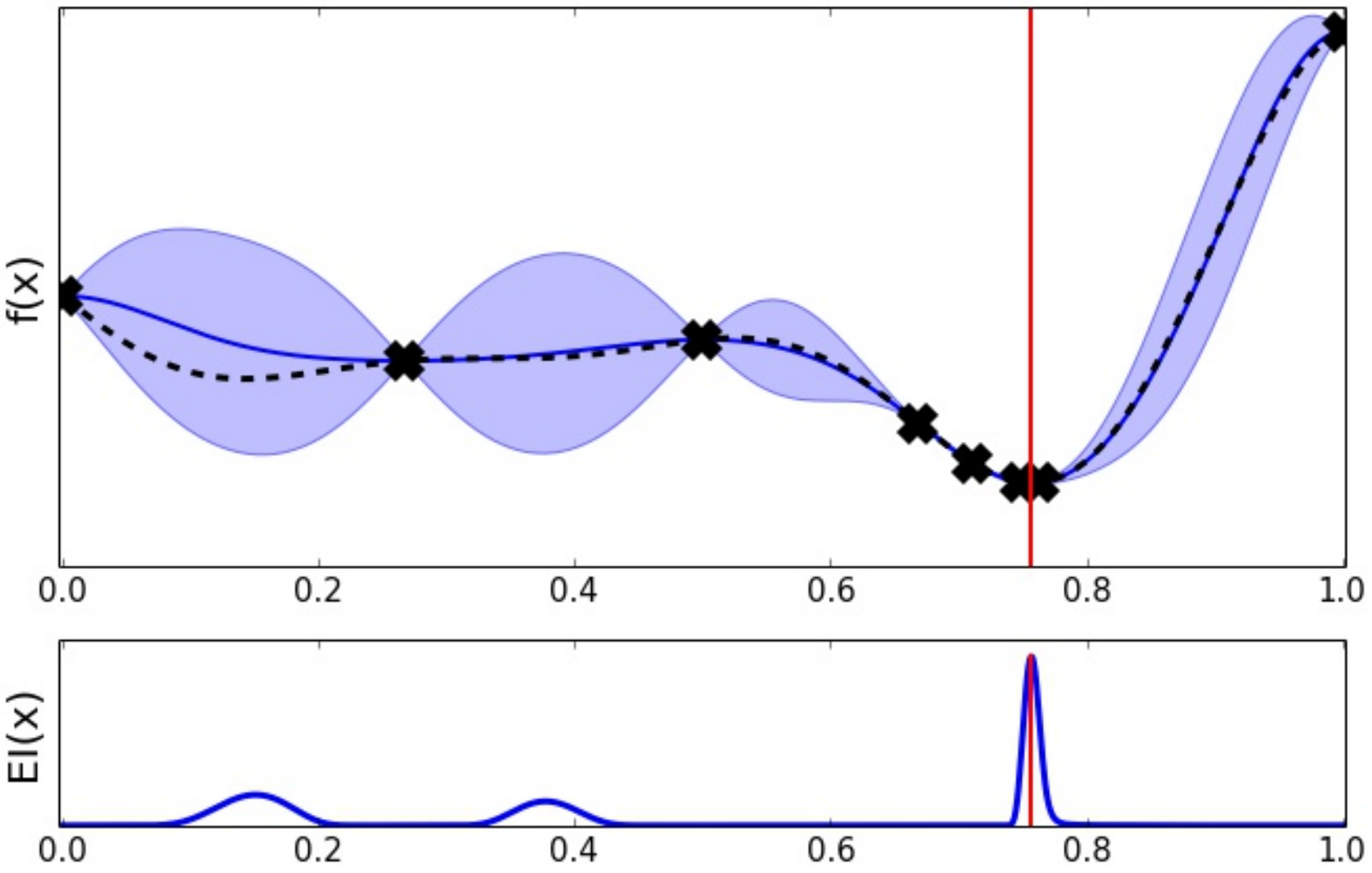
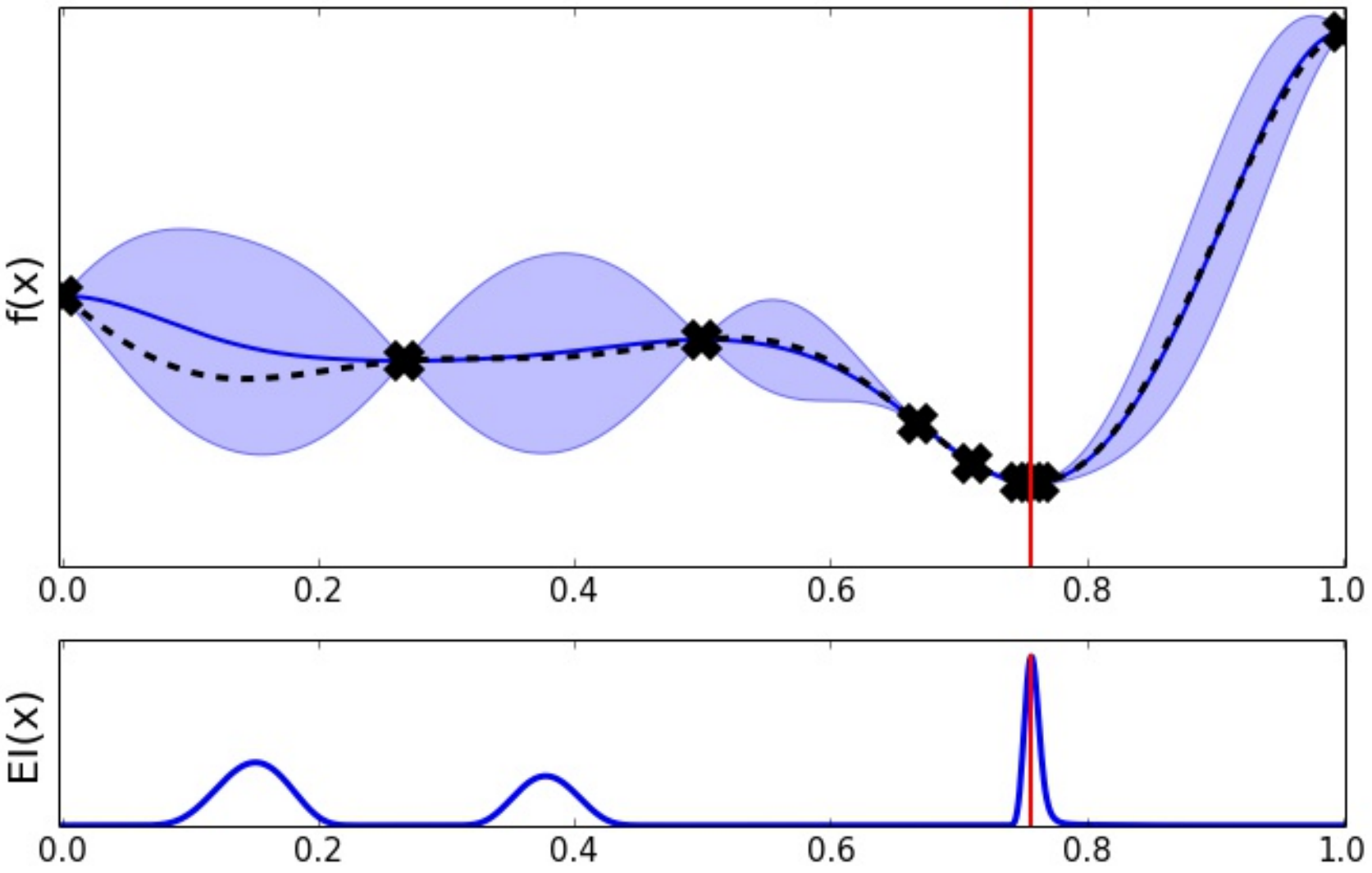


Illustration of BO



Why these ideas have been ignored for years?

- BO depends on its own parameters.
- Miss specification of the model has terrible consequences for the optimization.
- Lack of software to apply these methods as a black optimzation boxes.
- Reduced scalability (in dimensions and number of evaluations).

Practical Bayesian Optimization of Machine Learning Algorithms.
Snoek, Larochelle and Adams. NIPS 2012 (Spearmint)

+

Other works of M. Osborne, P. Hennig, N. de Freitas, etc.

Bayesian optimization now

- Hot topic in Machine Learning.
- The BO workshop at NIPS is well stablished and it is a mini-conference itself.

Extensions of Bayesian Optimization

- Multi-task Bayesian optimization [Wersky et all., 2013].

Extensions of Bayesian Optimization

- Multi-task Bayesian optimization [Wersky et all., 2013].
- Bayesian optimization for high dimensional problems [Wang et al., 2013].

Extensions of Bayesian Optimization

- Multi-task Bayesian optimization [Wersky et all., 2013].
- Bayesian optimization for high dimensional problems [Wang et al., 2013].
- Non-myopic methods [Osborne, 2010].

