Gaussian Processes and Bayesian Optimization

Evgeny Burnaev

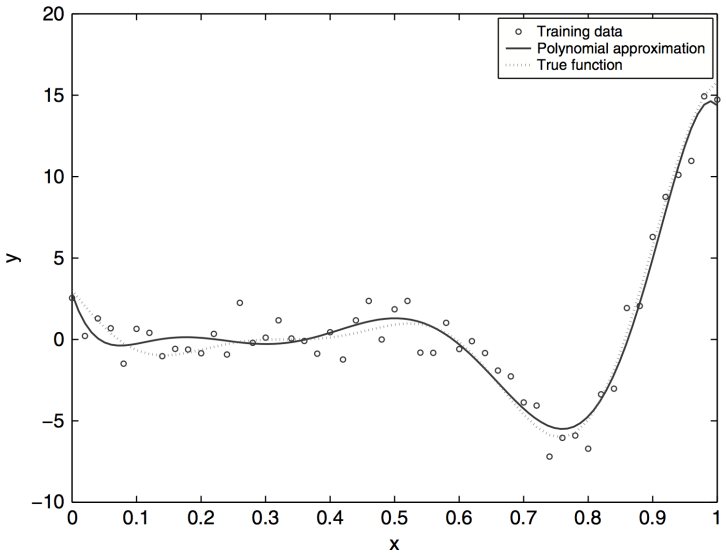
Head of ADASE group, Skoltech

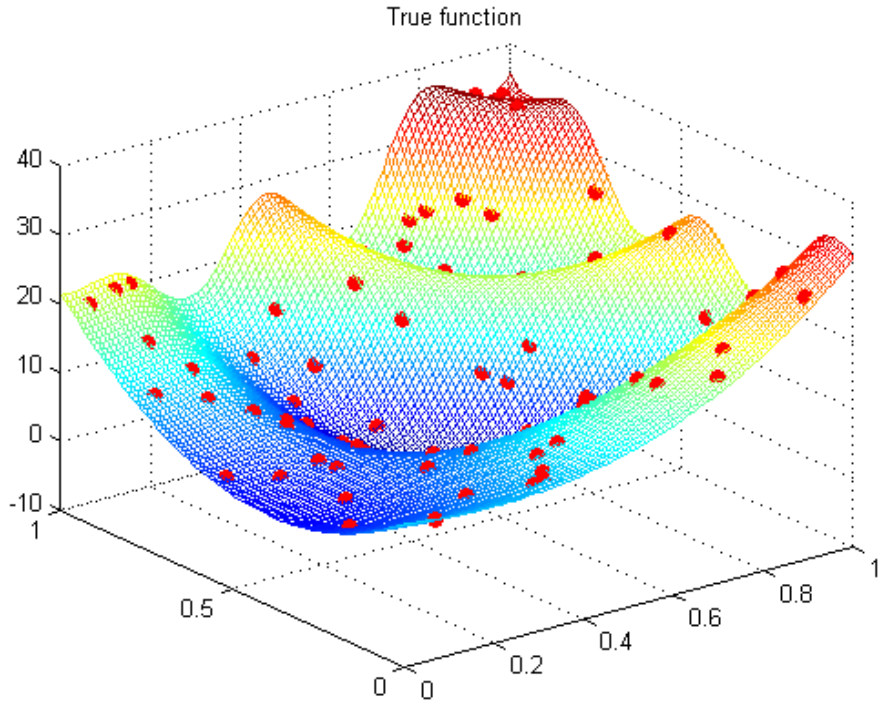
joint with HSE, IITP

Outline

The problem

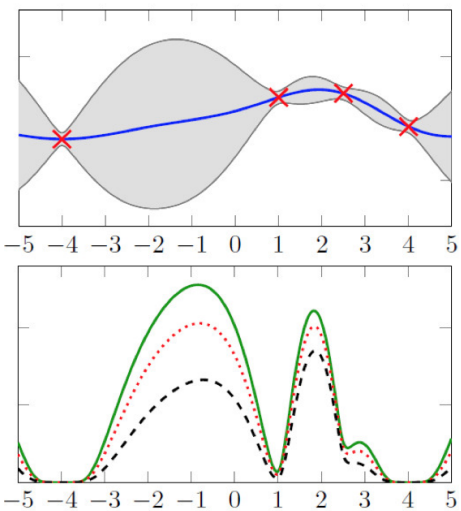
- Learn scalar function $f(\mathbf{x})$ of vector $\mathbf{x}$
- We have (possibly noisy) sample $S_m = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$

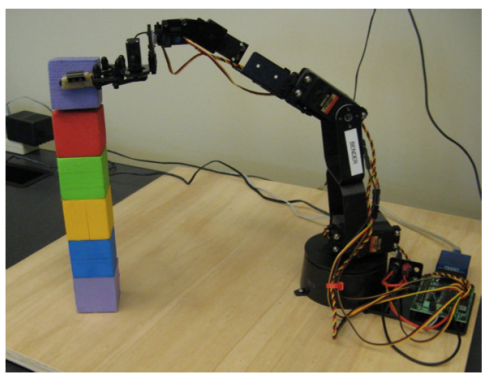


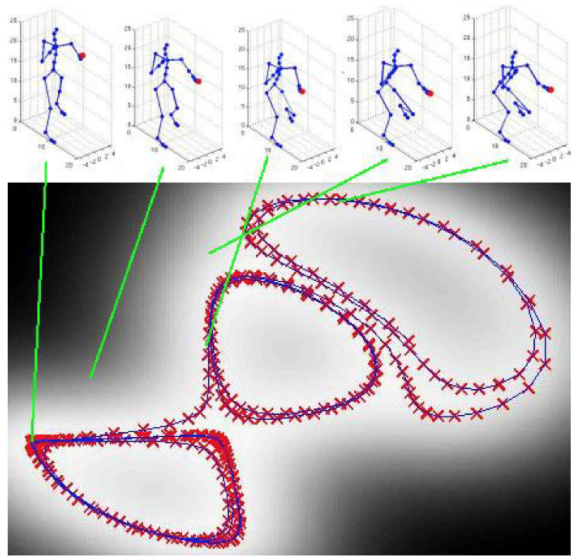


Applications

- **Real-valued regression**:
  - Model-based predictive control: predict yield, quality, losses, etc.
  - Surrogate surfaces for optimization or simulation
  - Robotics: target state $\rightarrow$ required torques
- **Classification**:
  - Recognition: handwritten digits on payment docs, etc.
  - Detection: fraud, screening in chemoinformatics
- **Ordinal Regression**:
  - Disease harmfulness prediction
  - User ratings (movies, shops, restaurants)

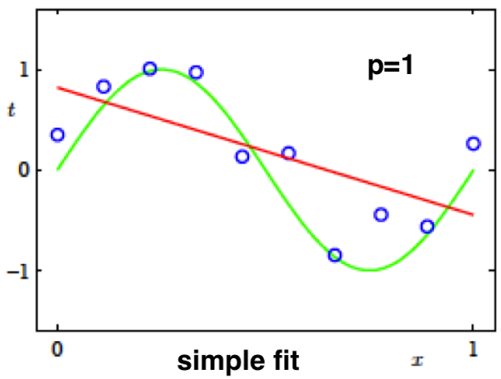


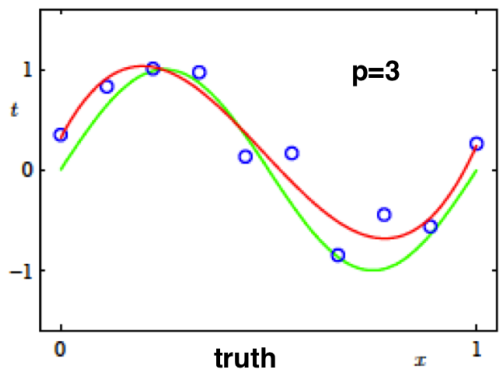


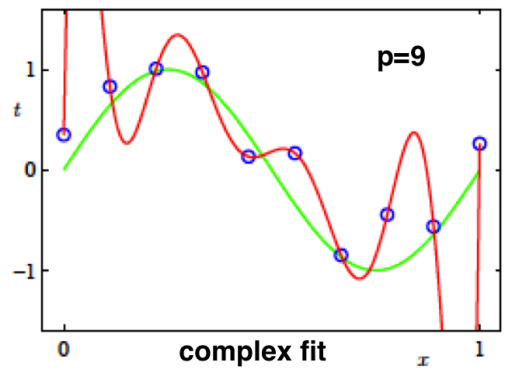


Model Complexity

- Data generation process is often unknown and can be complex:



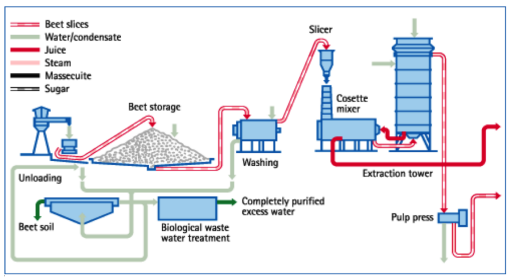




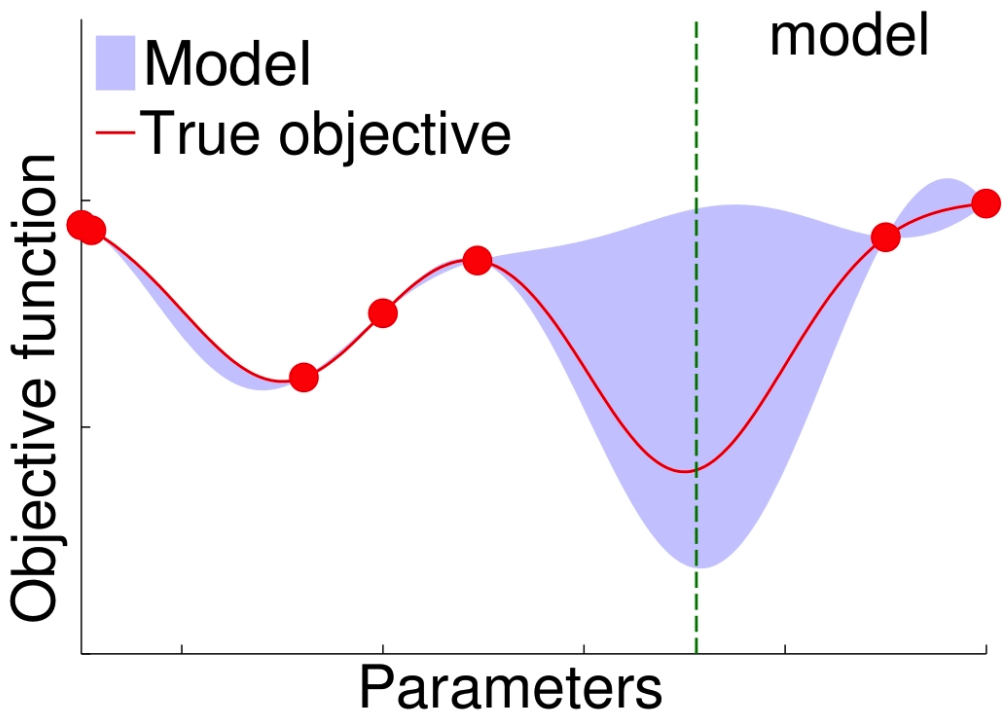
- **Problems**:
  - Fitting complicated models can be hard
  - How to find an appropriate model?
  - How to avoid over-fitting?

Optimize efficiency of sugar extraction

- **Problem**: Predicting Profit
  - **Object**: production of one batch of a product
  - **Input** $\mathbf{x}$: beet chips shape, quality, temperature, sugar content and flux; wash water temperature, pH and flux; the temperature inside the diffuser, etc.
  - **Output** y: costs, losses, efficiency of sugar extraction
- **Challenges**:
  - Heterogeneous data and noise,
  - Large volumes of high-dimensional stream data,
  - Missing values, outliers, etc.
- **Knowing error bars is very important**:
  - Setting $\mathbf{x}_1 \rightarrow$ profit of 40 ± 10 units
  - Setting $\mathbf{x}_2 \rightarrow$ profit of 60 ± 40 units
  - **Which are the best setting, $\mathbf{x}_1$ or $\mathbf{x}_2$?**



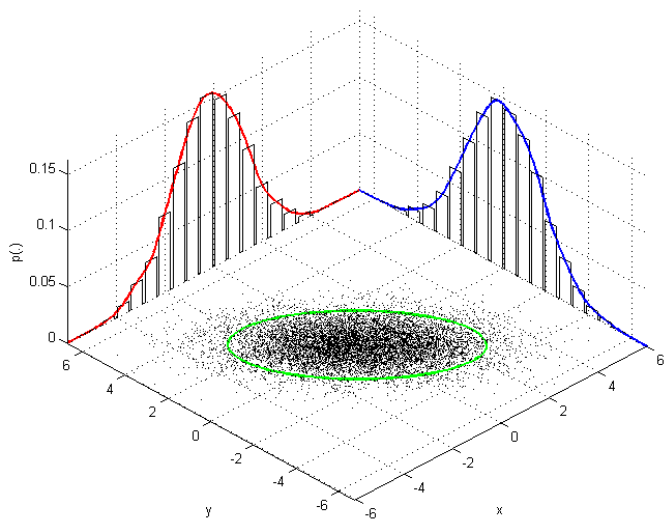
- In high-dimensional case we need many functions evaluations to be certain in results
- Often each evaluation is costly, e.g. in case of experiments



- Error bars are needed to see if a region is still promising

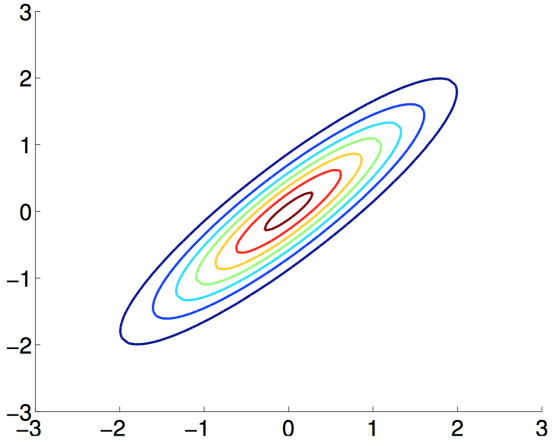
- Parametric family of functions $f(\mathbf{x}; \boldsymbol{\theta})$
- Define a prior over $\boldsymbol{\theta}$
- Perform predictions and estimate uncertainty
- For flexible models we a.s. get intractable integrals over $\boldsymbol{\theta}$

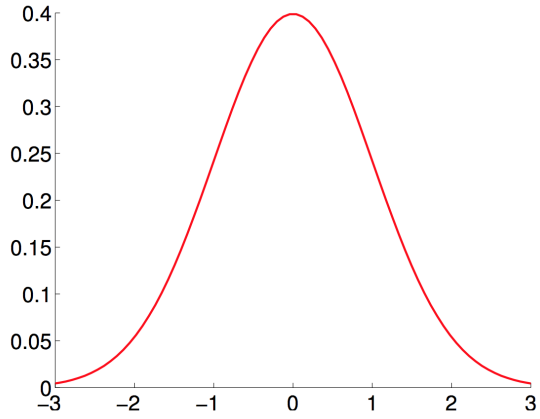
For Gaussian case usually everything is explicit

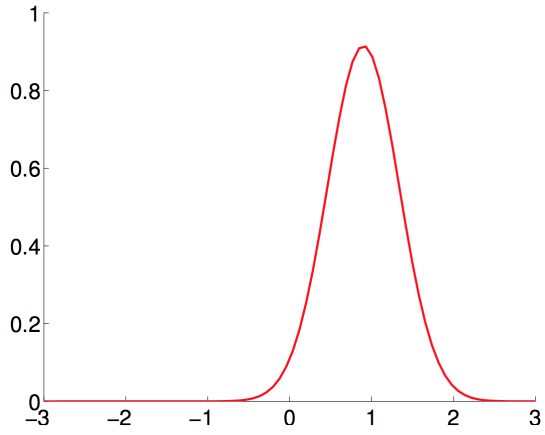


Solution of complex ML problems with simple Gaussian models?