Extensions of Bayesian Optimization

- Multi-task Bayesian optimization [Wersky et all., 2013].
- Bayesian optimization for high dimensional problems [Wang et al., 2013].
- Non-myopic methods [Osborne, 2010].
- Discrete domains (armed bandits) [Srinivas et al., 2010].

Extensions of Bayesian Optimization

- Multi-task Bayesian optimization [Wersky et all., 2013].
- Bayesian optimization for high dimensional problems [Wang et al., 2013].
- Non-myopic methods [Osborne, 2010].
- Discrete domains (armed bandits) [Srinivas et al., 2010].
- Parallel approaches [Chevalier and Ginsbourger 2012].

Extensions of Bayesian Optimization

- Multi-task Bayesian optimization [Wersky et all., 2013].
- Bayesian optimization for high dimensional problems [Wang et al., 2013].
- Non-myopic methods [Osborne, 2010].
- Discrete domains (armed bandits) [Srinivas et al., 2010].
- Parallel approaches [Chevalier and Ginsbourger 2012].
- Conditional parameter spaces [Swersky et al. 2013].

Extensions of Bayesian Optimization

- Multi-task Bayesian optimization [Wersky et all., 2013].
- Bayesian optimization for high dimensional problems [Wang et al., 2013].
- Non-myopic methods [Osborne, 2010].
- Discrete domains (armed bandits) [Srinivas et al., 2010].
- Parallel approaches [Chevalier and Ginsbourger 2012].
- Conditional parameter spaces [Swersky et al. 2013].
- Applications to robotics, molecule design, etc.

BO at SheffieldML

Open software

+

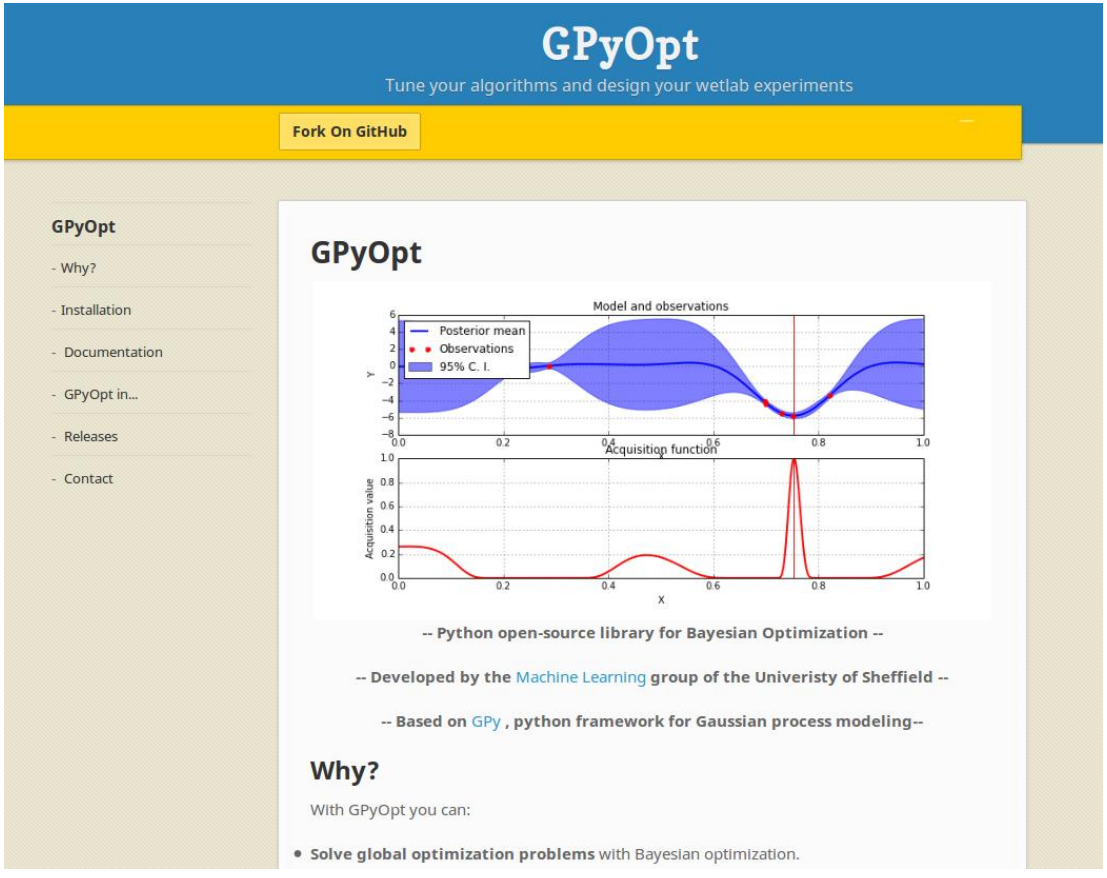
Scalable BO methods

+

Applications

Open Software: GPyOpt

http://sheffieldml.github.io/GPyOpt/



We will use it in the lab session

GPyOpt

- Python module for BO.
- Based on GPy. All functionalities available.
- Sparse GPs, Multi-output GPs, several likelihoods, etc.
- Parallel optimization.

GPyOpt: methods of use

Modular BO

```
k = GPy.kern.RBF(1)
BO = BayesianOptimization(f=f, bounds=b, acquisition='EI', kernel=k)
BO.run_optimization(max_iter)
```

Automatic ML

```
param = GPyOpt.methods.autoTune(objective, bounds)
```

Use GPyOpt using the same interface as Spearmint

config.json + problem.py

Scalable Methods: Parallel/batch BO

Avoiding the bottleneck of evaluating f

- Cost of $f(\mathbf{x}_n)$ = cost of $\{f(\mathbf{x}_{n,1}), \ldots, f(\mathbf{x}_{n,nb})\}$.
- Many cores available, simultaneous lab experiments, etc.

Considerations when designing a batch

- Available pairs $\{(\mathbf{x}_j, y_i)\}_{i=1}^{n}$ are augmented it with the evaluations of f on $\mathcal{B}_t^{n_b} = \{\mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,nb}\}$.

- Goal: design $\mathcal{B}_1^{n_b}, \ldots, \mathcal{B}_m^{n_b}$.

Notation:

- $\mathcal{I}_n$: represents the available data set $\mathcal{D}_n$ and the $\mathcal{GP}$ structure when n data points are available.

- $\alpha(\mathbf{x}; \mathcal{I}_n)$: generic acquisition function given $\mathcal{I}_n$.

Selecting $\mathbf{x}_{t,k}$, the k-th element of the t-th batch

Sequential policy

Maximize:

$$\alpha(\mathbf{x}; \mathcal{I}_{t,k-1})$$

Selecting $\mathbf{x}_{t,k}$, the k-th element of the t-th batch

Sequential policy

Maximize:

$$\alpha(\mathbf{x}; \mathcal{I}_{t,k-1})$$

Greedy batch policy: it is not tractable

Maximize:

$$\int \alpha(\mathbf{x}; \mathcal{I}_{t,k-1}) \prod_{j=1}^{k-1} p(y_{t,j}|\mathbf{x}_{t,j}, \mathcal{I}_{t,j-1}) p(\mathbf{x}_{t,j}|\mathcal{I}_{t,j-1}) d\mathbf{x}_{t,j} dy_{t,j}$$

where

- $p(y_{t,j}|\mathbf{x}_j, \mathcal{I}_{t,j-1})$: predictive distribution of the $\mathcal{GP}$.
- $p(\mathbf{x}_j|\mathcal{I}_{t,j-1}) = \delta(\mathbf{x}_{t,j} - \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha(\mathbf{x}; \mathcal{I}_{t,j-1}))$.

Available approaches

- Exploratory approaches, reduction in system uncertainty.
- Generate 'fake' observations of f using $p(y_{t,j}|\mathbf{x}_j, \mathcal{I}_{t,j-1})$.
- Simultaneously optimize elements on the batch using the joint distribution of $y_{t_1}, \dots y_{t,nb}$.