The Gaussian Distribution







$$p(\mathbf{f}_1, \mathbf{f}_2) \sim \mathcal{N}(\mathbf{f}_1, \mathbf{f}_2 | \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Joint

$$p(\mathbf{f}_1)$$

Marginal

$$p(\mathbf{f}_1 | \mathbf{f}_2)$$

Conditional

The **marginal** and **conditional** distributions are also Gaussians:

$$\begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{pmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{12}^{\mathrm{T}} & \boldsymbol{\Sigma}_{22} \end{bmatrix} \right)$$

$$p(\mathbf{f}_1) = \int p(\mathbf{f}_1, \mathbf{f}_2) d\mathbf{f}_2 = \mathcal{N}\left(\mathbf{f}_1 | \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11}\right)$$

$$p(\mathbf{f}_1 | \mathbf{f}_2) = \mathcal{N}\left(\mathbf{f}_1 | \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{f}_2 - \boldsymbol{\mu}_2), \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{12}^{\mathrm{T}}\right)$$

- **Hypothesis set** $\mathcal{F} \subset Y^X$ is a subset of functions out of which the learner selects his/her hypothesis
  - represents a prior knowledge about the task at hand
  - depends on available features
- Typical examples (from statistics)
  - Sobolev-type classes:

$$\mathcal{F}_L^k = \left\{ f : \int \left\| \frac{\partial^k f(\mathbf{x})}{\partial \mathbf{x}^k} \right\|^2 d\mathbf{x} \leq L \right\}$$

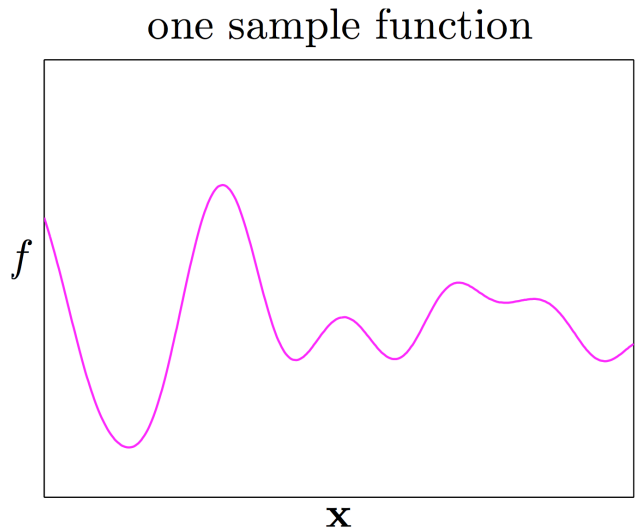
  - Lipschitz classes:

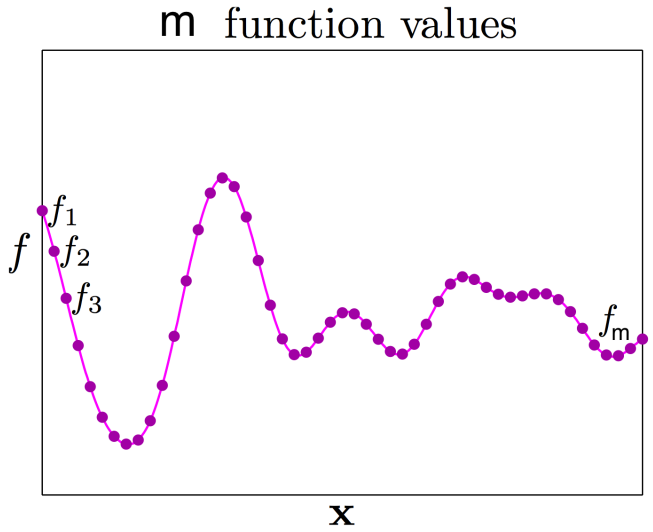
$$\mathcal{F}_L = \{ f : |f(\mathbf{x}) - f(\mathbf{x}')| \leq L\rho(\mathbf{x}, \mathbf{x}') \}$$

- How to impose regularity on a function in Bayesian case?

What is a Gaussian process?

- Continuous stochastic process — random functions — a set of random variables $f(\mathbf{x})$ indexed by a continuous variable $\mathbf{x}$





- Let us denote by
  - $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ a set of d-dimensional inputs,
  - $\mathbf{f} = \{f_1, \ldots, f_m\}$ a set of random function values $f_i = f(\mathbf{x}_i)$
- **GP**: Any set of function variables $\{f_i\}_{i=1}^m$ has joint Gaussian distribution

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$$

- Conditional model — density of inputs is not modeled

Where do the mean value $\boldsymbol{\mu}$ and covariance matrix $\mathbf{K}$ come from?

- Mean value is constructed from a priori given deterministic function

$$\boldsymbol{\mu} = \{\mu_i\} = \{\mu(\mathbf{x}_i)\}$$

- Covariance matrix is constructed from covariance function

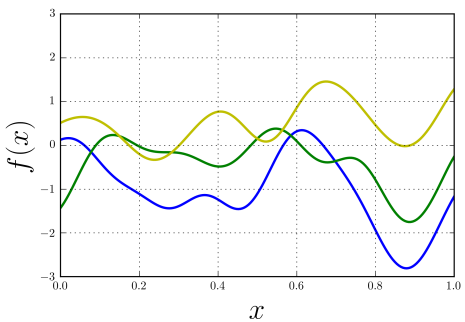
$$\mathbf{K} = \{\mathbf{K}_{ij}\} = \{K(\mathbf{x}_i, \mathbf{x}_j)\}$$

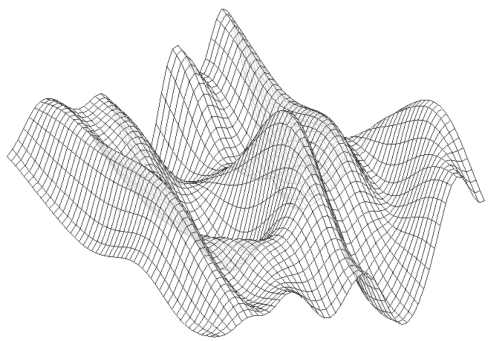
- Covariance function characterizes covariance between points in the process

$$K(\mathbf{x}, \mathbf{x}') = \operatorname{cov}(f(\mathbf{x}), f(\mathbf{x}')) = \mathbb{E}\left(f(\mathbf{x}) - \mu(\mathbf{x})\right)\left(f(\mathbf{x}') - \mu(\mathbf{x}')\right)$$

- Must produce positive semidefinite covariance matrices $\mathbf{v}^{\mathrm{T}}\mathbf{K}\mathbf{v} \geq 0$

$$f(\mathbf{x}) \sim \mathcal{GP}(\cdot|\mu(\mathbf{x}), K(\mathbf{x}, \mathbf{x}'))$$





Example: Bayesian linear regression

- In order to define $f(\mathbf{x}) \sim \mathcal{GP}(\cdot)$ we have to define a mean function $\mu(\mathbf{x})$ and a covariance function $K(\mathbf{x}, \mathbf{x}')$
- Bayesian linear model

$$f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}) + b = \sum_{i=1}^{p} w_i \phi_i(\mathbf{x}) + b$$

$$\mathbf{w} \sim \mathcal{N}(0, \sigma_f^2 \cdot \mathrm{I}_p)$$

$$b \sim \mathcal{N}(0, \sigma_1^2)$$

- We can easily obtain that

$$\mu(\mathbf{x}) = 0, \ \operatorname{cov}(f(\mathbf{x}), f(\mathbf{x}')) = \sigma_f^2 \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}') + \sigma_1^2$$

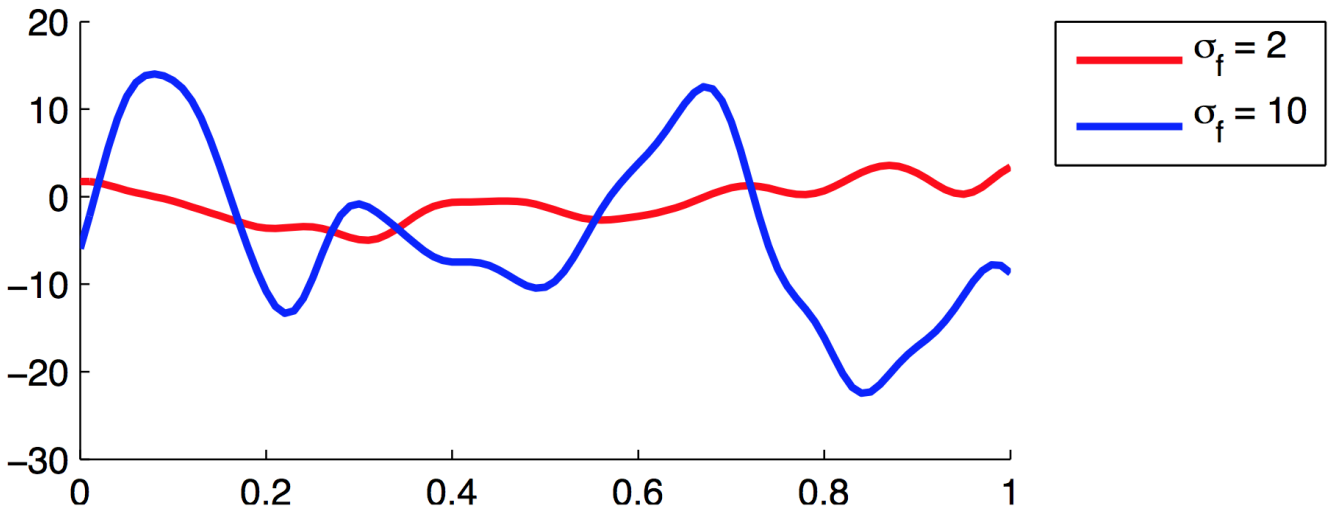
- Kernel parameters σ_f^2 and σ_1^2 are hyper-parameters in the Bayesian hierarchical model
- More interesting kernels come from models with a large or infinite feature space $p \gg 1$. Because feature weights $\mathbf{w}$ are integrated out, this is computationally no more expensive

Example: squared-exponential kernel

- An infinite number of radial-basis functions can give

$$K(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left\{ -\sum_{i=1}^{d} \frac{(x_i - x_i')^2}{2r_i^2} \right\}$$

  the most commonly-used kernel in machine learning
- *A GP need not use the Gaussian kernel. There are a lot of other choices*
- Consider $\mathbf{x} = \mathbf{x}' \rightarrow$ marginal function variance is σ_f^2

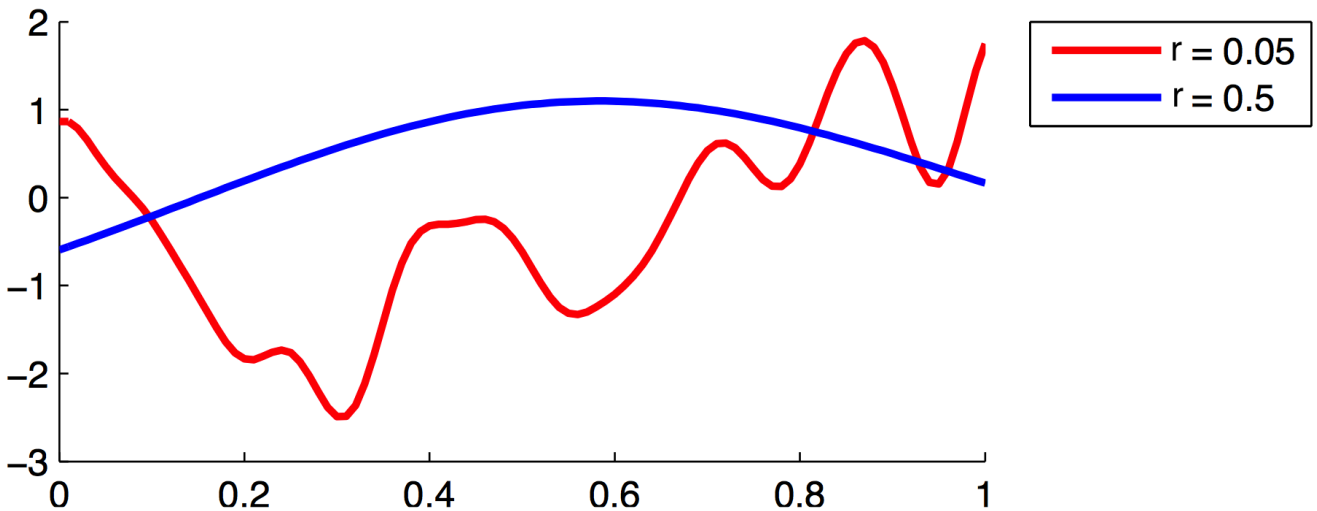


Example: squared-exponential kernel

- The r_i parameters give the overall lengthscale in dimension i

$$K(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left\{ -\sum_{i=1}^{d} \frac{(x_i - x_i')^2}{2r_i^2} \right\}$$

- Typical distance between peaks $\approx r$



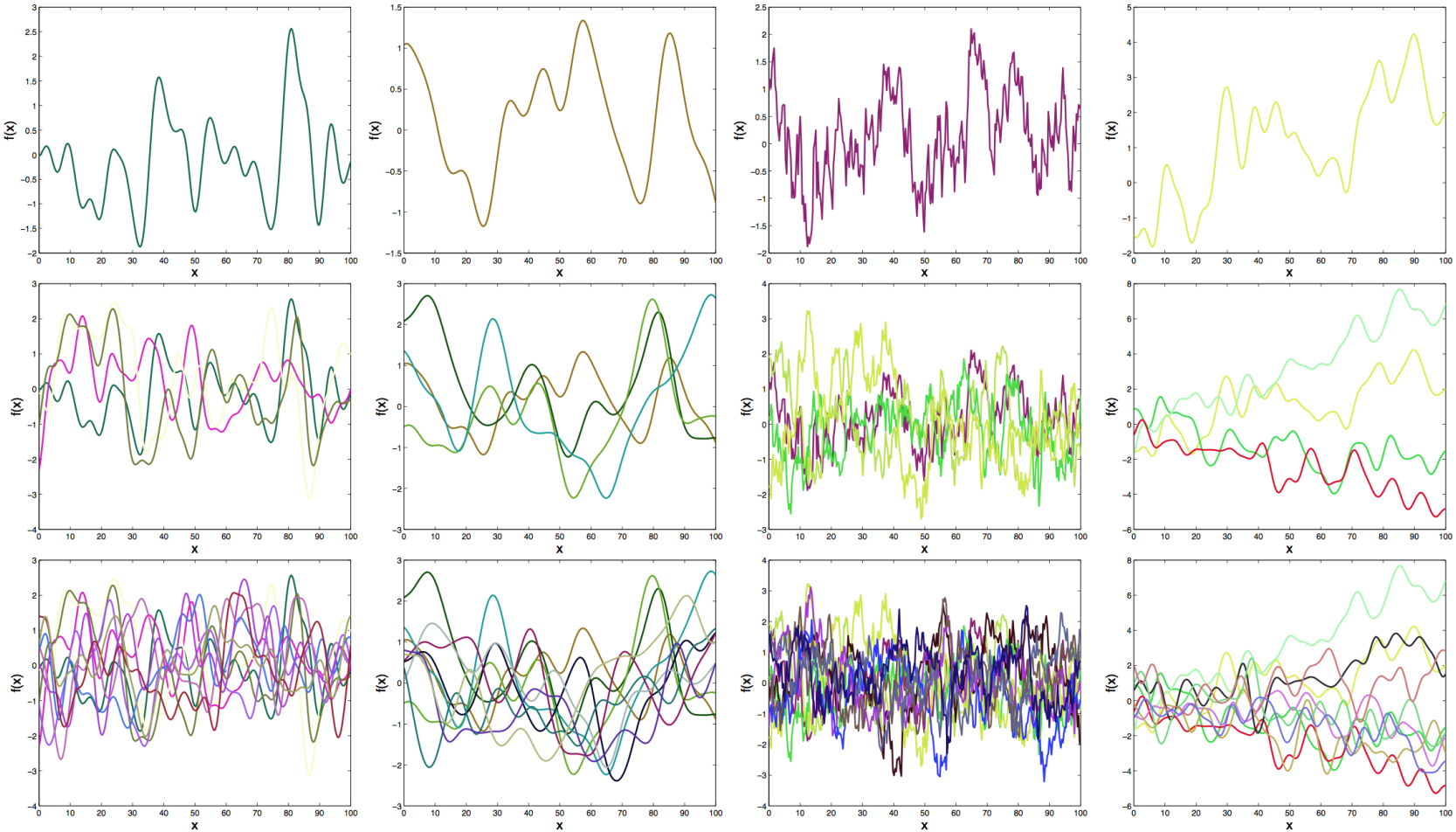
Example: Gaussian process covariance functions

- More general example of covariance function

$$K(x, x') = \sigma_f^2 \exp\left\{ - \left(\frac{|x - x'|}{r} \right)^{\alpha} \right\} + \sigma_1^2 + \sigma_2^2 \delta(x - x')$$

- As usual covariance function parameters are interpretable
  - σ_f^2 — marginal function variance
  - σ_1^2 — variance of bias
  - σ_2^2 — noise variance
  - r — lengthscale
  - $\alpha \geq 1$ — roughness
- Once the mean and covariance functions are defined, everything else about GPs follows from the basic rules of probability applied to multivariate Gaussians

Samples from GPs with different $K(x, x')$



Example: Matérn class of covariances

$$K(x, x') = \frac{\sigma_f^2 2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}|x - x'|}{\lambda} \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}|x - x'|}{\lambda} \right),$$

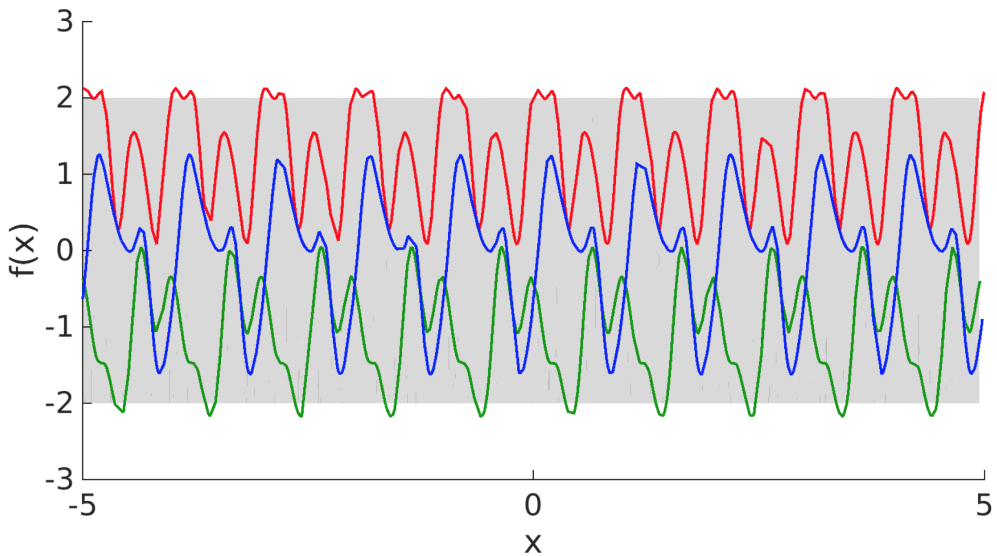
where K_ν is a modified Bessel function

- Stationary, isotropic
- $\nu \to \infty$: squared-exponential covariance
- Finite ν: much rougher sample functions
- $\nu = 1/2$: $K(x, x') = \sigma_f^2 \exp(-|x - x'|/r)$, very rough sample functions (seminar)

Example: Nonstationary covariances

- Linear covariance: $K(x, x') = \sigma_f^2 \cdot xx' + \sigma_1^2$
- Periodic covariance (seminar):

$$K(x, x') = \sigma_f^2 \cdot \exp\left(-\frac{2\sin^2\left(\frac{x-x'}{2}\right)}{r^2}\right)$$



- Neural network covariance

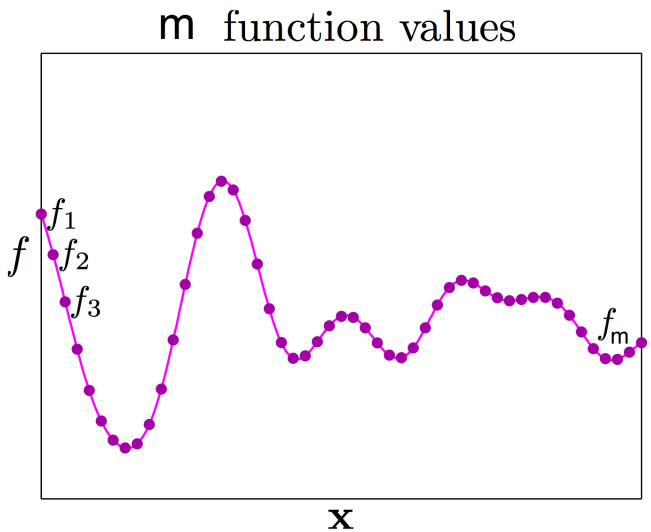
Example: Constructing new covariances from old

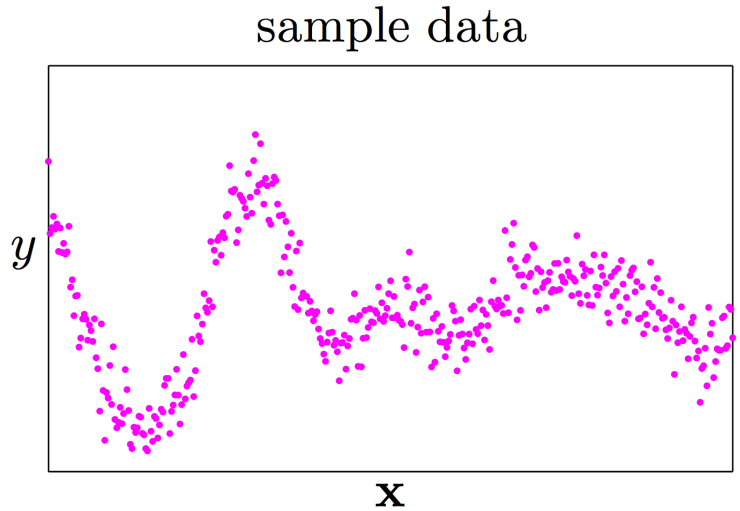
There are several ways to combine covariances (seminar)

- Sum: $K(x, x') = K_1(x, x') + K_2(x, x')$
- Product: $K(x, x') = K_1(x, x') \cdot K_2(x, x')$
- Convolution: $K(x, x') = \int dz dz' h(x, z) K(z, z') h(x', z')$ (blurring of process with kernel h)

- Training data set $S_m = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$
- Model:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i,$$
$$f \sim \mathcal{GP}(\cdot|0, K)$$
$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ is a white noise}$$





- Training data set $S_m = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$
- Model:

$$
\begin{aligned}
y_i &= f(\mathbf{x}_i) + \varepsilon_i, \\
f &\sim \mathcal{GP}(\cdot|0, K) \\
\varepsilon_i &\sim \mathcal{N}(0, \sigma^2) \text{ is a white noise}
\end{aligned}
$$

- The prior is

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$

- The noise model, or likelihood is

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}\left(\mathbf{y}|\mathbf{f}, \sigma^2 \mathrm{I}_m\right)$$

- Integrating over the function variables $\mathbf{f}$ we get the marginal likelihood

$$p(\mathbf{y}) = \int d\mathbf{f} p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}) d\mathbf{f} = \mathcal{N}(\mathbf{0},\, \mathbf{K} + \sigma^2 \mathrm{I}_m)$$

- Let us denote input test point as $\mathbf{x}_*$, and output

$$y_* = f_* + \varepsilon_*, \;\; f_* = f(\mathbf{x}_*)$$

- Consider joint training and test marginal likelihood

$$p(\mathbf{y}, f_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma^2 \mathrm{I}_m & \mathbf{k}_* \\ \mathbf{k}_*^{\mathrm{T}} & K_{**} \end{bmatrix}\right),$$

  where $\mathbf{k}_* = \{K(\mathbf{x}_*, \mathbf{x}_i)\}_{i=1}^m$ and $K_{**} = K(\mathbf{x}_*, \mathbf{x}_*)$
- What we know about noiseless value $f(\mathbf{x}_*)$?

- Joint training and test marginal likelihood

$$p(\mathbf{y}, f_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma^2 \mathrm{I}_m & \mathbf{k}_* \\ \mathbf{k}_*^{\mathrm{T}} & K_{**} \end{bmatrix}\right),$$

  where $\mathbf{k}_* = \{K(\mathbf{x}_*, \mathbf{x}_i)\}_{i=1}^m$ and $K_{**} = K(\mathbf{x}_*, \mathbf{x}_*)$

- Condition on training outputs we get

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^{\mathrm{T}}[\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1}\mathbf{y}, \;\; \sigma_*^2 = K_{**} - \mathbf{k}_*^{\mathrm{T}}[\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1}\mathbf{k}_*$$

- In fact μ_* has the form

$$\mu_* = \sum_{i=1}^{m} \alpha_i K(\mathbf{x}, \mathbf{x}_i), \;\; \boldsymbol{\alpha} = [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1}\mathbf{y}$$

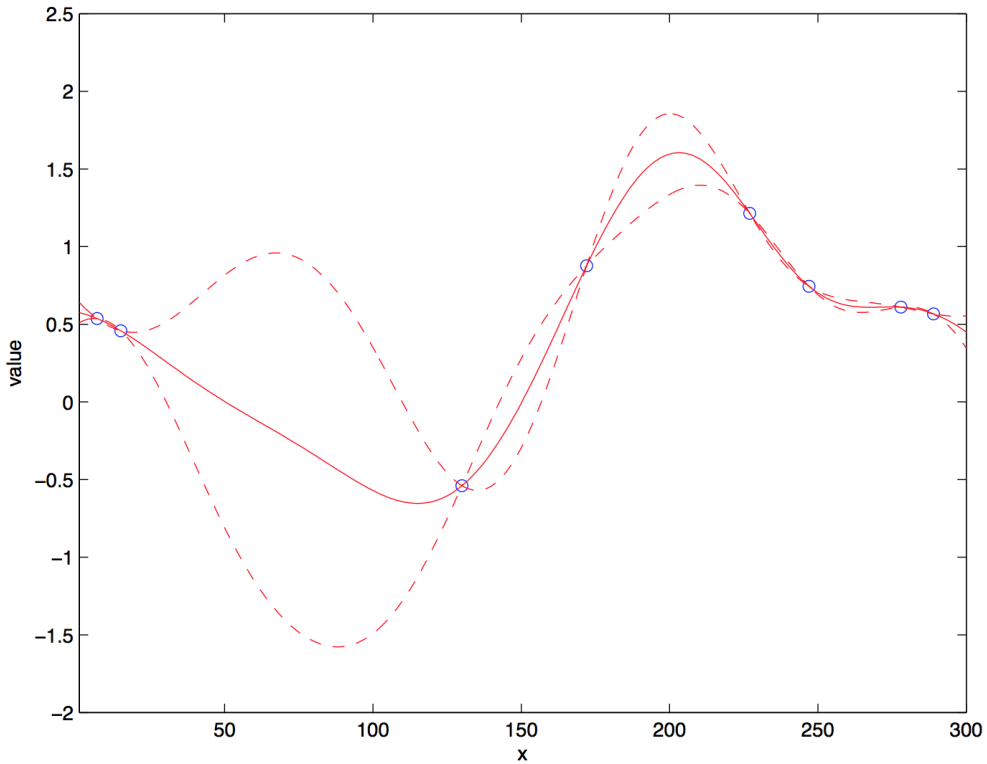
- Computational complexity: $O(m^3)$ for inversion in $\boldsymbol{\alpha}$, $O(m)$ for μ_* and $O(m^2)$ for σ_*^2

- Conditional distribution

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^{\mathrm{T}}[\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1}\mathbf{y},\ \sigma_*^2 = K_{**} - \mathbf{k}_*^{\mathrm{T}}[\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1}\mathbf{k}_*$$

- What we know about noiseless value $f_* = f(\mathbf{x}_*)$? Let us assume that $\sigma = 0$ (interpolation)



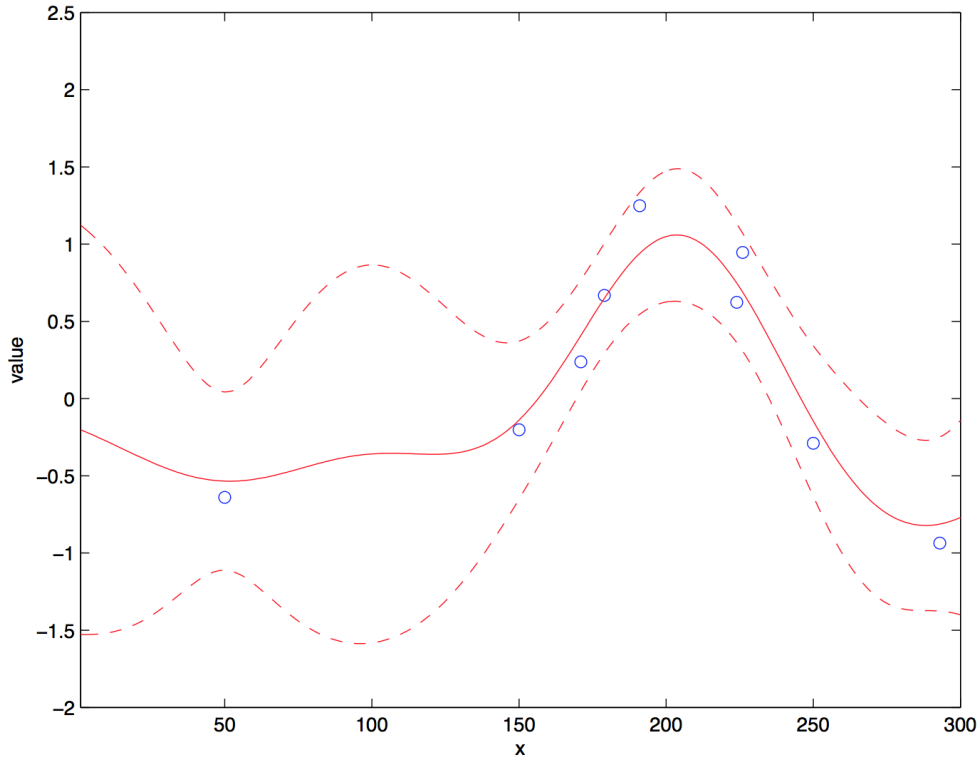
Smoothing

- Conditional distribution

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^{\mathrm{T}}[\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1}\mathbf{y}, \;\; \sigma_*^2 = K_{**} - \mathbf{k}_*^{\mathrm{T}}[\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1}\mathbf{k}_*$$

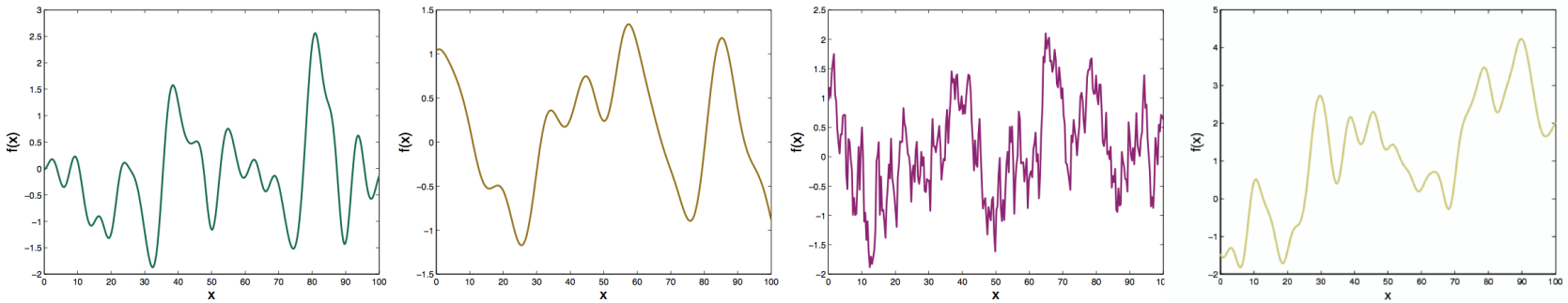
- What we know about noiseless value $f_* = f(\mathbf{x}_*)$?



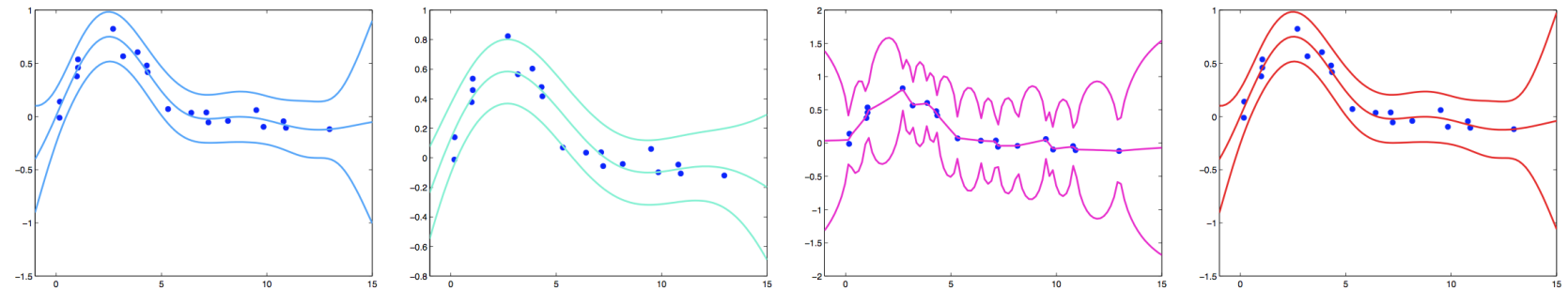
- $p(y_*|\mathbf{y}) = \mathcal{N}\left(y_*|\mu_*, \sigma_*^2 + \sigma^2\right)$ predicts what we'll see next

Prediction using GPs with different $K(x, x')$

A sample from the prior for each covariance function:



Corresponding predictions, mean with two standard deviations:



- Advantage of the probabilistic GP framework — ability to choose hyperparameters and covariances directly from the training data
- Other models, e.g. SVMs, require cross-validation
- **GP**: minimize negative log marginal likelihood $\mathcal{L}(\boldsymbol{\theta})$ wrt covariance function parameters and noise level $\boldsymbol{\theta}$. Since

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{0},\, \mathbf{K} + \sigma^2 \mathrm{I}_m),$$

then

$$\mathcal{L} = -\log p(\mathbf{y}|\boldsymbol{\theta}) = \underbrace{\frac{1}{2}\log\det\mathbf{C}(\boldsymbol{\theta})}_{regularization} + \underbrace{\frac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{C}^{-1}(\boldsymbol{\theta})\mathbf{y}}_{data-fit} + \frac{m}{2}\log(2\pi),$$

where $\mathbf{C} = \mathbf{K} + \sigma^2 \mathrm{I}_m$

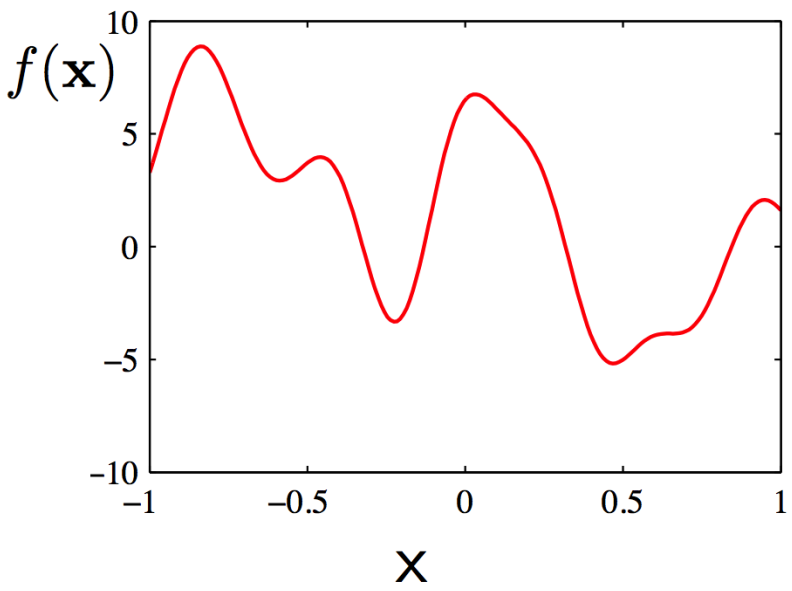
- Uncertainty in function variables $\mathbf{f}$ is taken into account