[Azimi et al., 2010; Azimi et al., 2011; Azimi et al., 2012; Desautels et al., 2012; Chevalier et al., 2013; Contal et al. 2013]

Bottleneck

All these methods require to iteratively update $p(y_{t,j}|\mathbf{x}_j, \mathcal{I}_{t,j-1})$ to model the iteration between the elements in the batch: $\mathcal{O}(n^3)$

How to design batches reducing this cost? BBO-LP

Goal: eliminate the marginalization step

"To develop an heuristic approximating the 'optimal batch design strategy' at lower computational cost, while incorporating information about global properties of f from the $\mathcal{GP}$ model into the batch design"

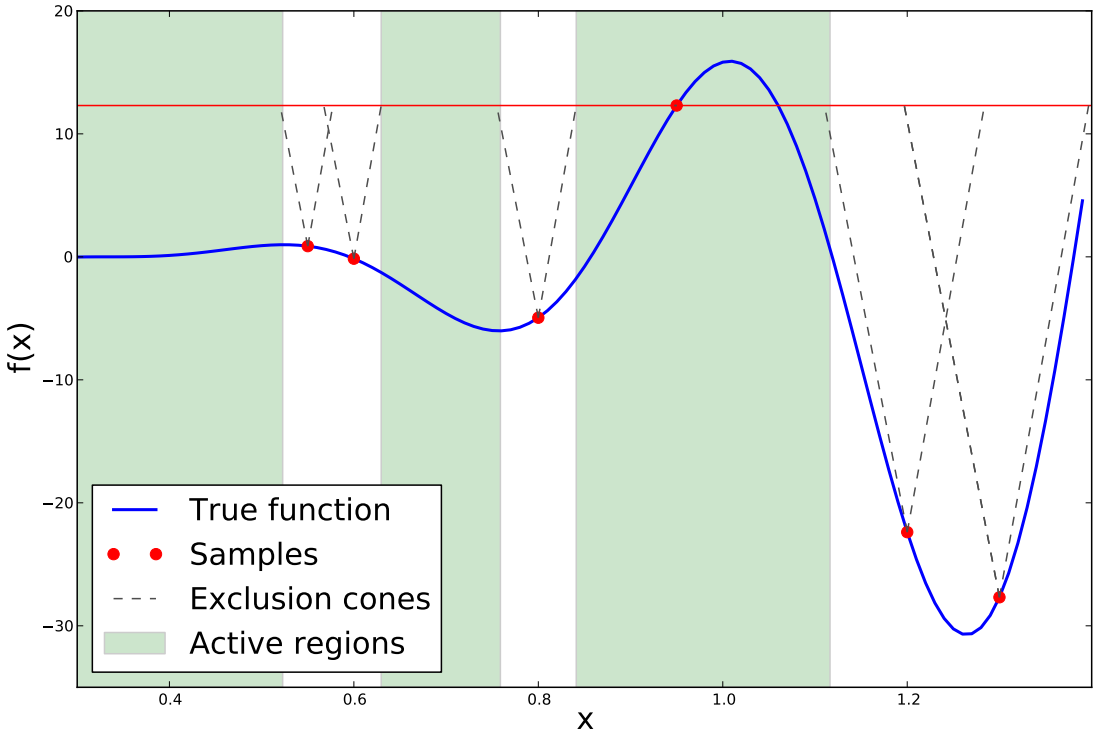
Lipschitz continuity:

$$|f(\mathbf{x}_1) - f(\mathbf{x}_2)| \leq L\|\mathbf{x}_1 - \mathbf{x}_2\|_p.$$

Interpretation of the Lipschitz continuity of f

$M = \max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$ and $B_{r_{x_j}}(\mathbf{x}_j) = \{\mathbf{x} \in \mathcal{X} : \|\mathbf{x} - \mathbf{x}_j\| \leq r_{x_j}\}$ where

$$r_{x_j} = \frac{M - f(\mathbf{x}_j)}{L}$$



$x_M \notin B_{r_{x_j}}(\mathbf{x}_j)$ otherwise, the Lipschitz condition is violated.

Probabilistic version of $B_{r_x}(\mathbf{x})$

We can do this because $f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$

- r_{x_j} is Gaussian with $\mu(r_{x_j}) = \frac{M - \mu(\mathbf{x}_j)}{L}$ and $\sigma^2(r_{x_j}) = \frac{\sigma^2(\mathbf{x}_j)}{L^2}$.

Probabilistic version of $B_{r_x}(\mathbf{x})$

We can do this because $f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$

- r_{x_j} is Gaussian with $\mu(r_{x_j}) = \frac{M - \mu(\mathbf{x}_j)}{L}$ and $\sigma^2(r_{x_j}) = \frac{\sigma^2(\mathbf{x}_j)}{L^2}$.

Local penalizers: $\varphi(\mathbf{x}; \mathbf{x}_j) = p(\mathbf{x} \notin B_{r_{\mathbf{x}_j}}(\mathbf{x}_j))$

$$\begin{aligned} \varphi(\mathbf{x}; \mathbf{x}_j) &= p(r_{\mathbf{x}_j} < \|\mathbf{x} - \mathbf{x}_j\|) \\ &= 0.5\text{erfc}(-z) \end{aligned}$$

where $z = \frac{1}{\sqrt{2\sigma_n^2(\mathbf{x}_j)}}(L\|\mathbf{x}_j - \mathbf{x}\| - M + \mu_n(\mathbf{x}_j))$.

- Reflects the size of the 'Lipschitz' exclusion areas.
- Approaches to 1 when $\mathbf{x}$ is far form $\mathbf{x}_j$ and decreases otherwise.

Idea to collect the batches

Without using explicitly the model.

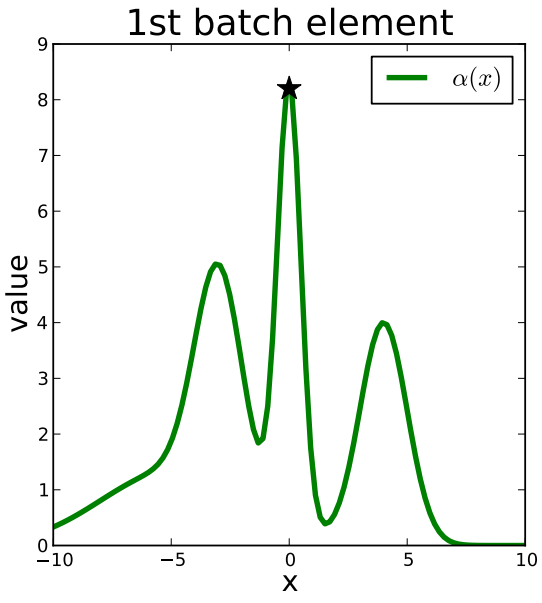
Optimal batch: maximization-marginalization

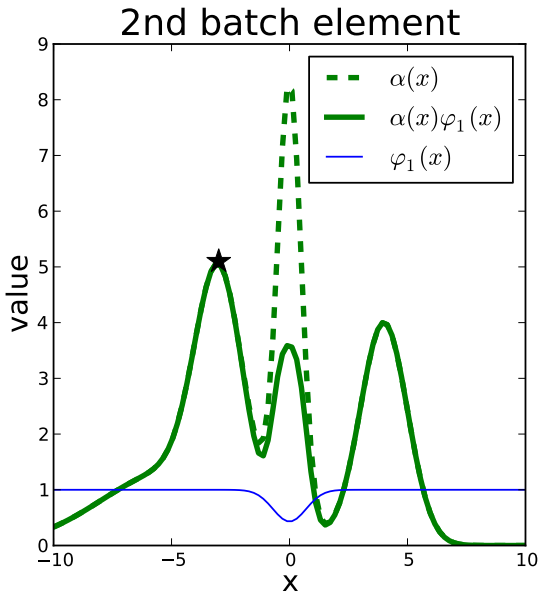
$$\int \alpha(\mathbf{x}; \mathcal{I}_{t,k-1}) \prod_{j=1}^{k-1} p(y_{t,j}|\mathbf{x}_{t,j}, \mathcal{I}_{t,j-1}) p(\mathbf{x}_{t,j}|\mathcal{I}_{t,j-1}) d\mathbf{x}_{t,j} dy_{t,j}$$

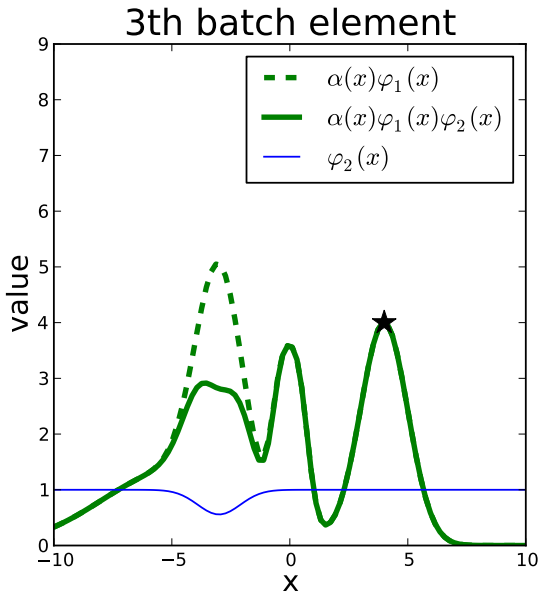
Proposal: maximization-penalization.

Use the $\varphi(\mathbf{x}; \mathbf{x}_j)$ to penalize the acquisition and predict the expected change in $\alpha(\mathbf{x}; \mathcal{I}_{t,k-1})$.