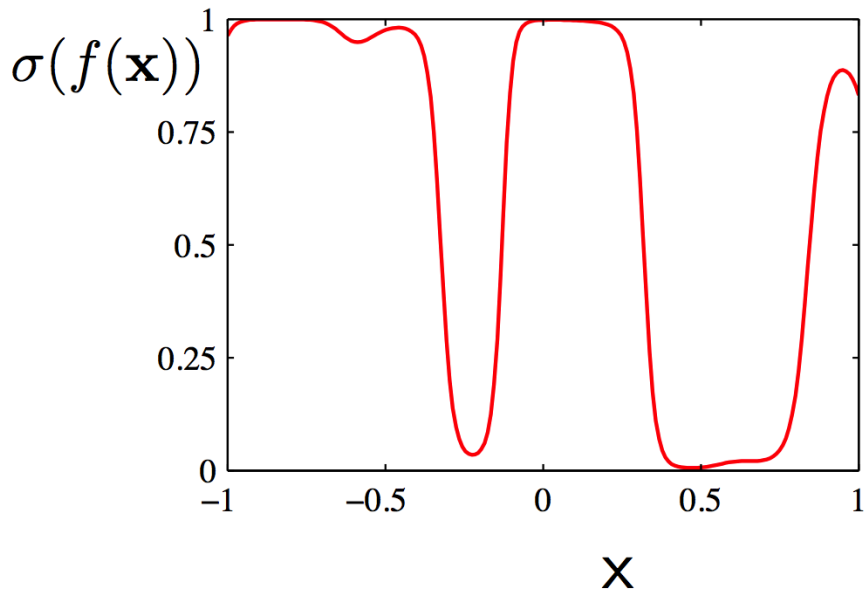
- Minimization of $\mathcal{L}(\boldsymbol{\theta})$ is a non-convex optimization problem
- Standard gradient based techniques, such as CG or quasi-Newton
- Gradients:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{1}{2}\mathrm{tr}\left(\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\right) - \frac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\mathbf{C}^{-1}\mathbf{y}$$

- Local minima, but usually not much of a problem with few parameters, and/or using restarts
- Use weighted sums of covariances and let ML choose
- Each iteration of optimization costs $O(m^3)$

- Binary classification task: $y = \pm 1$
- GLM likelihood: $p(y = +1|\mathbf{x}, \mathbf{w}) = \pi(\mathbf{x}) = \sigma(\mathbf{x}^{\mathrm{T}}\mathbf{w})$
- $\sigma(z)$ — sigmoid function such as the logistic or cumulative normal
- Weight space viewpoint: prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_w)$
- Function space viewpoint: let $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x})$, then likelihood $\pi(\mathbf{x}) = \sigma(f(\mathbf{x}))$, Gaussian prior on $\mathbf{f}$





- Place a GP prior directly on $f(\mathbf{x})$
- Use a sigmoidal likelihood: $p(y = +1|f) = \sigma(f)$
- Non-Gaussian likelihood makes integrating over $\mathbf{f}$ intractable:

$$p(f_*|\mathbf{y}) = \int d\mathbf{f} p(f_*|\mathbf{f}) p(\mathbf{f}|\mathbf{y}),$$

  where the posterior $p(\mathbf{f}|\mathbf{y}) \sim p(\mathbf{y}|\mathbf{f})p(\mathbf{f})$
- Make tractable by using a Gaussian approximation to posterior. Then prediction:

$$p(y_* = +1|\mathbf{y}) = \int df_* \sigma(f_*) p(f_*|\mathbf{y})$$

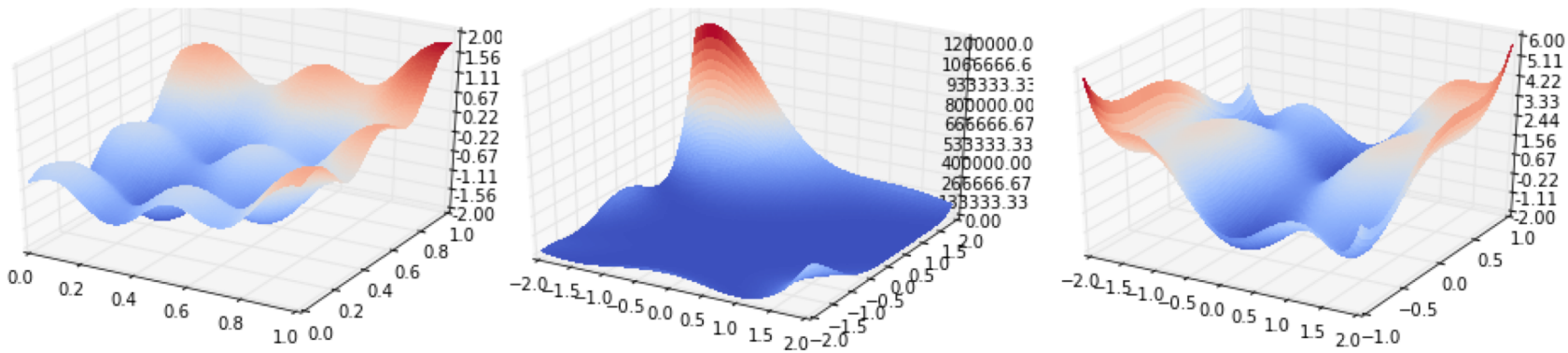
Two common ways to make Gaussian approximation to posterior:

1. Laplace approximation. Second order Taylor approximation about mode of posterior
2. Expectation propagation (EP). EP can be thought of as approximately minimizing $\mathrm{KL}\left[p(\mathbf{f}|\mathbf{y})||q(\mathbf{f}|\mathbf{y})\right]$ by an iterative procedure

- Kuss and Rasmussen [2005] evaluate both methods experimentally and find EP to be often better
- Classification accuracy on digits data sets comparable to SVMs
- Advantages: uncertainty quantification, probabilistic predictions, kernel parameters by ML

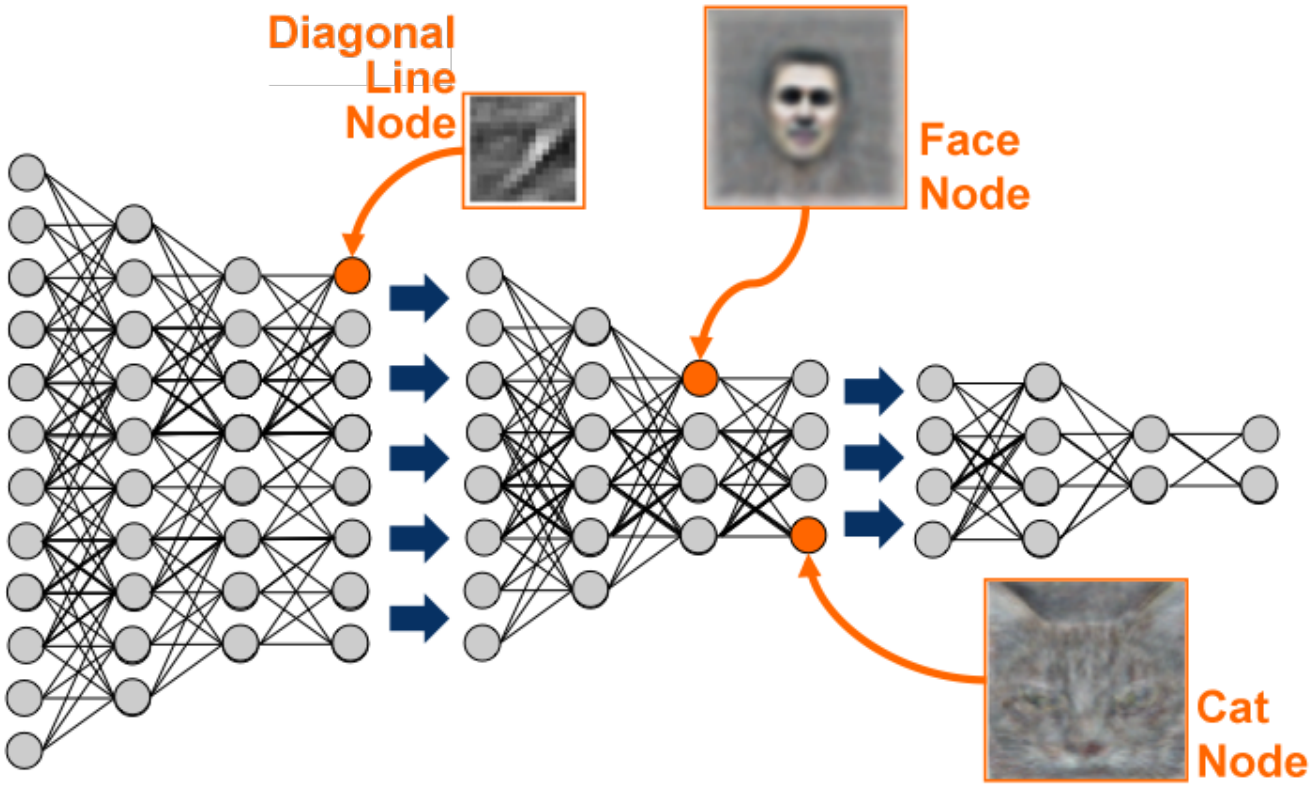
Consider a "well behaved" function $f : \mathcal{X} \to \mathbb{R}$, with $\mathcal{X} \subseteq \mathbb{R}^d$ being a compact set

$$\mathbf{x}_{\min} = \arg\min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$$



- f is explicitly unknown and multimodal
- Evaluations of f may be perturbed
- Evaluations of f are expensive $\Rightarrow$
  - Gradient and Hessian are not computable
  - Grid search is not possible

Parameter tuning in ML algorithms



- Number of layers/units per layer
- Types of each layer
- Regularization coefficients
- Learning rates, etc.

Other expensive functions

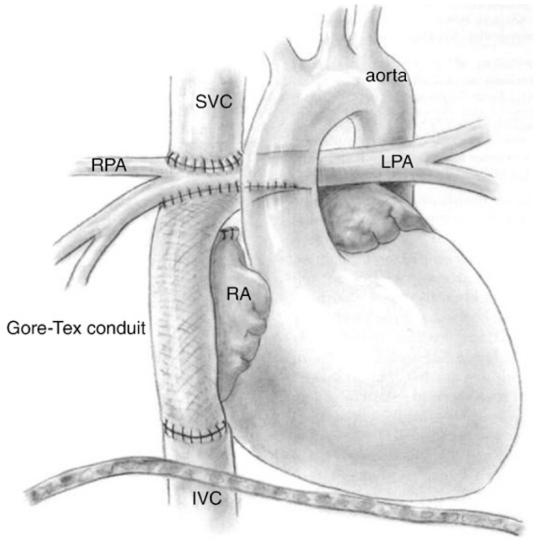
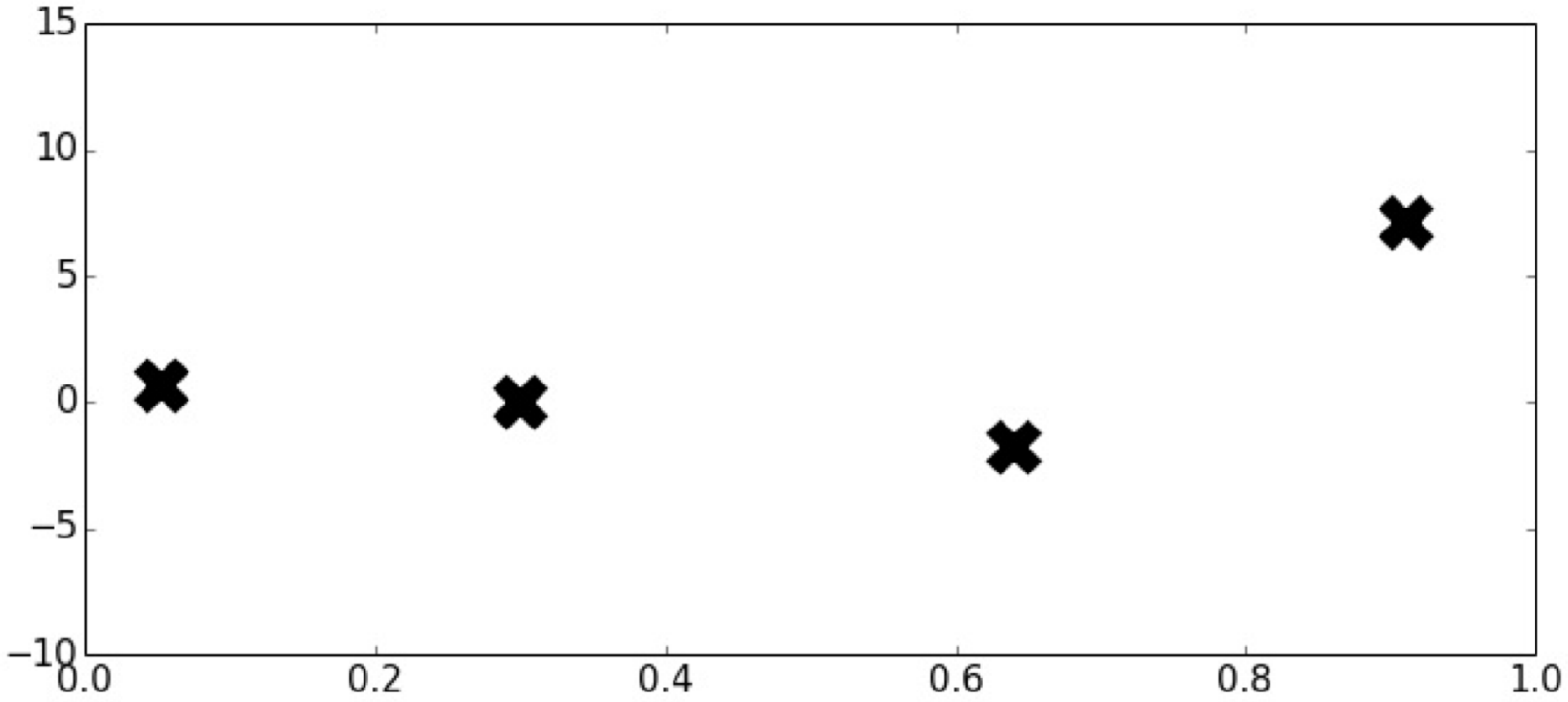


Fig. 1. Extracardiac total cavopulmonary connection. The IVC is disconnected from the right atrium (RA) and connected to the PAs via a Gore-Tex conduit. Figure taken from Reddy et al. [13].

- Design of grafts to be used in heart surgery
- Design of aeorodynamic structures, e.g., cars, airplanes
- Calibrating parameters of complex physical models to experimental data
- A/B testing data to optimize the web design to maximize sign-ups, downloads, purchases, etc.

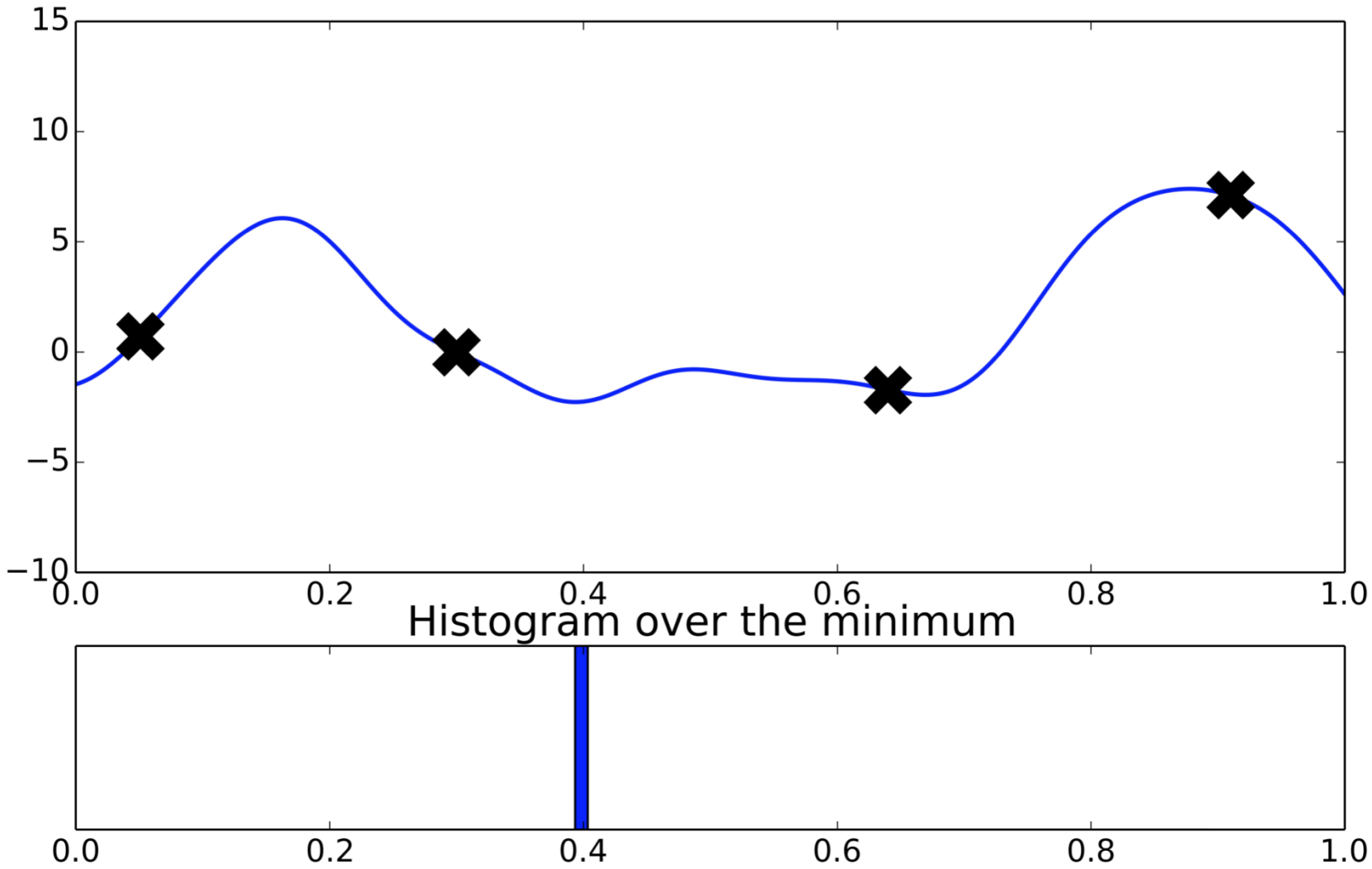
NB! There exists commercial services for optimizing black-box functions: SIGOPT, Google Vizier, etc.

Typical situation

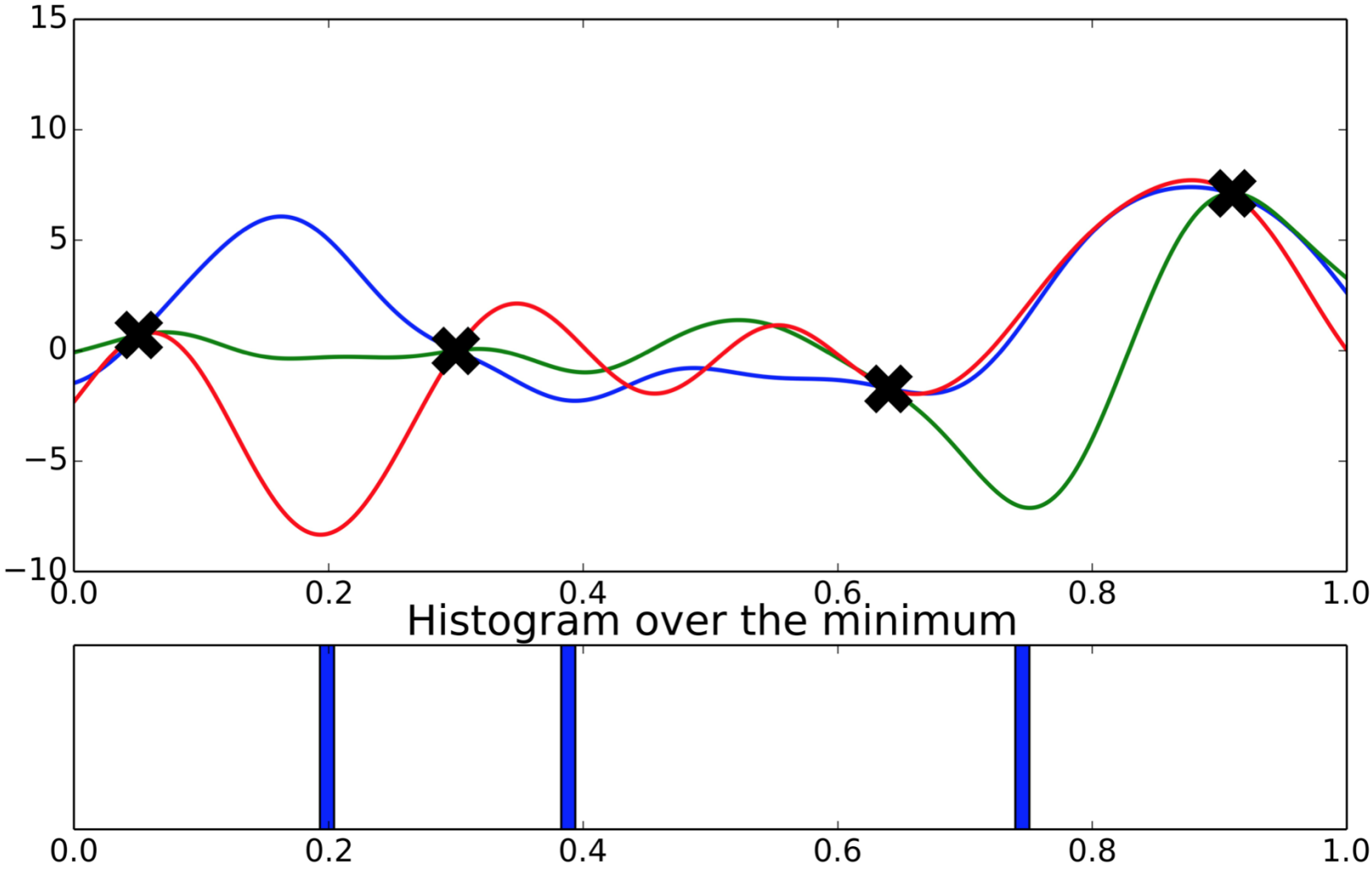


Where is the minimum of f?
Where should we evaluate the function next?

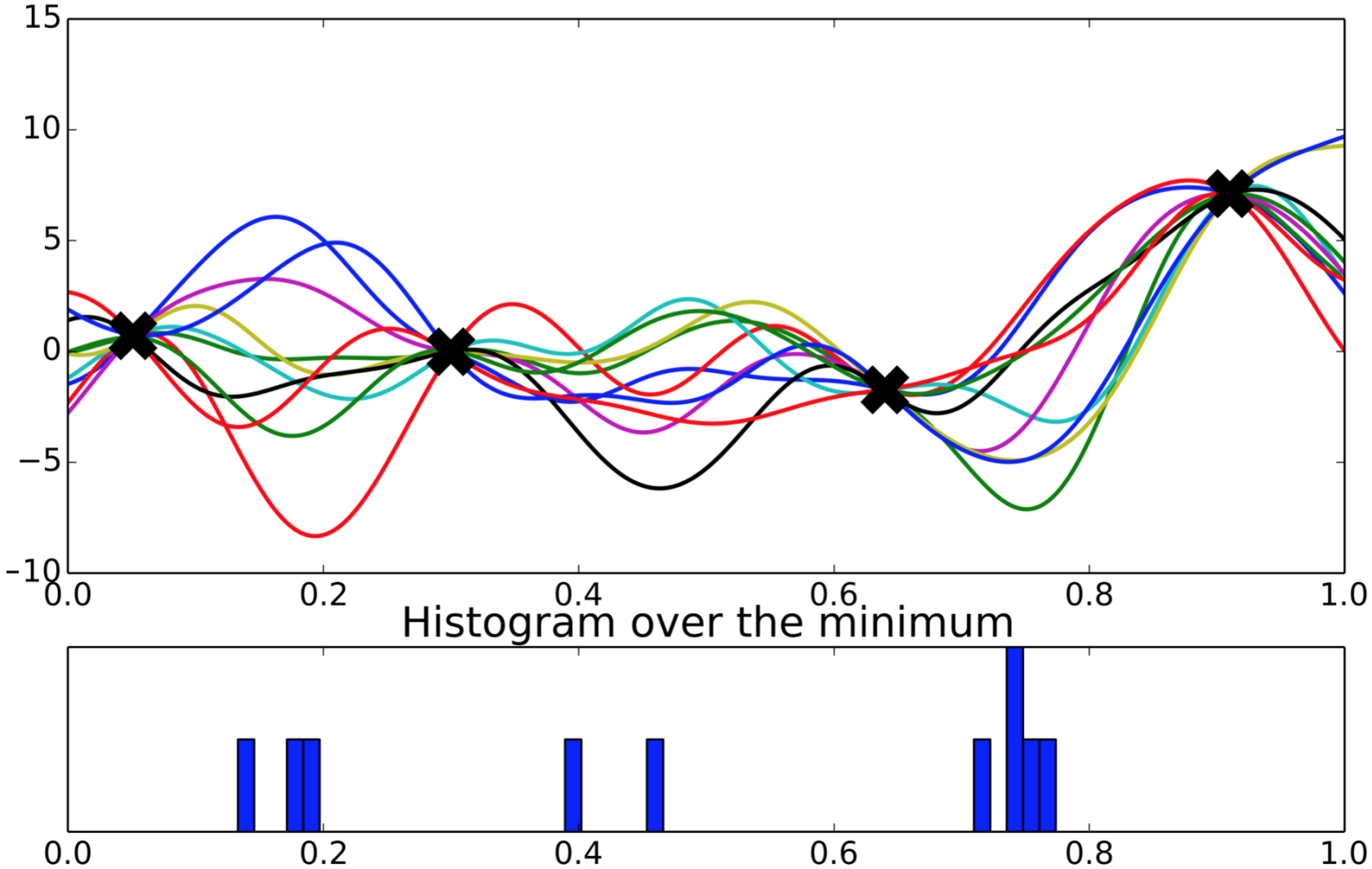
Intuition: one curve



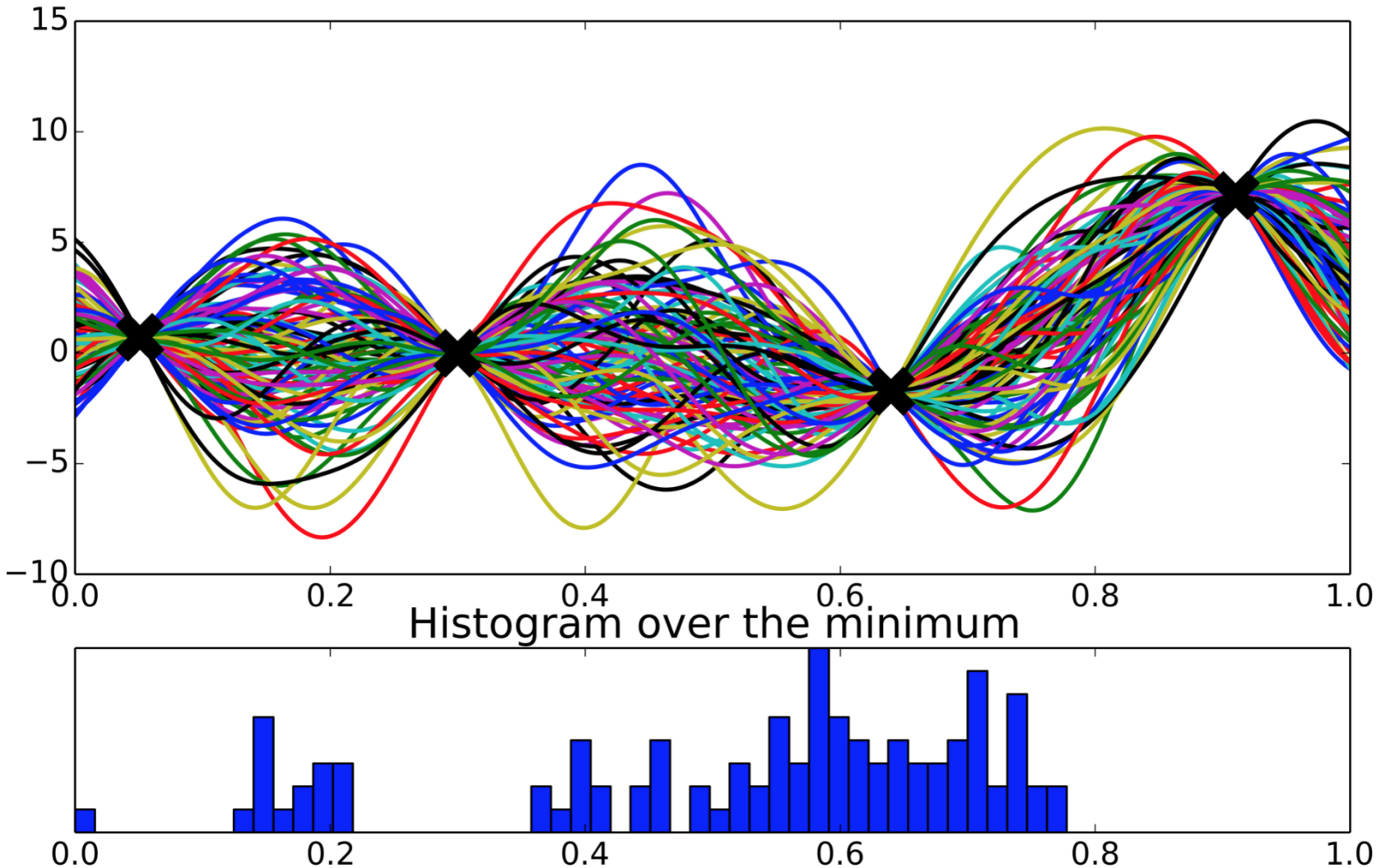
Intuition: three curves



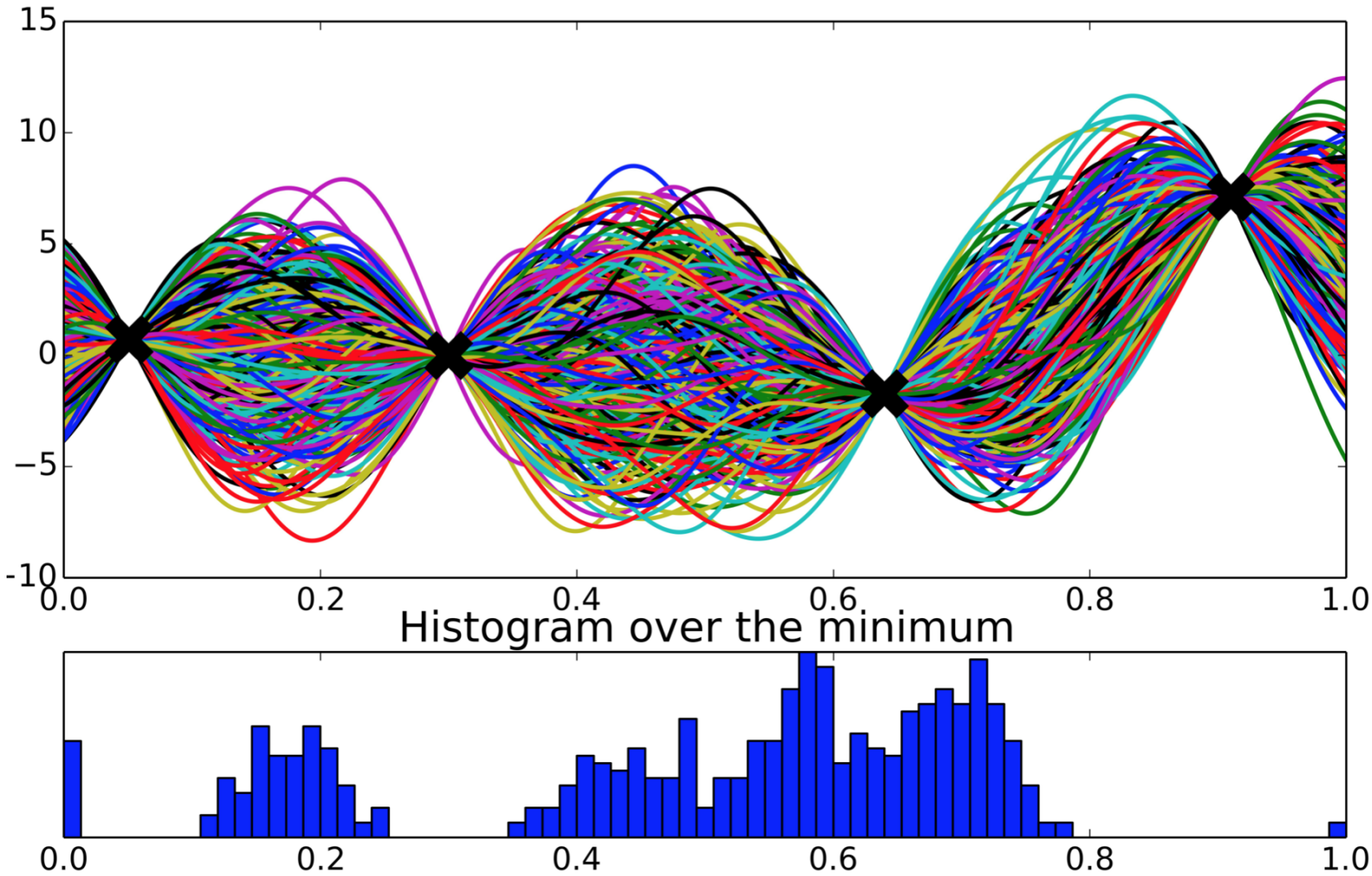
Intuition: ten curves



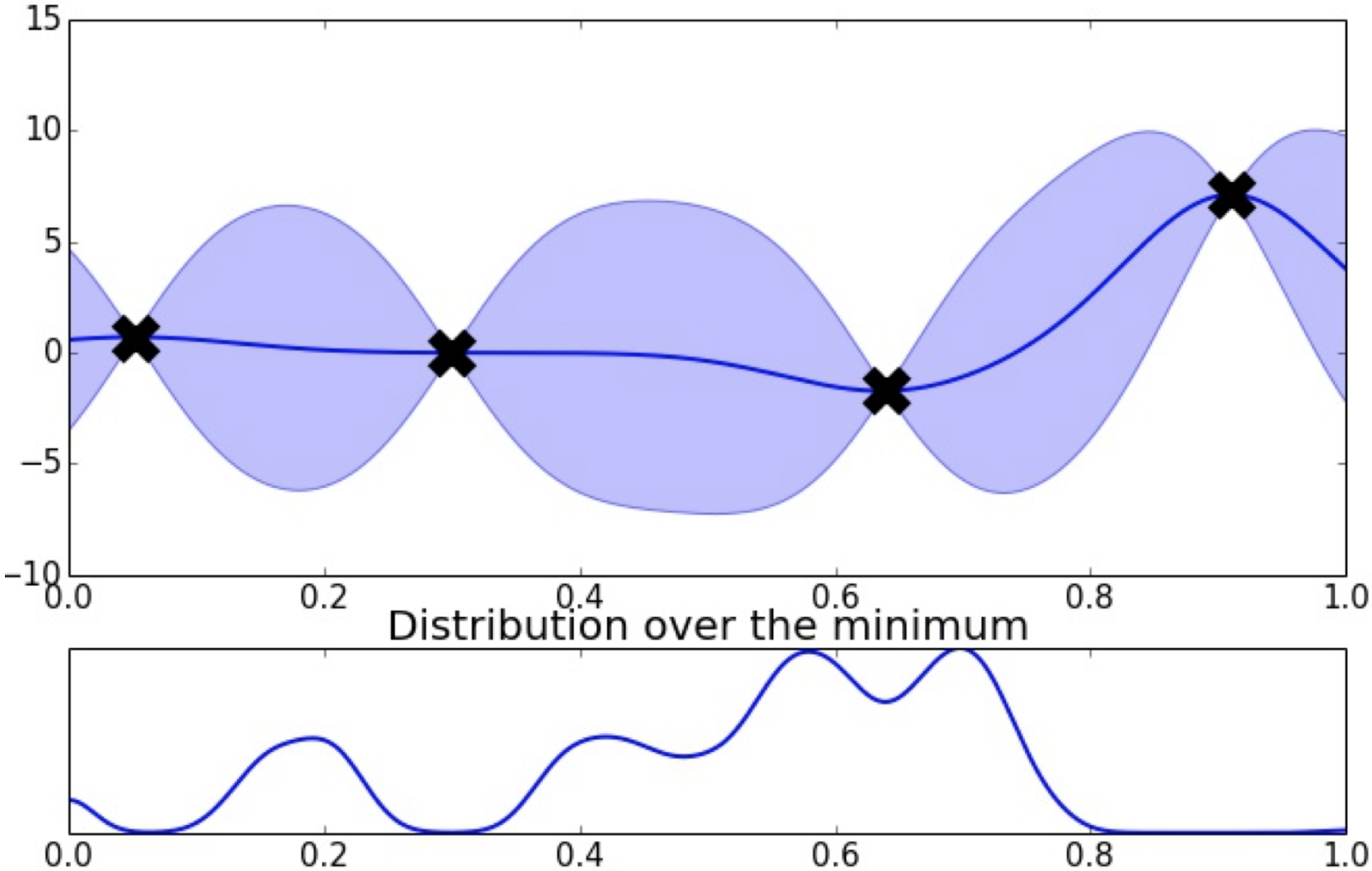
Intuition: hundred curves



Intuition: many curves



Intuition: infinite number of curves



Formalization

- We made prior assumption about f
- Information about the minimum is now encoded in a new function (the probability distribution $p_{\min}$ over the minimum in this case)
- We can use $p_{\min}$ (or a functional of it) 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.