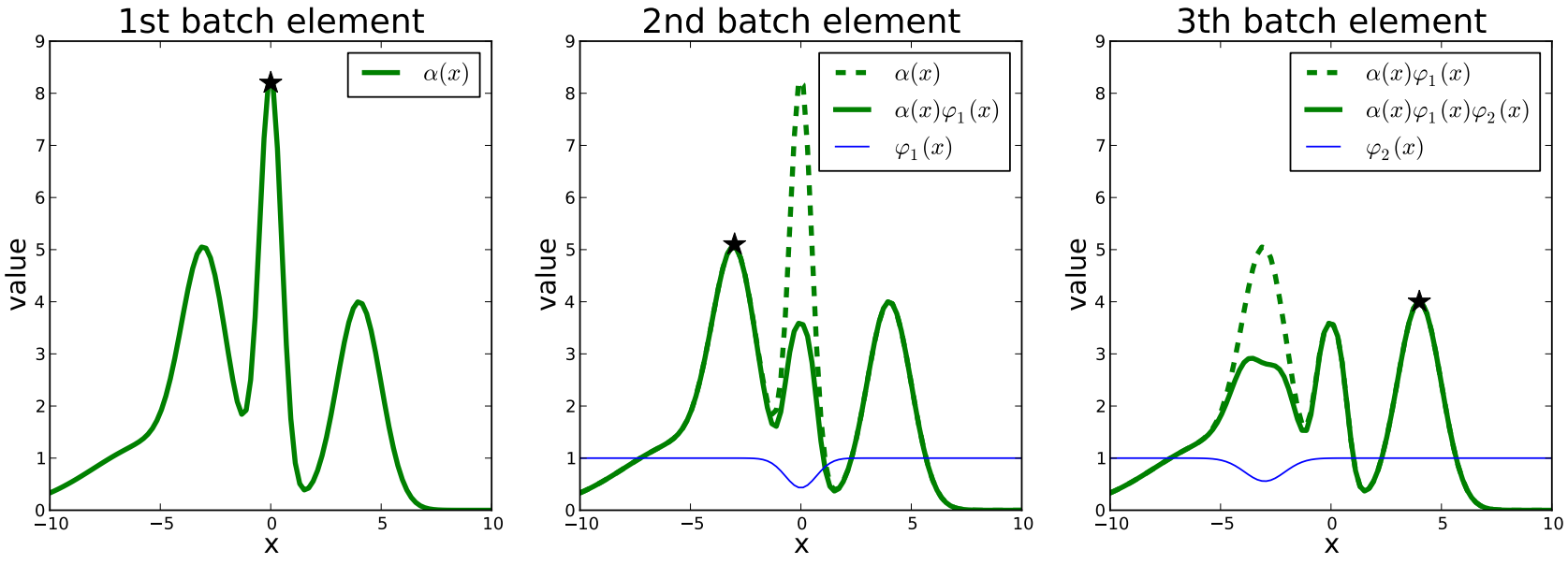
Local penalization strategy







Local penalization strategy

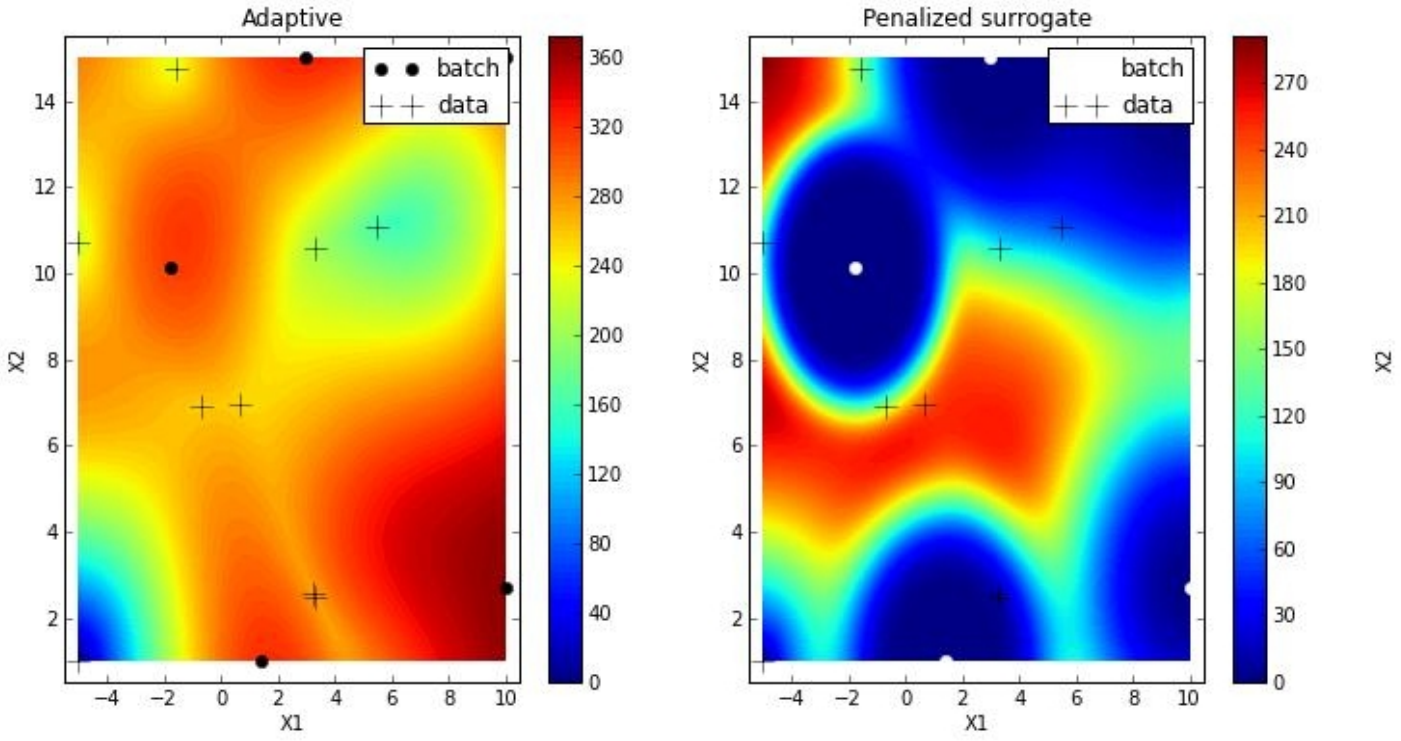


The maximization-penalization strategy selects $\mathbf{x}_{t,k}$ as

$$\mathbf{x}_{t,k} = \arg\max_{x \in \mathcal{X}} \left\{ g(\alpha(\mathbf{x}; \mathcal{I}_{t,0})) \prod_{j=1}^{k-1} \varphi(\mathbf{x}; \mathbf{x}_{t,j}) \right\},$$

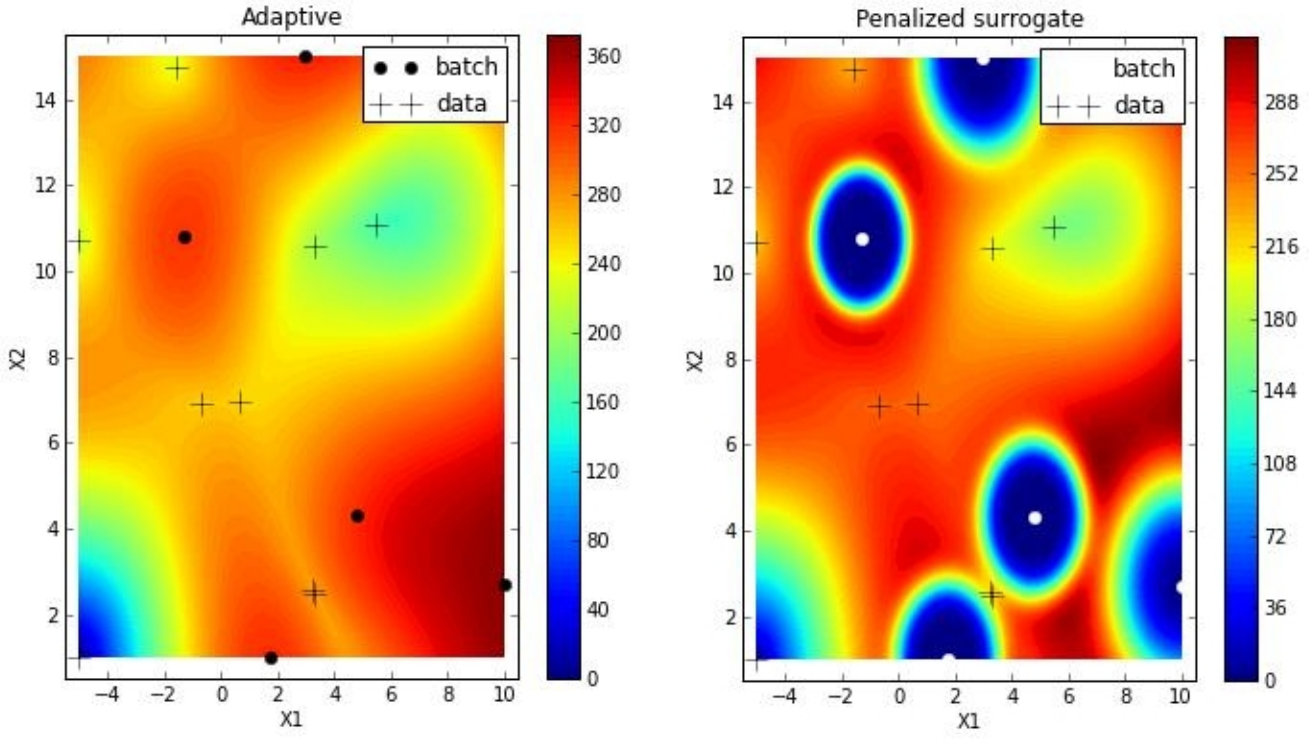
where g is a transformation of $\alpha(\mathbf{x}; \mathcal{I}_{t,0})$ to make it always positive [González, Dai, Hennig, Lawrence, 2015]