Methodology to perform global optimization of multimodal black-box functions

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 set

Iterate between 2 and 4 until the evaluation budget is over

Comment: BO can be theoretically formalized in the framework of dynamic programming principle

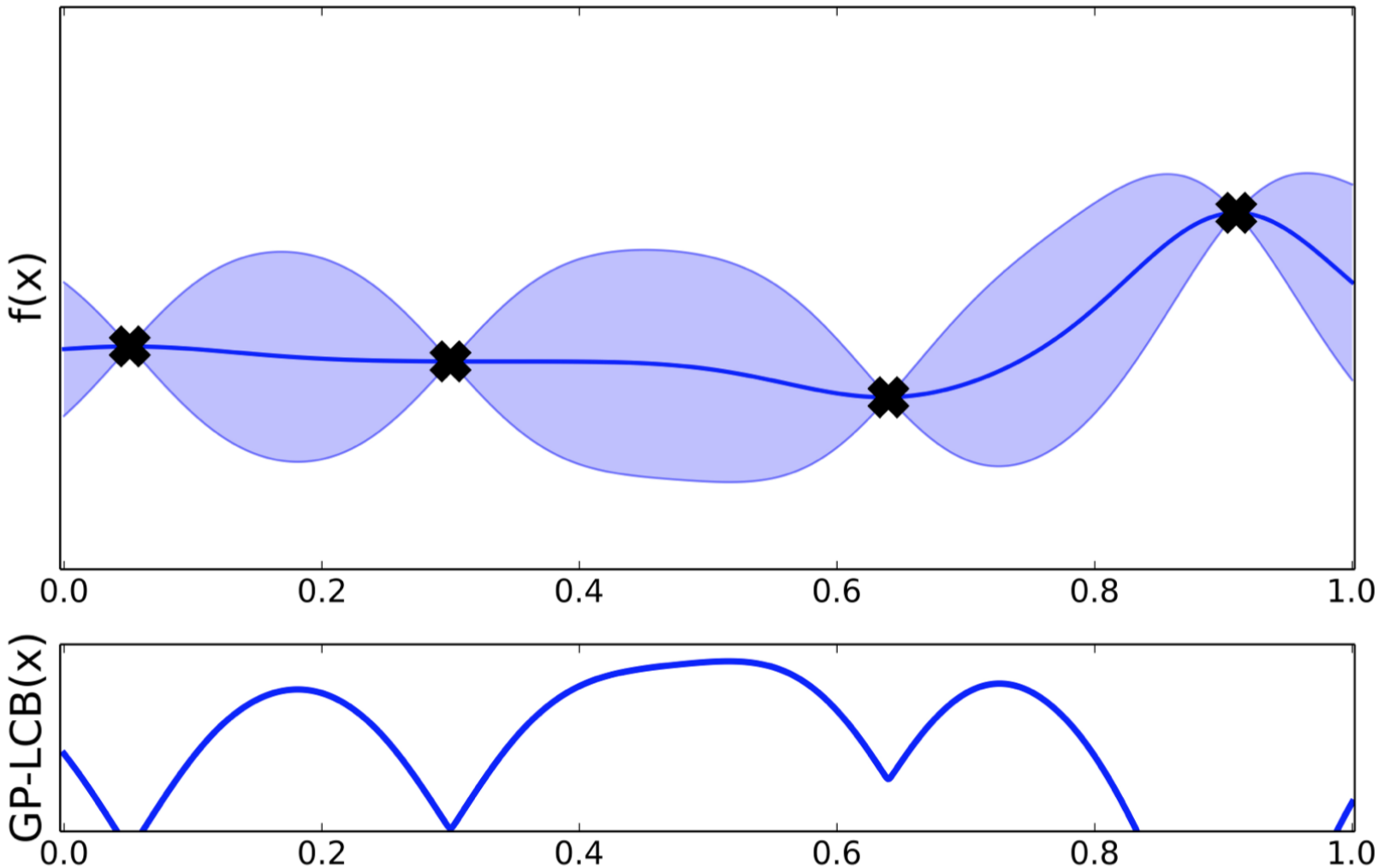
- Use GP $\mathcal{GP}(\cdot|\mu(\mathbf{x}), K(\mathbf{x}, \mathbf{x}'))$ as a prior for $f(\cdot)$
- GP has marginal closed-form for the posterior mean $\mu_*(\mathbf{x})$ and variance $\sigma_*^2(\mathbf{x}) \Rightarrow$ efficient calculation of acquisition function
  - **Exploration**: Evaluate in 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

GP Upper (lower) Confidence Band

Direct balance between exploration and exploitation:

$$\alpha_{LCB}(\mathbf{x}) = -\mu_*(\mathbf{x}) + \zeta\sigma_*(\mathbf{x})$$

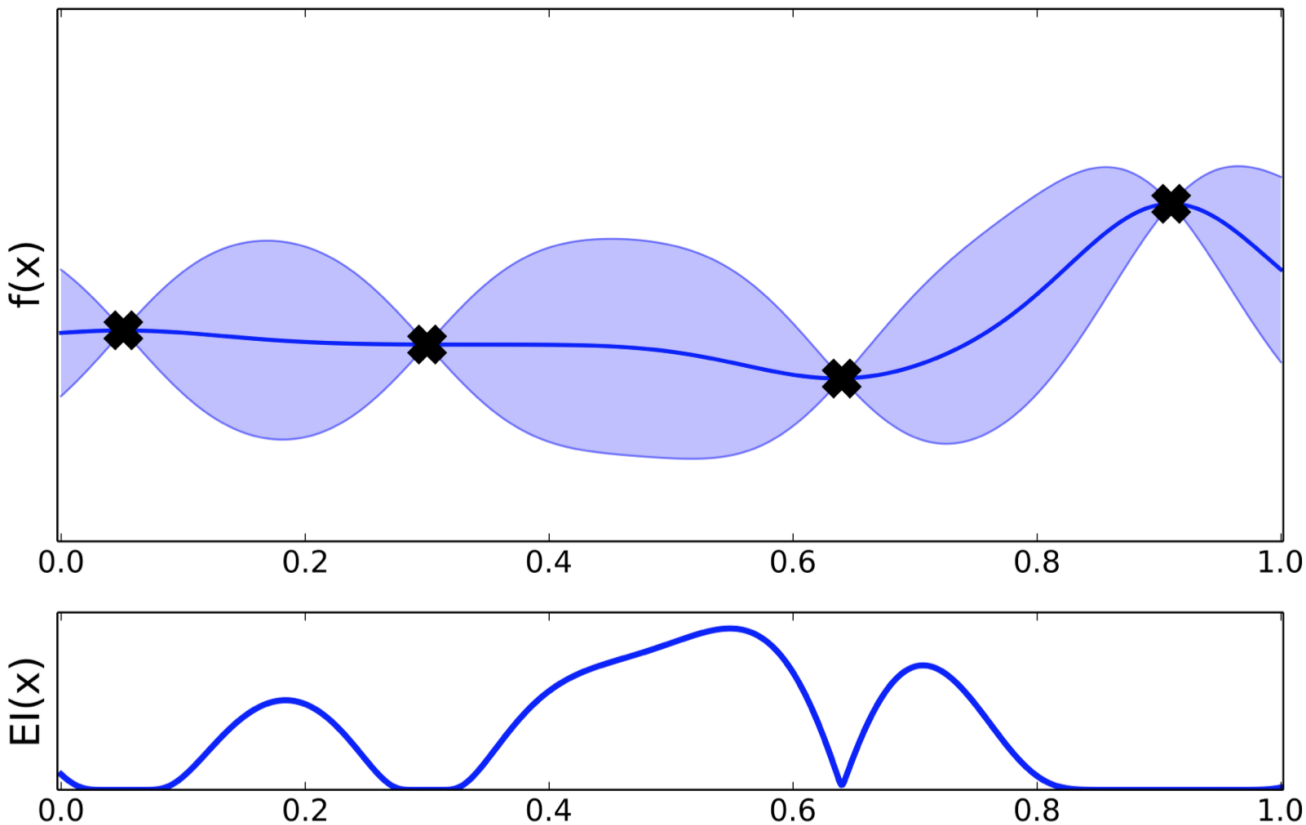


Expected Improvement

Let us denote by $\Delta(\mathbf{x}) = y_{\text{best}} - \mu_*(\mathbf{x})$, then

$$\alpha_{EI}(\mathbf{x}) = \int_y \max(0, y_{\text{best}} - y_*) p(y_*|\mathbf{x}) dy_* =$$

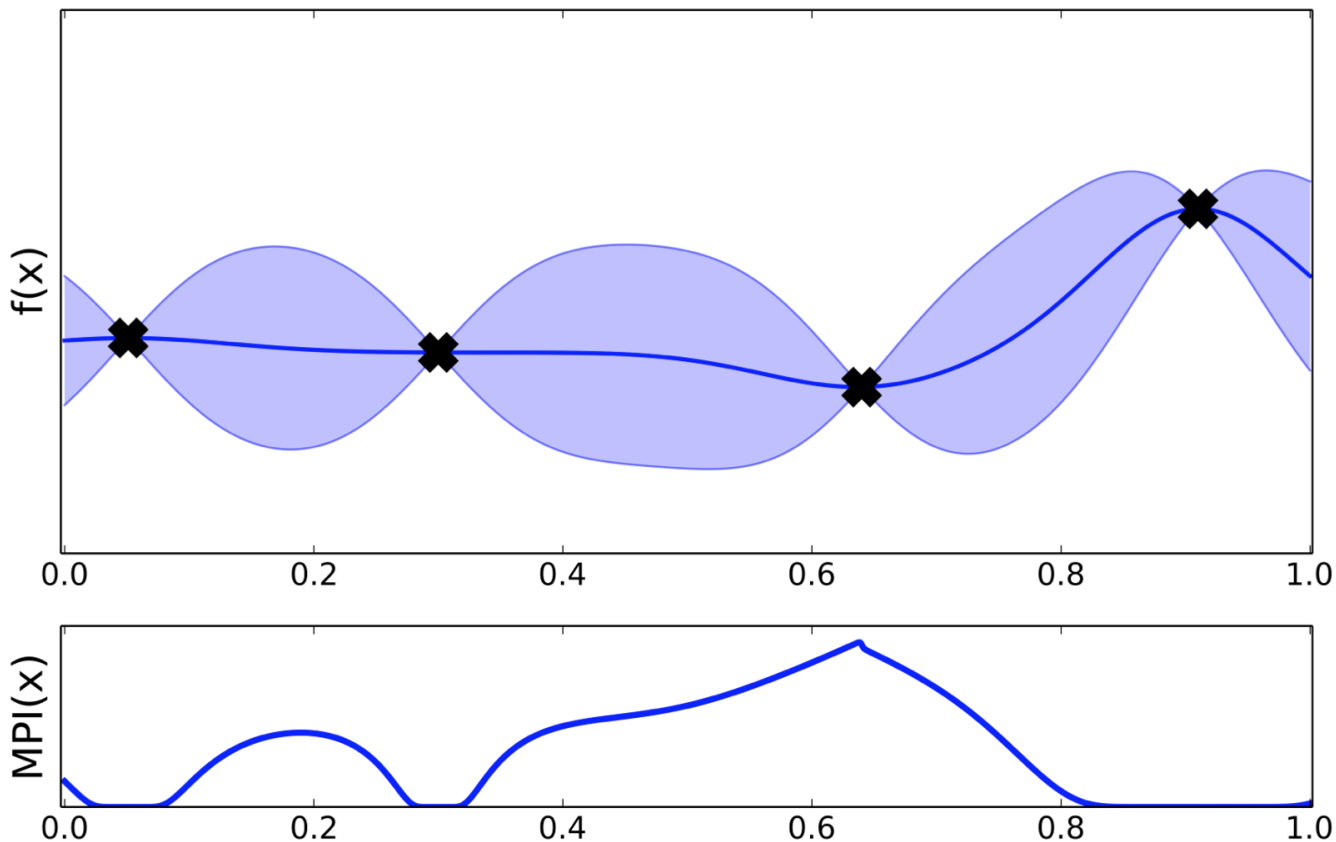
$$= \max(0, \Delta(\mathbf{x})) - \sigma_*(\mathbf{x}) \varphi\left(\frac{\Delta(\mathbf{x})}{\sigma_*(\mathbf{x})}\right) + |\Delta(\mathbf{x})| \Phi\left(-\frac{|\Delta(\mathbf{x})|}{\sigma_*(\mathbf{x})}\right)$$



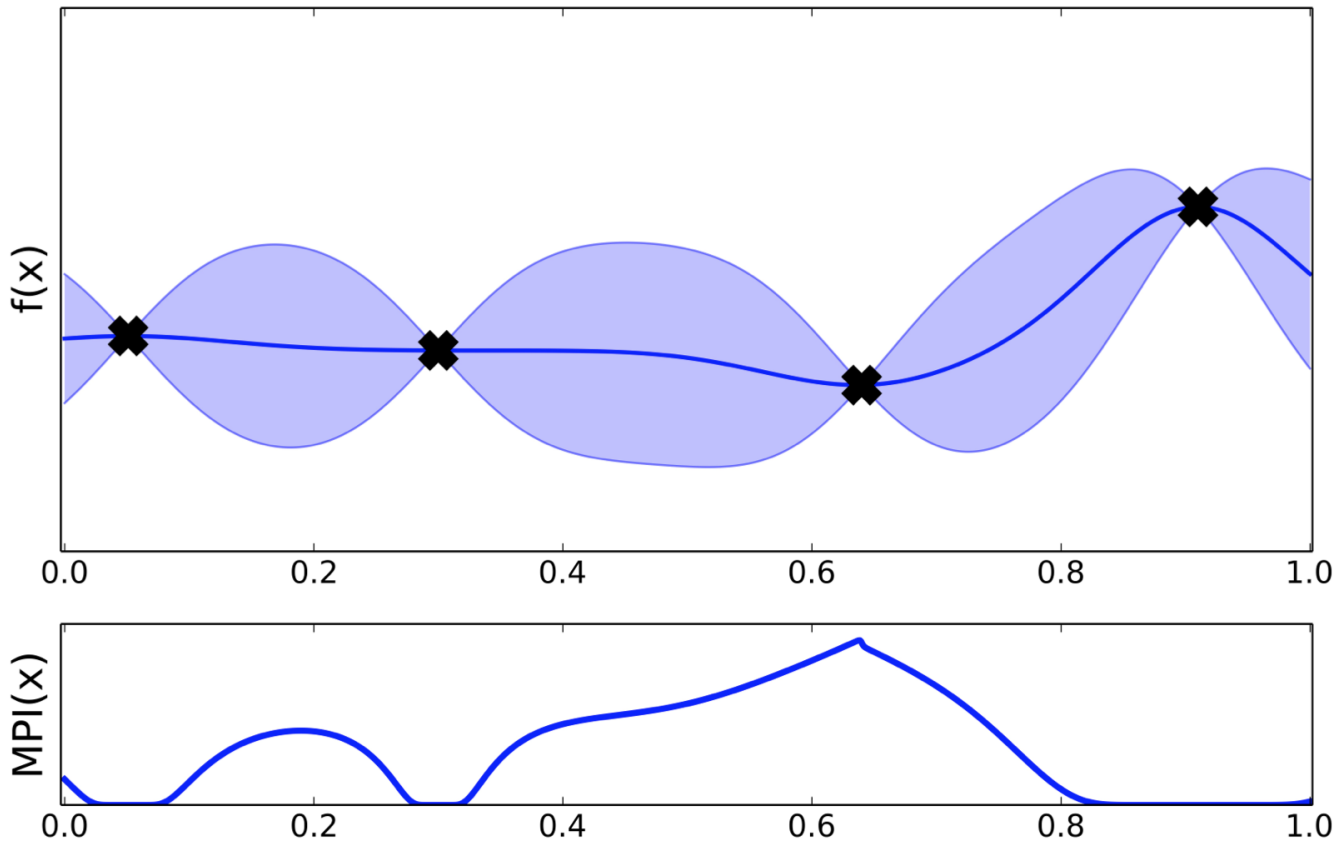
Maximum Probability of Improvement

$$\gamma(\mathbf{x}) = \frac{\mu(\mathbf{x}) - y_{\text{best}}}{\sigma(\mathbf{x})}$$

$$\alpha_{MPI}(\mathbf{x}) = \mathbb{P}(f(\mathbf{x}) < y_{\text{best}}) = \Phi(\gamma(\mathbf{x}))$$



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



- BO is an strategy to transform the problem

$$\mathbf{x}_{\min} = \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$$

unsolvable!

into a series of problems

$$\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha(\mathbf{x}|S_t),$$

solvable!

where

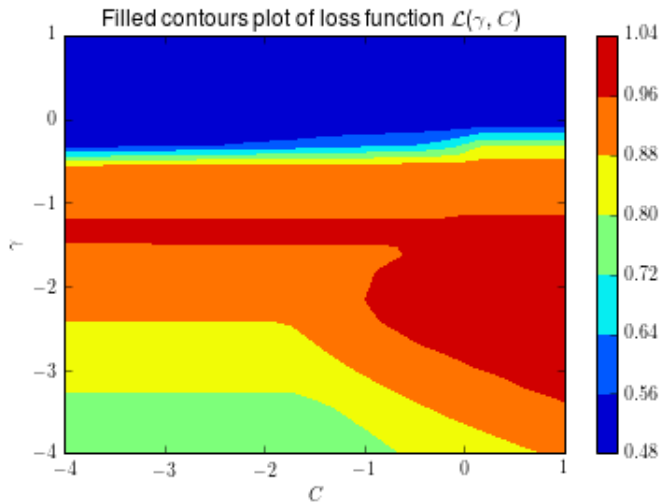
  - $\alpha(\mathbf{x})$ is not so expensive to evaluate
  - Gradients of $\alpha(\mathbf{x})$ are typically available
  - Still need to find $\mathbf{x}_{t+1}$: DIRECT, gradient methods, SA

AutoML example: SVM hyperparameters tuning

- Scikit-learn dummy data for testing classifiers:

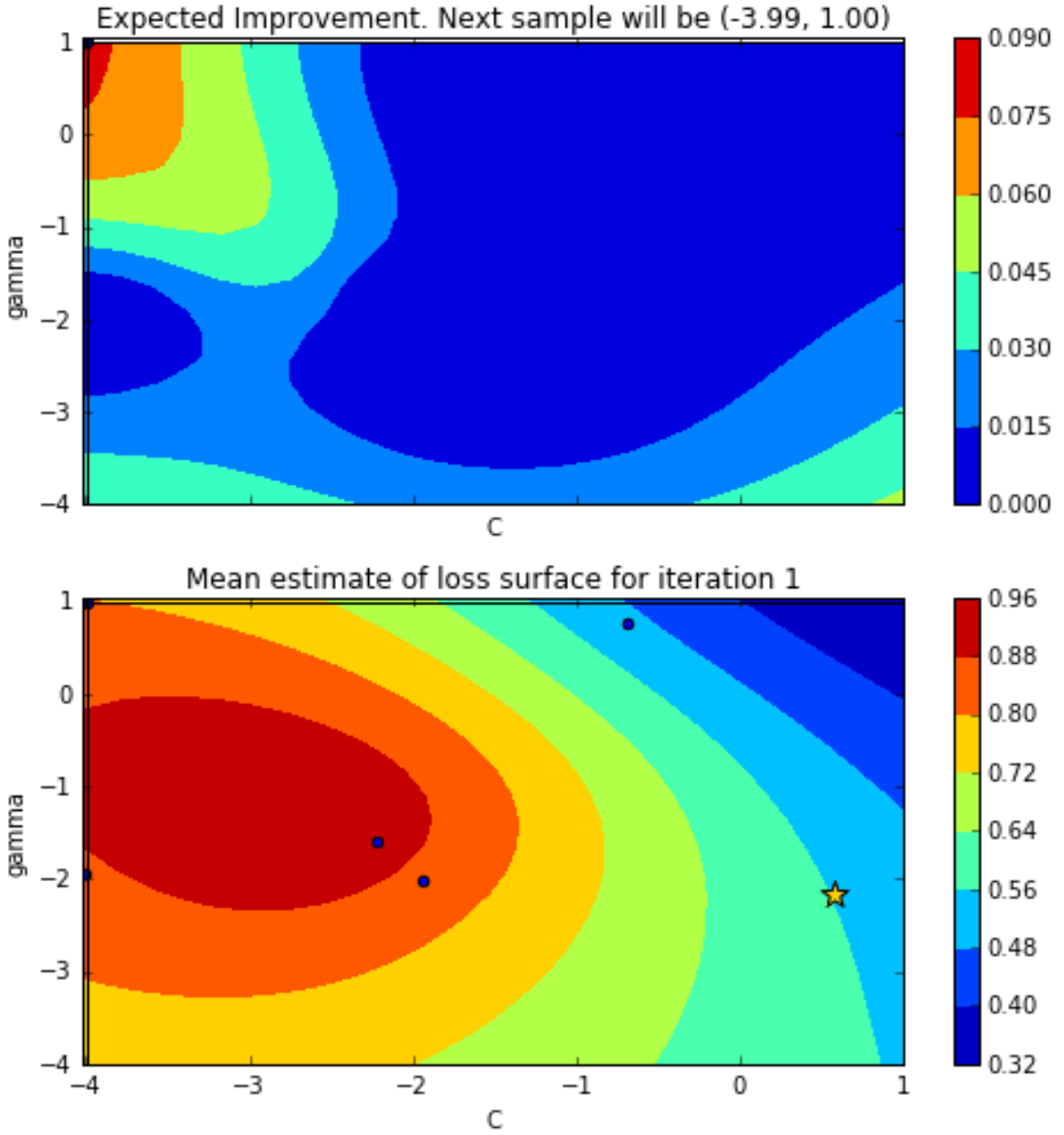
  n_samples=2500, n_features=45, n_informative=15, n_redundant=5

- Optimize penalization parameter C, and kernel width parameter γ
- The loss function is the cross-validated area under the curve (AUC), based on three folds
- We computed the loss surface as a function of C and γ to get an accurate estimate of where the true optimum of the loss surface is

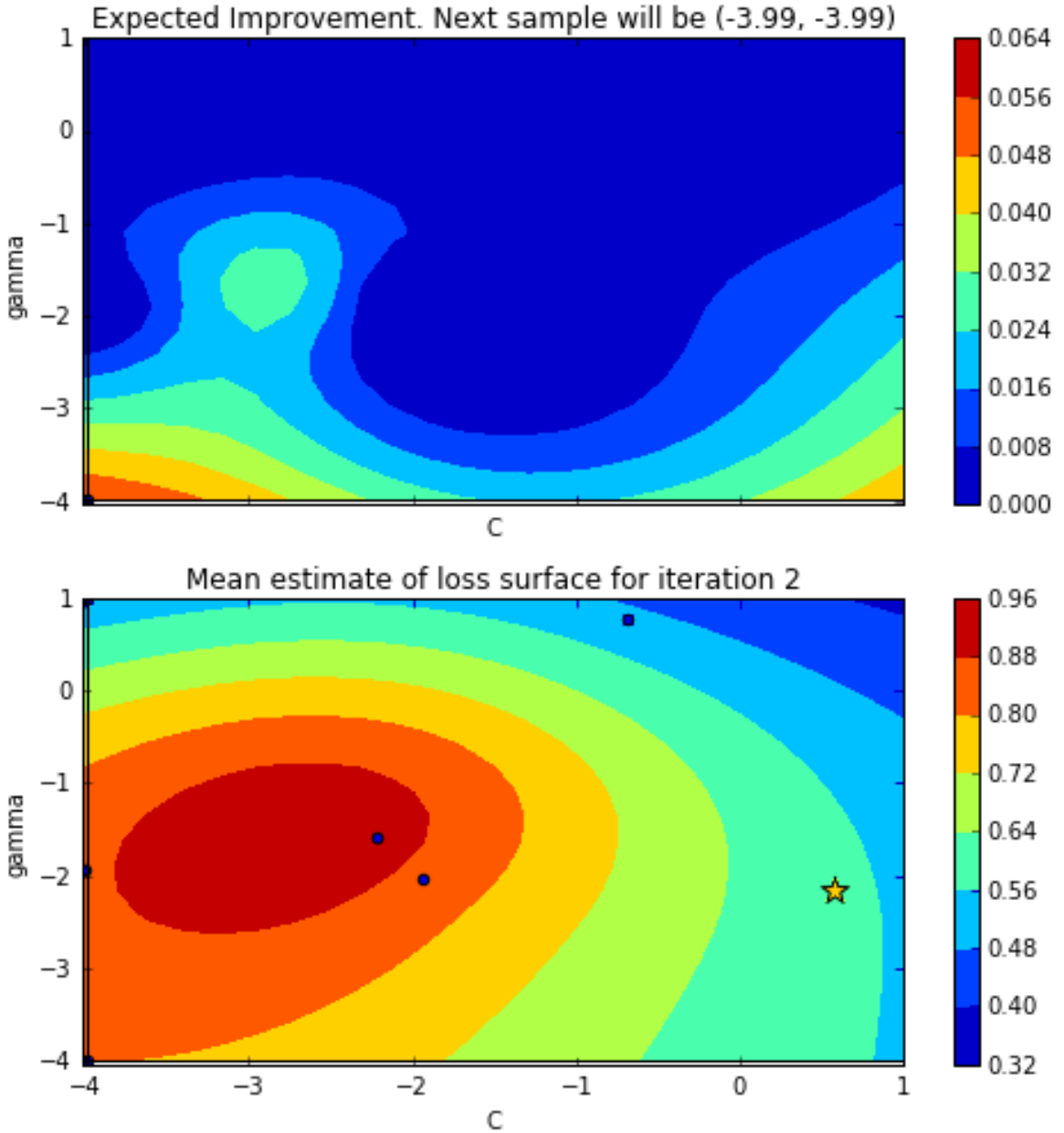


- The figures below show the sequence of points selected, if we run the Bayesian optimization algorithm in this setting

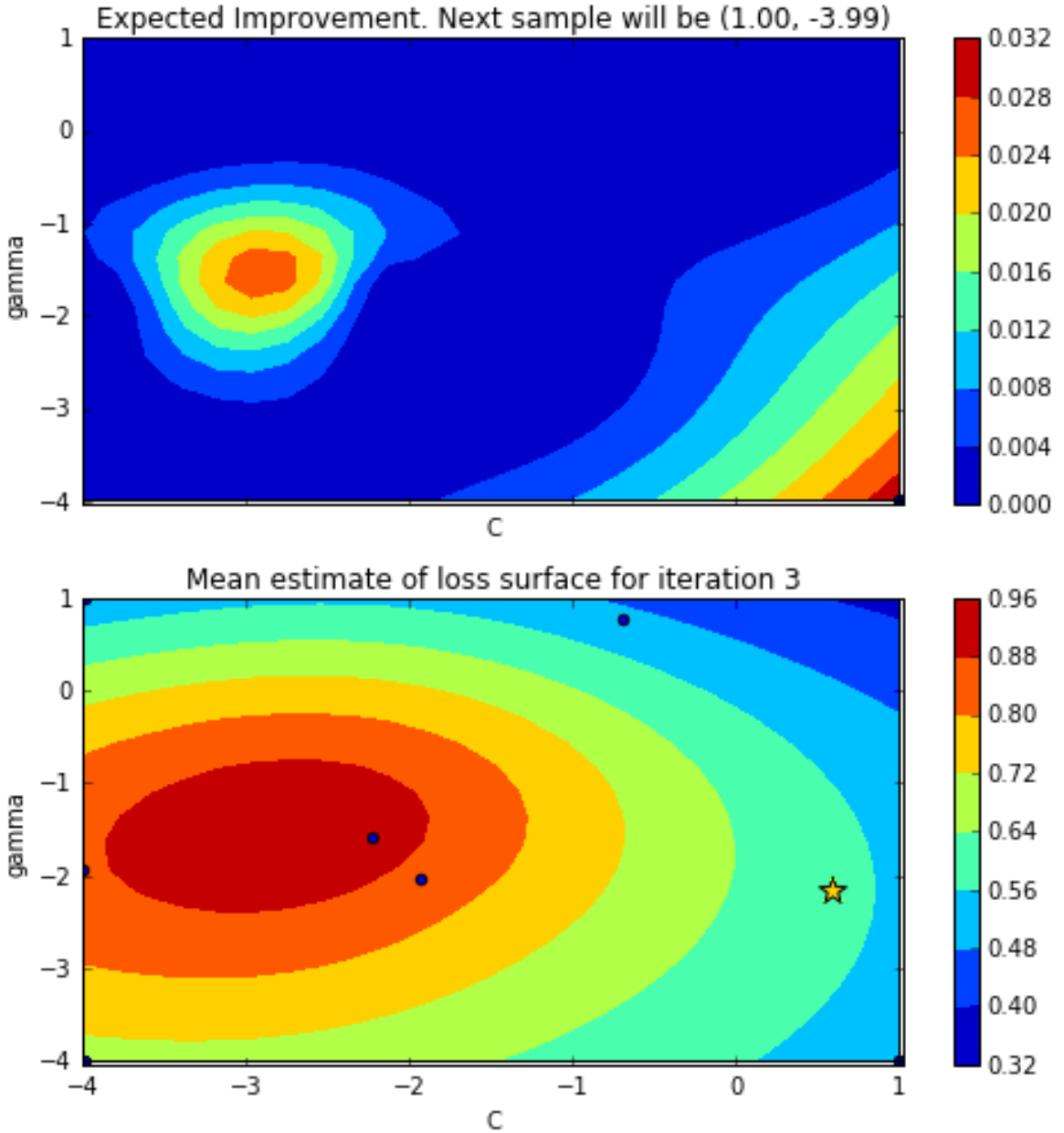
Iterations


Expected Improvement. Next sample will be (-3.99, 1.00)
gamma
C
1
0
−1
−2
−3
−4
0.090
0.075
0.060
0.045
0.030
0.015
0.000
Mean estimate of loss surface for iteration 1
gamma
C
−4
−3
−2
−1
0
1
0.96
0.88
0.80
0.72
0.64
0.56
0.48
0.40
0.32

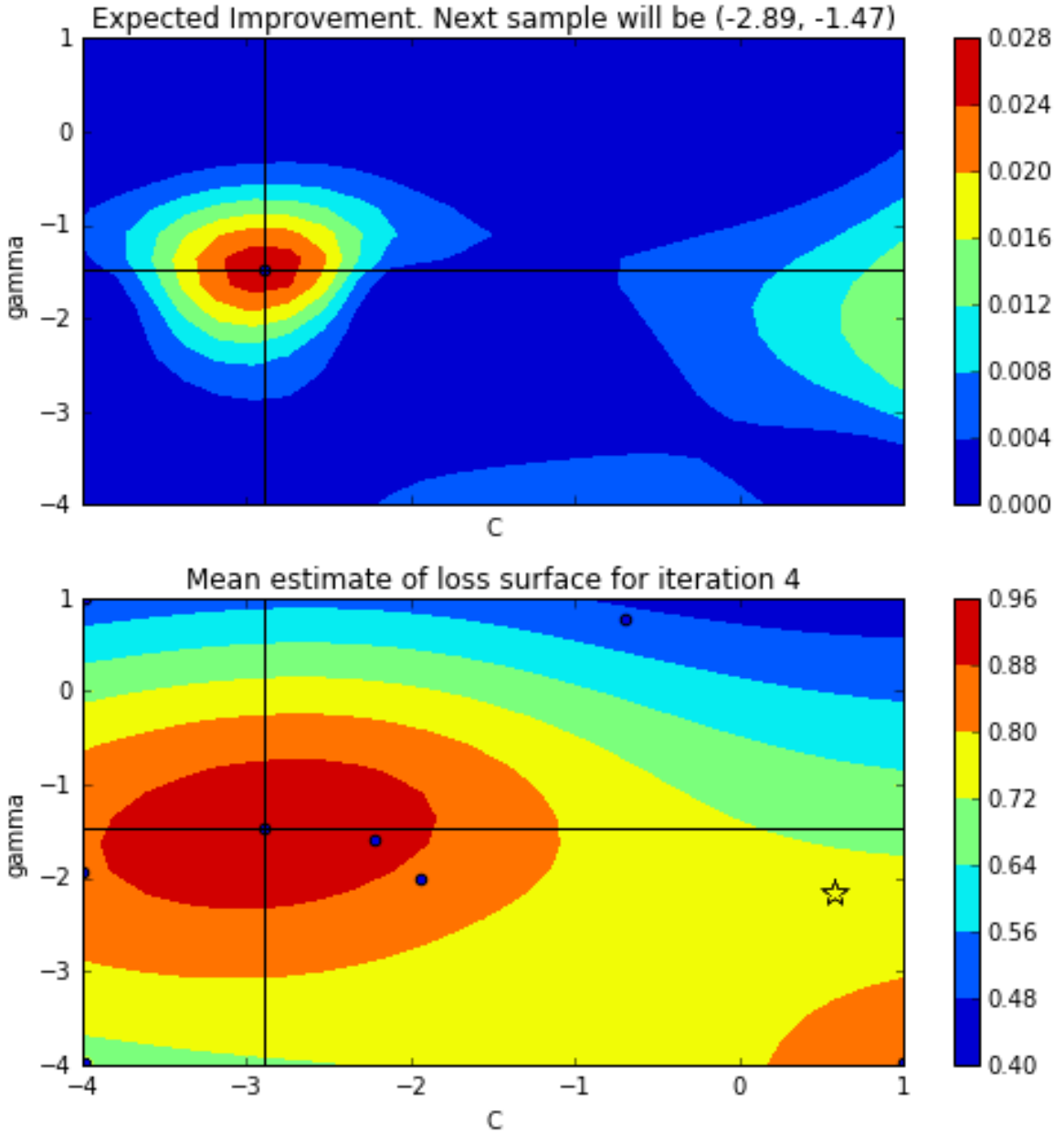
Iterations


Expected Improvement. Next sample will be (-3.99, -3.99)
gamma
C
1
0
−1
−2
−3
−4
0.064
0.056
0.048
0.040
0.032
0.024
0.016
0.008
0.000
Mean estimate of loss surface for iteration 2
gamma
C
−4
−3
−2
−1
0
1
0.96
0.88
0.80
0.72
0.64
0.56
0.48
0.40
0.32