Example for $L = 50$



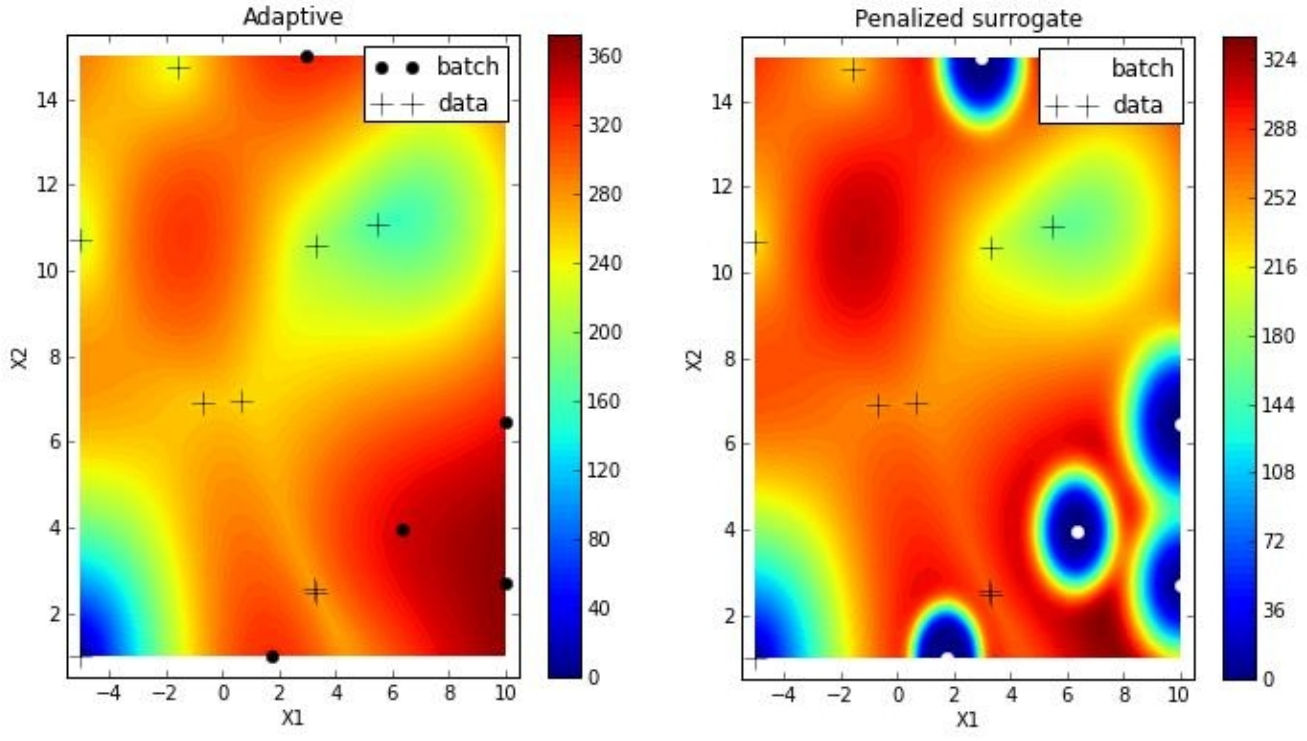
L controls the exploration-exploitation balance within the batch.

Example for $L = 100$



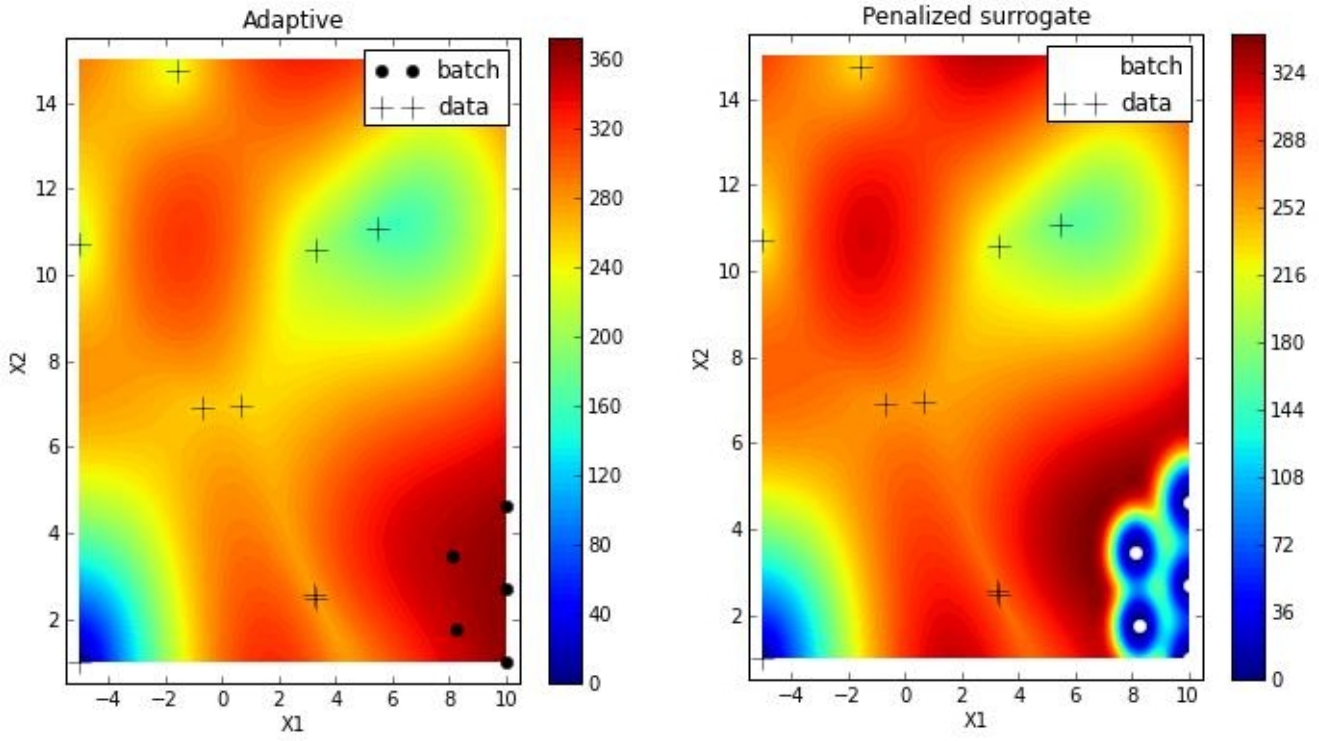
L controls the exploration-exploitation balance within the batch.

Example for $L = 150$



L controls the exploration-exploitation balance within the batch.

Example for $L = 250$



L controls the exploration-exploitation balance within the batch.

Finding an unique Lipschitz constant

Let $f : \mathcal{X} \to \mathbb{R}$ be a L-Lipschitz continuous function defined on a compact subset $\mathcal{X} \subseteq \mathbb{R}^D$. Then

$$L_p = \max_{\mathbf{x} \in \mathcal{X}} \|\nabla f(\mathbf{x})\|_p,$$

is a valid Lipschitz constant.

The gradient of f at $\mathbf{x}^*$ is distributed as a multivariate Gaussian

$$\nabla f(\mathbf{x}^*) | \mathbf{X}, \mathbf{y}, \mathbf{x}^* \sim \mathcal{N}(\mu_\nabla(\mathbf{x}^*), \Sigma^2_\nabla(\mathbf{x}^*))$$

We choose:

$$\hat{L}_{GP-LCA} = \max_{\mathcal{X}} \|\mu_\nabla(\mathbf{x}^*)\|$$

Sobol function

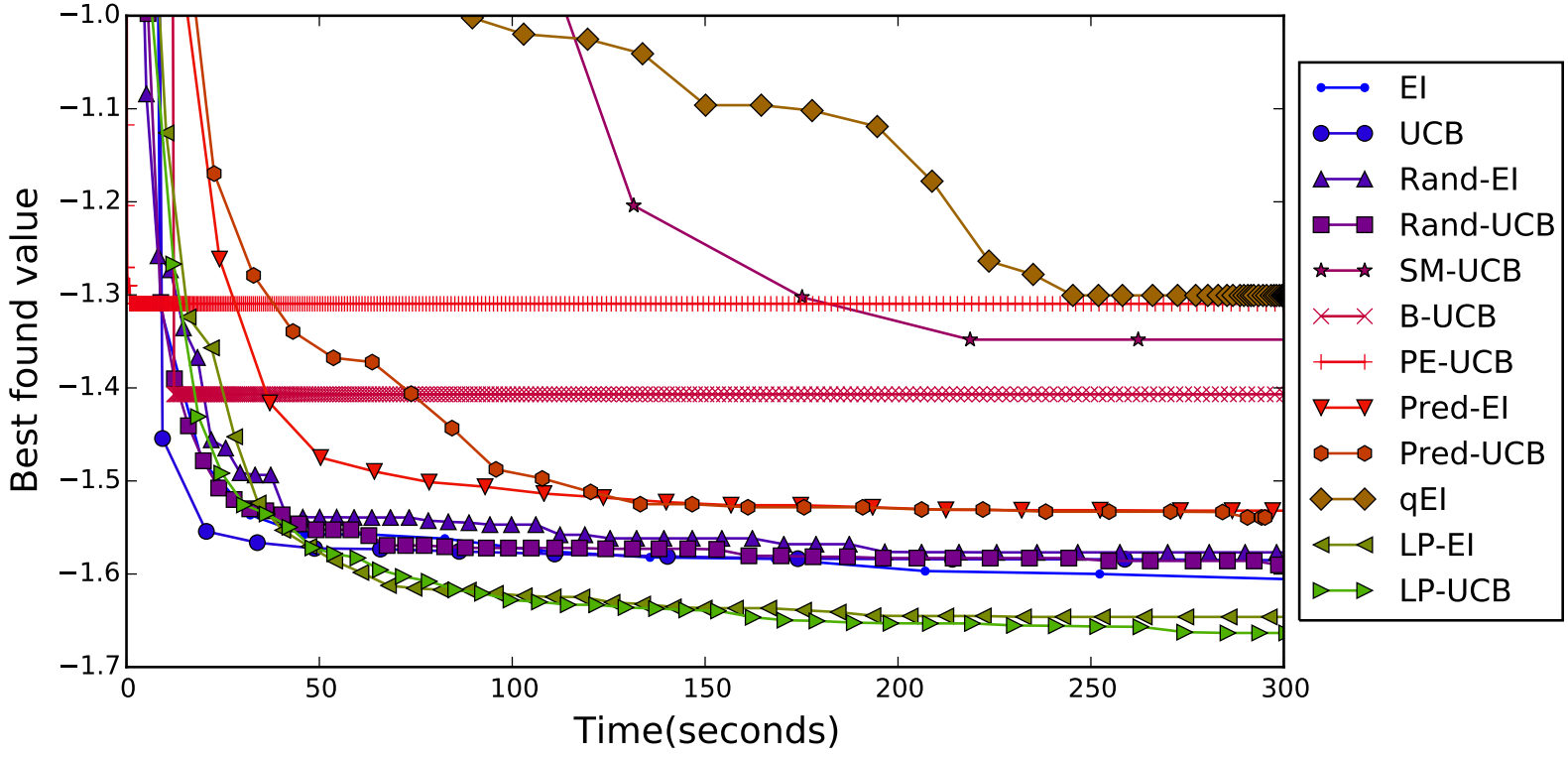
Best (average) result for some given time budget.

d	n_b	EI	UCB	Rand-EI	Rand-UCB	SM-UCB	B-UCB
2	5	0.31±0.03	0.32±0.06	0.32±0.05	**0.31±0.05**	1.86±1.06	0.56±0.03
	10			0.65±0.32	0.79±0.42	4.40±2.97	**0.59±0.00**
	20			0.67±0.31	0.75±0.32	-	0.57±0.01
5	5	8.84±3.69	11.89±9.44	9.19±5.32	10.59±5.04	137.2±113.0	**6.01±0.00**
	10			1.74±1.47	2.20±1.85	108.7±74.38	3.77±0.00
	20			2.18±2.30	2.76±3.06	-	2.53±0.00
10	5	559.1±1014	1463±1803	**690.5±947.5**	1825±2149	9e+04±7e+04	2098±0.00
	10			200.9±455.9	1149±1830	9e+04±1e+05	857.8±0.00
	20			639.4±1204	385.9±642.9	-	1656±0.00

d	n_b	PE-UCB	Pred-EI	Pred-UCB	qEI	LP-EI	LP-UCB
2	5	0.99±0.74	0.41±0.15	0.45±0.16	1.53±0.86	0.35±0.11	**0.31±0.06**
	10	0.66±0.29	1.16±0.70	1.26±0.81	3.82±2.09	0.66±0.48	0.69±0.51
	20	0.75±0.44	1.28±0.93	1.34±0.77	-	**0.50±0.21**	0.58±0.21
5	5	123.5±81.43	10.43±4.88	11.77±9.44	15.70±8.90	11.85±5.68	10.85±8.08
	10	120.8±78.56	9.58±7.85	11.66±11.48	17.69±9.04	3.88±4.15	**1.88±2.46**
	20	98.60±82.60	8.58±8.13	10.86±10.89	-	6.53±4.12	**1.44±1.93**
10	5	2e+05±2e+05	793.0±1226	1412±3032	-	1881±1176	1194±1428
	10	6e+04±8e+04	442.6±717.9	1725±3205	-	1042±1562	**100.4±338.7**
	20	5e+04±4e+04	1091±1724	2231±3110	-	1249±1570	**20.75±50.12**

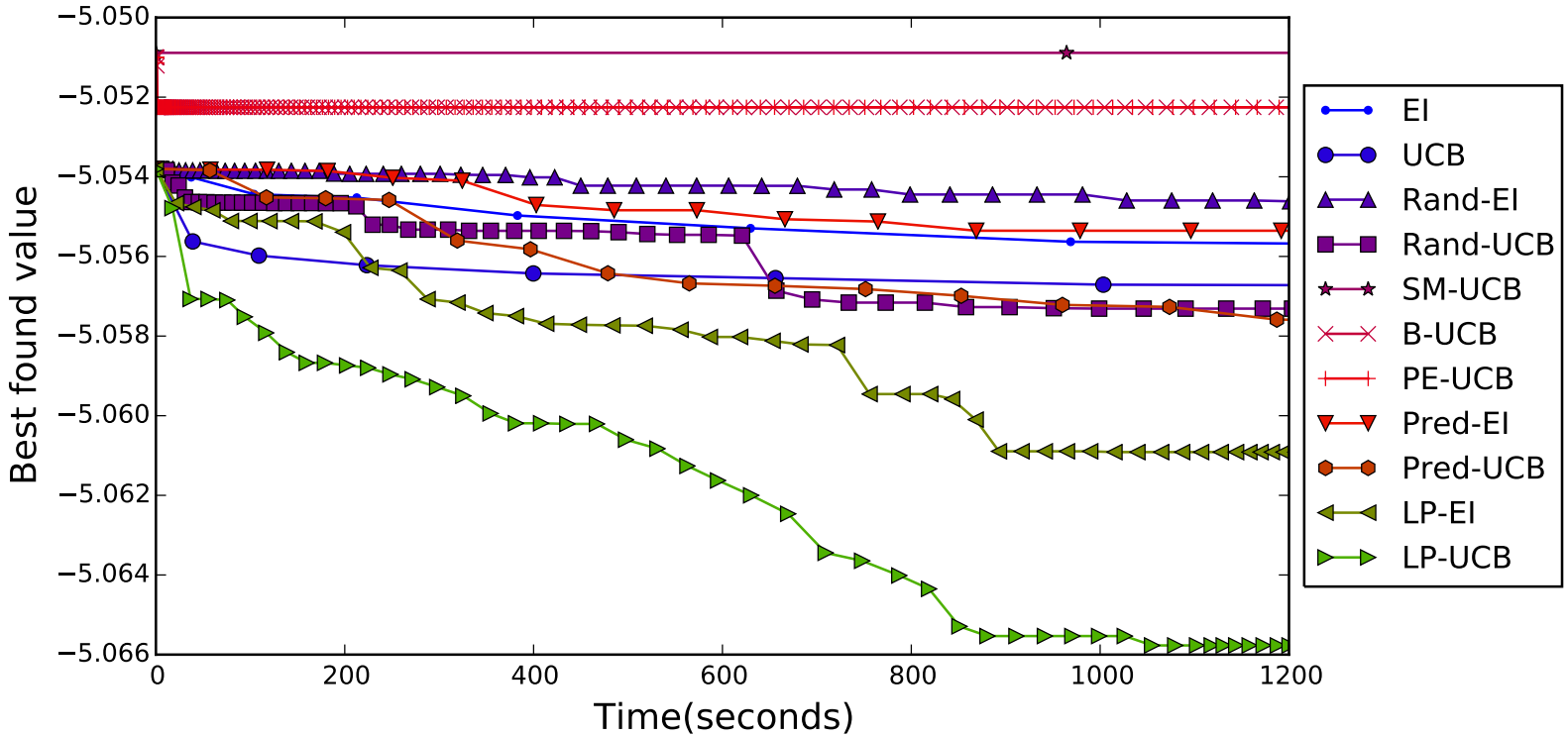
2D experiment with ‘large domain’

Comparison in terms of the wall clock time



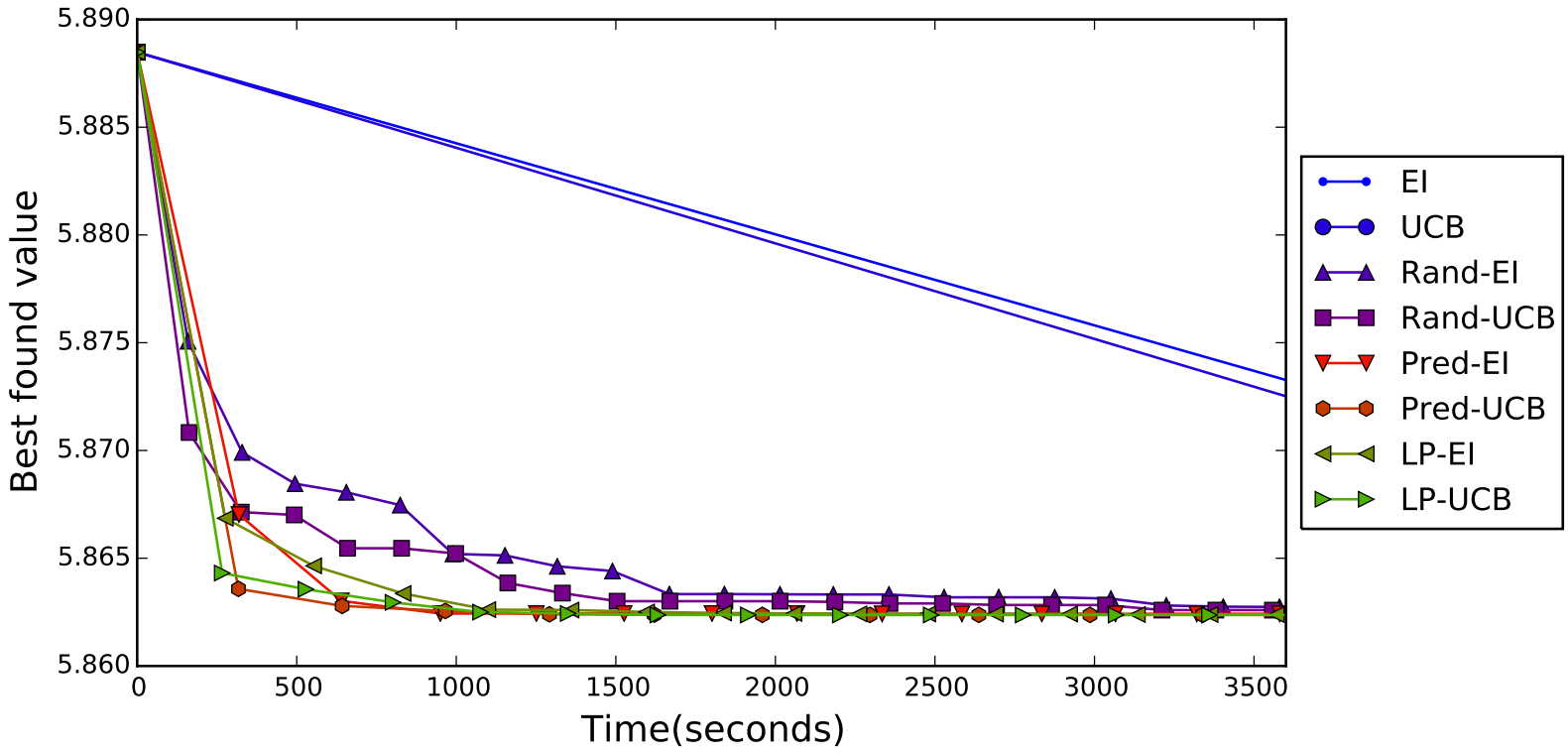
Maximizing gene translation

- Maximization of a 70 dimensional surface representing the efficiency of hamster cells producing proteins.

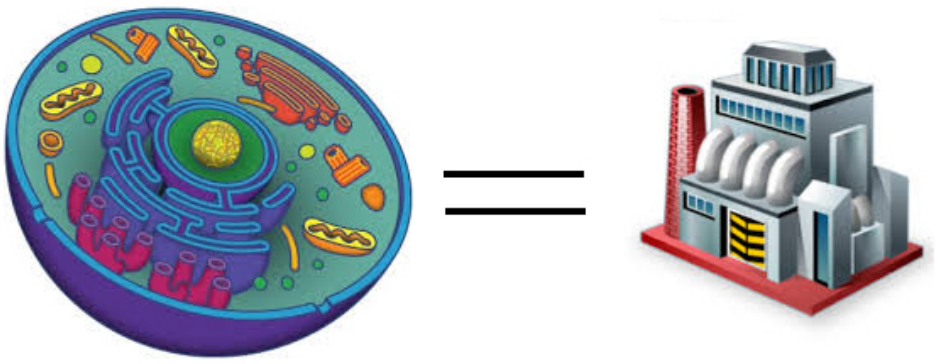


Support Vector Regression

- Minimization of the RMSE on a test set over 3 parameters.
- 'Physiochemical' properties of protein tertiary structure?.
- 45730 instances and 9 continuous attributes.



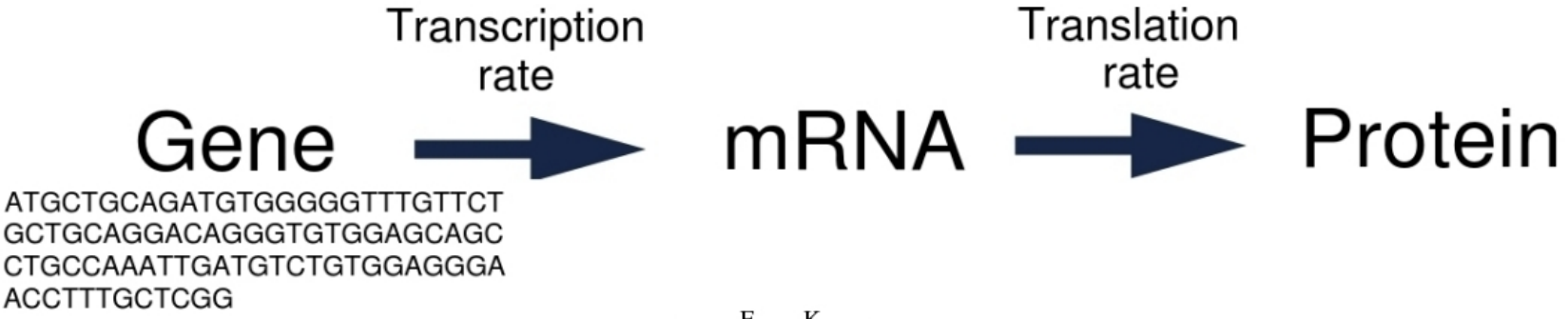
Synthetic gene design

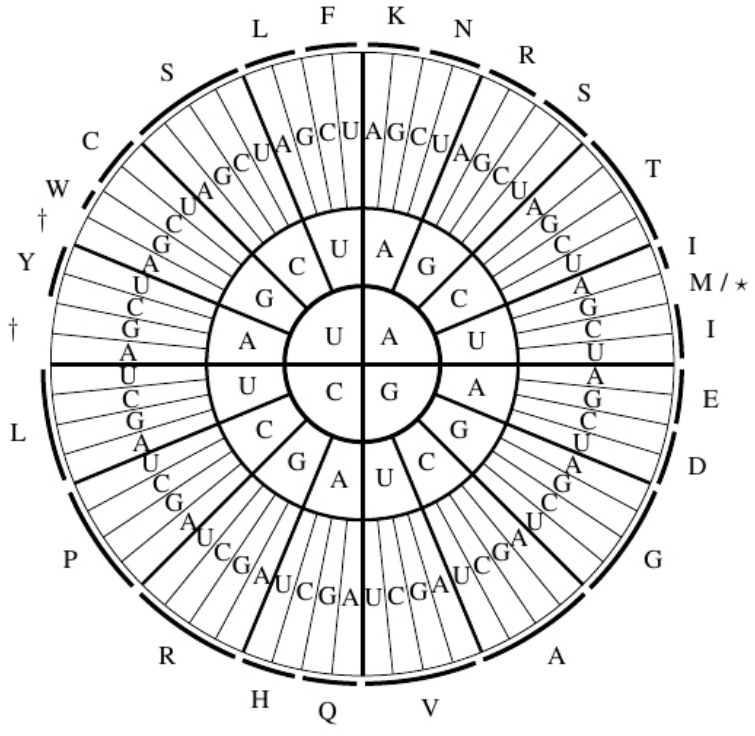


- Use mammalian cells to make protein products.
- Control the ability of the cell-factory to use synthetic DNA.

Optimize genes (ATTGGTUGA...) to best enable the cell-factory to operate most efficiently [González et al. 2014].

Surrogate model for the cell





How to design a synthetic gene?

A good model is crucial

Gene sequence features $\rightarrow$ protein production efficiency.

Bayesian Optimization principles for gene design

do:

1. Build a GP model as an emulator of the cell behavior.
2. Obtain a set of gene design rules (features optimization).
3. Design one/many new gene/s coherent with the design rules.
4. Test genes in the lab (get new data).

until the gene is optimized (or the budget is over...).

Model as an emulator of the cell behavior

Model inputs

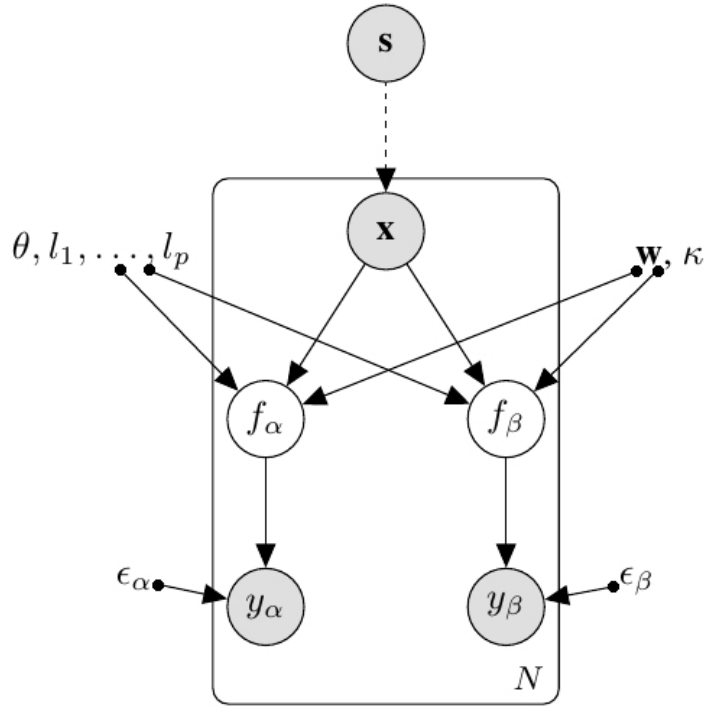
Features ($\mathbf{x}_i$) extracted gene sequences ($\mathbf{s}_i$): codon frequency, cai, gene length, folding energy, etc.

Model outputs

Transcription and translation rates $\mathbf{f} := (f_\alpha, f_\beta)$.

Model type

Multi-output Gaussian process $\mathbf{f} \approx \mathcal{GP}(\mathbf{m}, \mathbf{K})$ where $\mathbf{K}$ is a corregionalization covariance for the two-output model (+ SE with ARD).



The correlation in the outputs help!

Obtaining optimal gene design rules

Maximize the averaged EI [Swersky et al. 2013]

$$\alpha(\mathbf{x}) = \bar{\sigma}(\mathbf{x})(-u\Phi(-u) + \phi(u))$$

where $u = (y_{max} - \bar{m}(\mathbf{x}))/\bar{\sigma}(x)$ and

$$\bar{m}(\mathbf{x}) = \frac{1}{2}\sum_{l=\alpha,\beta} \mathbf{f}_*(\mathbf{x}),\ \bar{\sigma}^2(\mathbf{x}) = \frac{1}{2^2}\sum_{l,l'=\alpha,\beta} (\mathbf{K}_*(\mathbf{x},\mathbf{x}))_{l,l'}.$$

A batch method is used when several experiments can be run in parallel

Designing new genes coherent with the optimal design rules

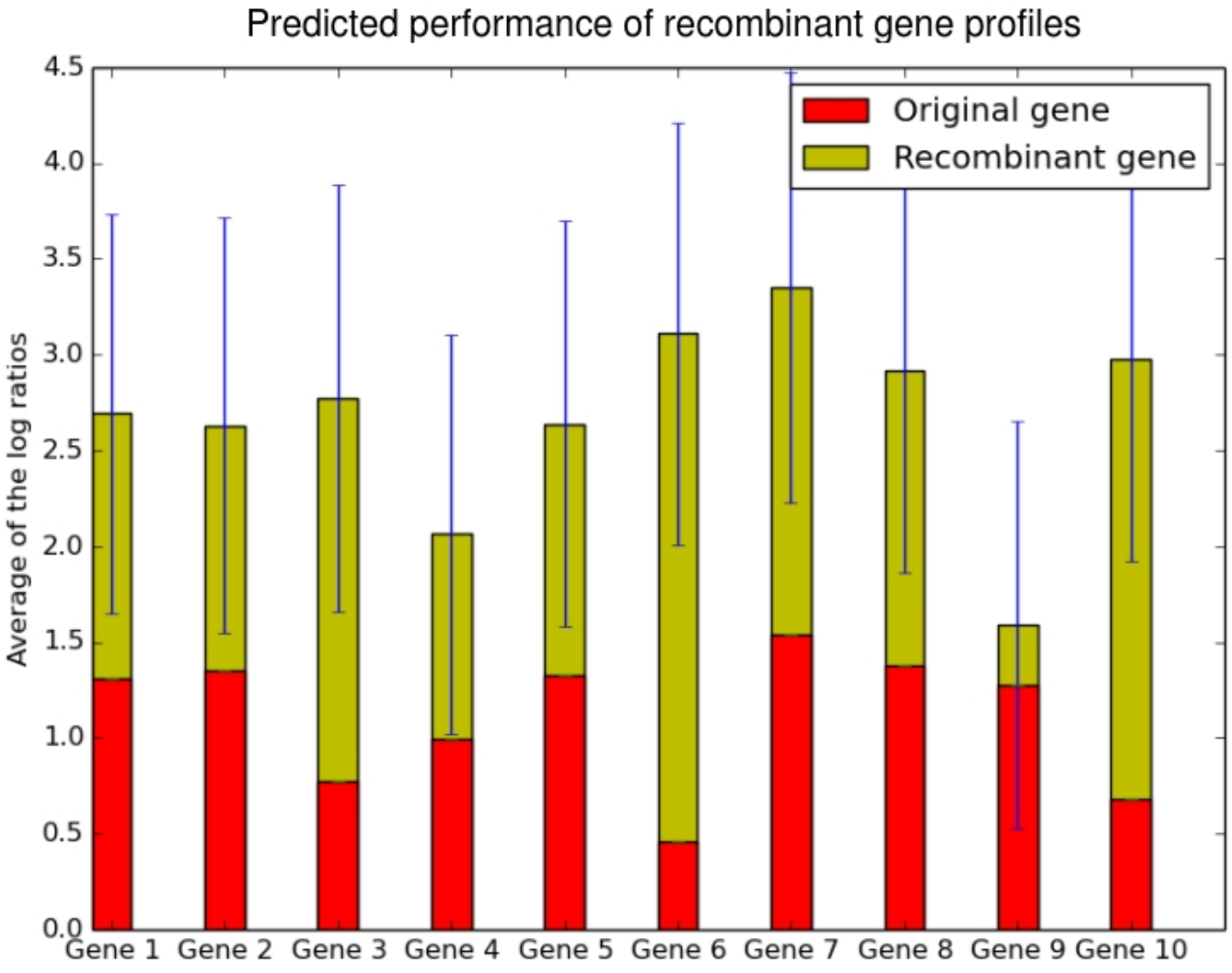
Simulating-matching approach:

1. Simulate genes 'coherent' with the target (same amino-acids).
2. Extract features.
3. Rank synthetic genes according to their similarity with the 'optimal' design rules.

Ranking criterion: $eval(\mathbf{s}|\mathbf{x}^\star) = \sum_{j=1}^{p} w_j |\mathbf{x}_j - \mathbf{x}_j^\star|$

- $\mathbf{x}^\star$: optimal gene design rules.
- $\mathbf{s}$, $\mathbf{x}_j$ generated 'synonyms sequence' and its features.
- w_j: weights of the p features (inverse length-scales of the model covariance).

Results for 10 low-expressed genes



Wrapping up

- BO is fantastic tool for parameter optimization in ML and experimental design.
- The model and acquisition function are the two most important bits.
- Many useful extensions for BO.
- To scale BO is a current challenge.
- Software available!

Many thanks to!

Working on BO:

- Neil Lawrence
- Philipp Hennig
- Zhenwen Dai
- Mike Osborne

Collaborators at CBE:

- David James
- Joseph Longworth
- Mark Dickman

Picture source: http://peakdistrictcycleways.co.uk

Use Bayesian optimization!