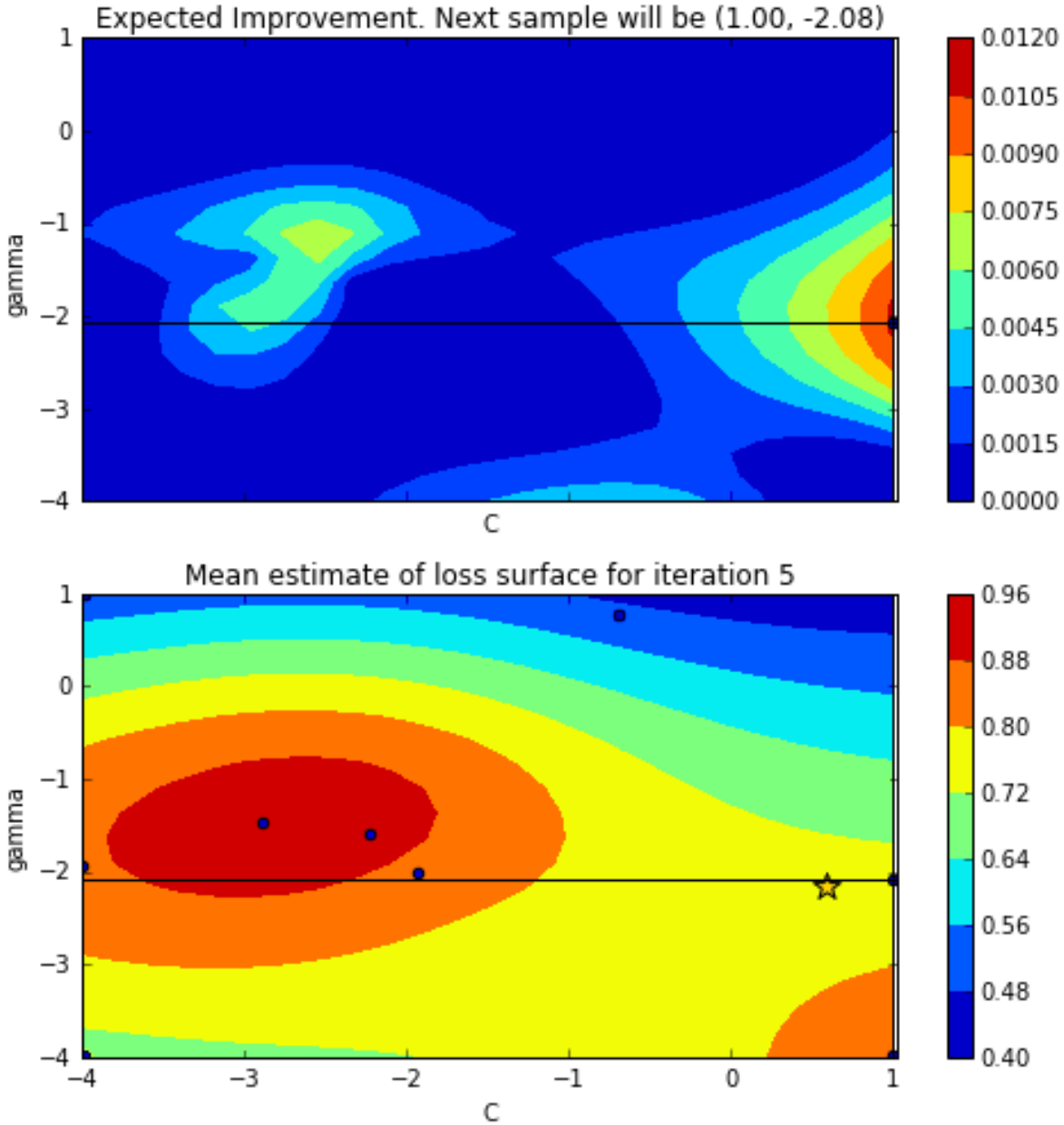
Iterations


Expected Improvement. Next sample will be (1.00, -3.99)
gamma
C
0.032
0.028
0.024
0.020
0.016
0.012
0.008
0.004
0.000
Mean estimate of loss surface for iteration 3
gamma
C
0.96
0.88
0.80
0.72
0.64
0.56
0.48
0.40
0.32

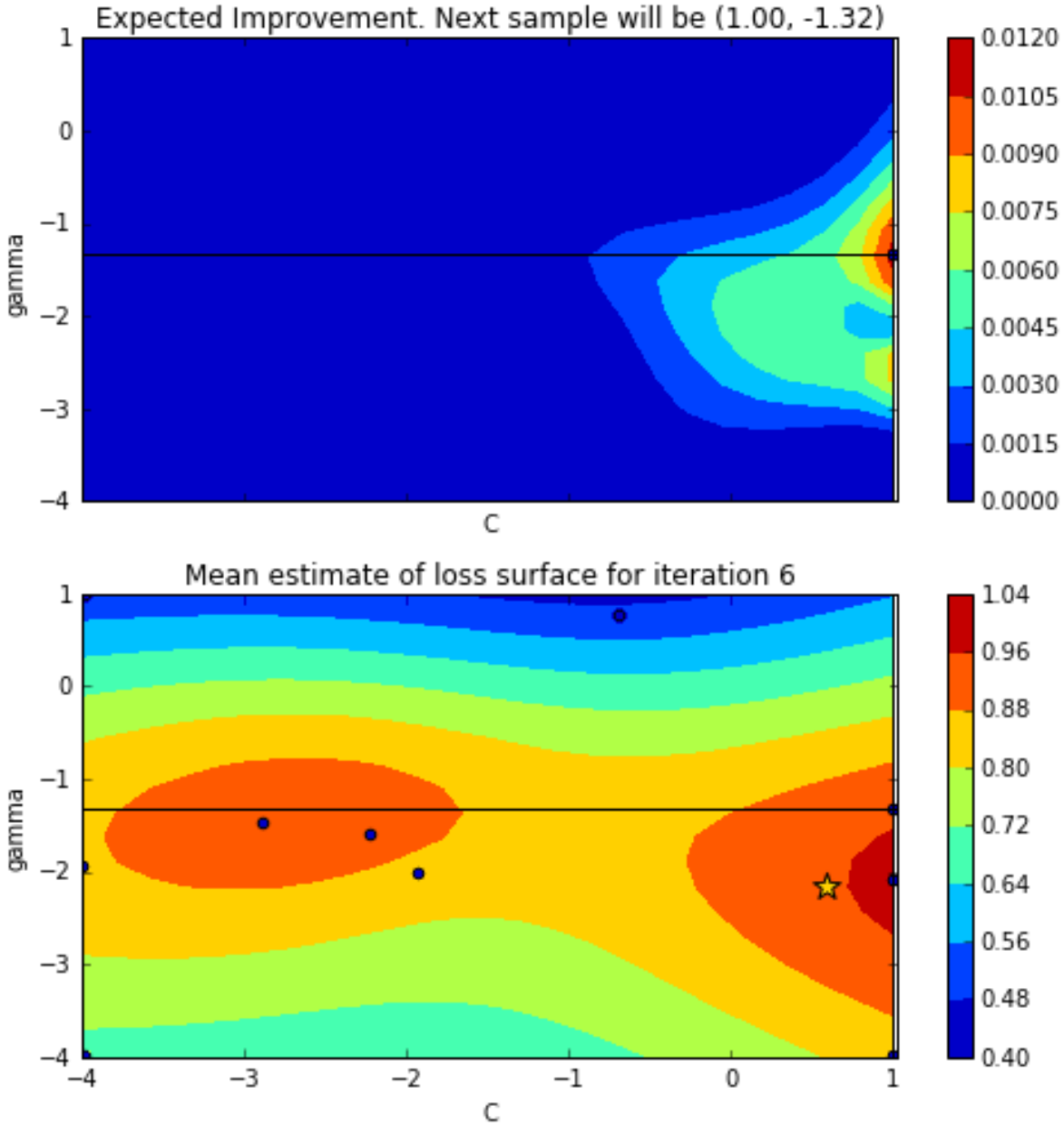
Iterations


Expected Improvement. Next sample will be (-2.89, -1.47)
gamma
C
1
0
−1
−2
−3
−4
0.028
0.024
0.020
0.016
0.012
0.008
0.004
0.000
Mean estimate of loss surface for iteration 4
gamma
C
−4
−3
−2
−1
0
1
0.96
0.88
0.80
0.72
0.64
0.56
0.48
0.40

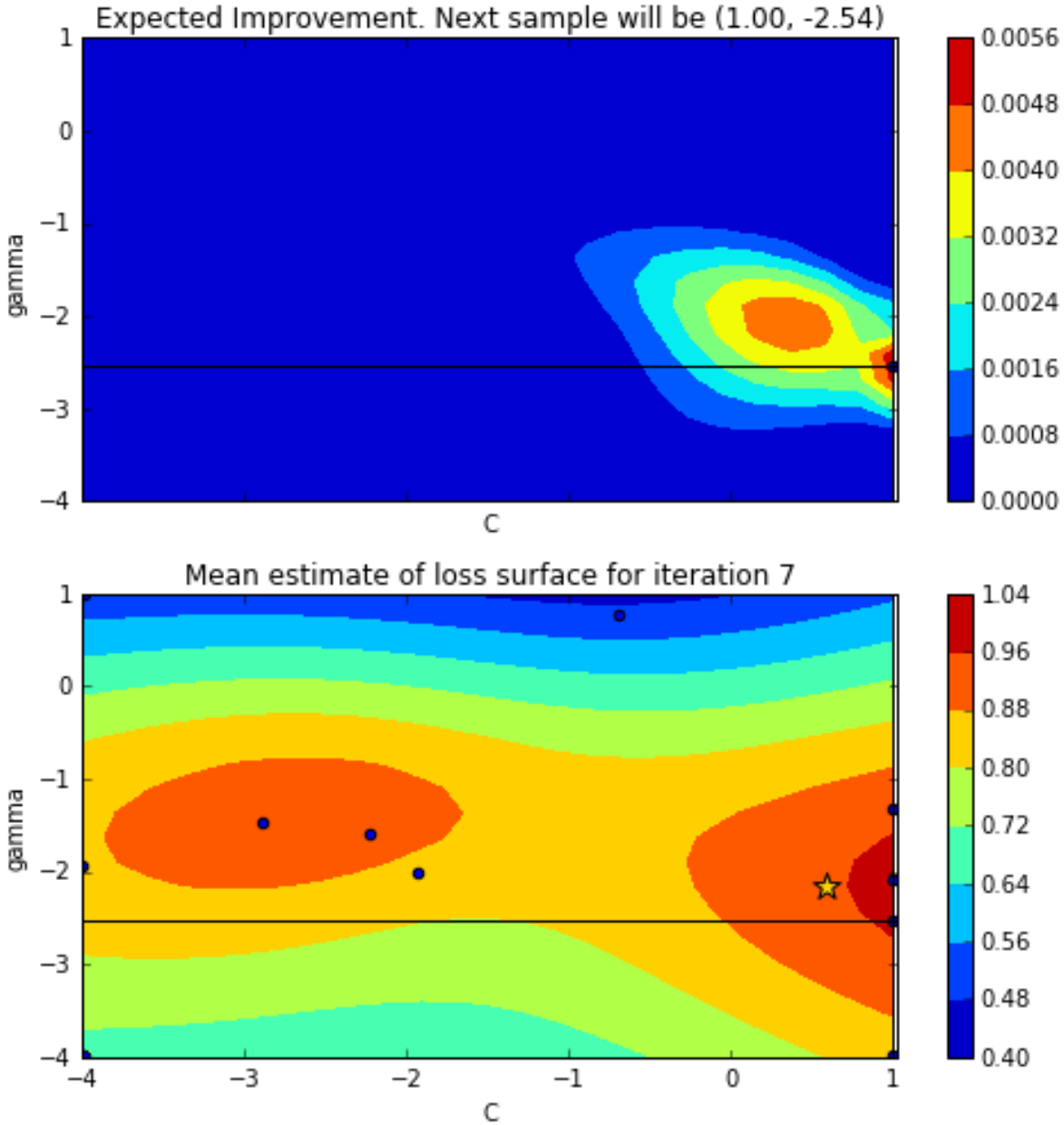
Iterations


Expected Improvement. Next sample will be (1.00, -2.08)
gamma
C
0.0120
0.0105
0.0090
0.0075
0.0060
0.0045
0.0030
0.0015
0.0000
Mean estimate of loss surface for iteration 5
gamma
C
0.96
0.88
0.80
0.72
0.64
0.56
0.48
0.40

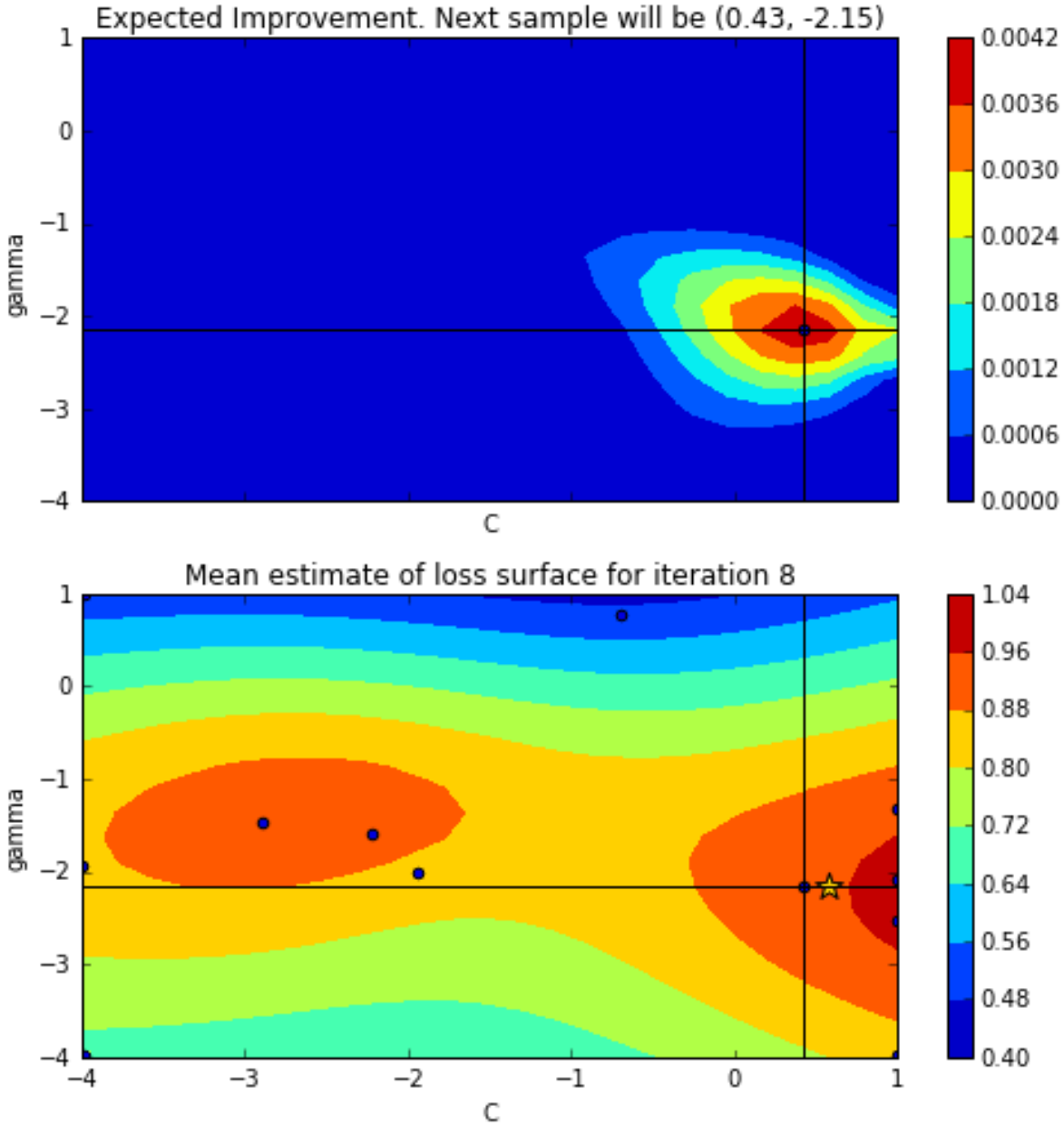
Iterations


Expected Improvement. Next sample will be (1.00, -1.32)
gamma
C
0.0120
0.0105
0.0090
0.0075
0.0060
0.0045
0.0030
0.0015
0.0000
Mean estimate of loss surface for iteration 6
gamma
C
1.04
0.96
0.88
0.80
0.72
0.64
0.56
0.48
0.40

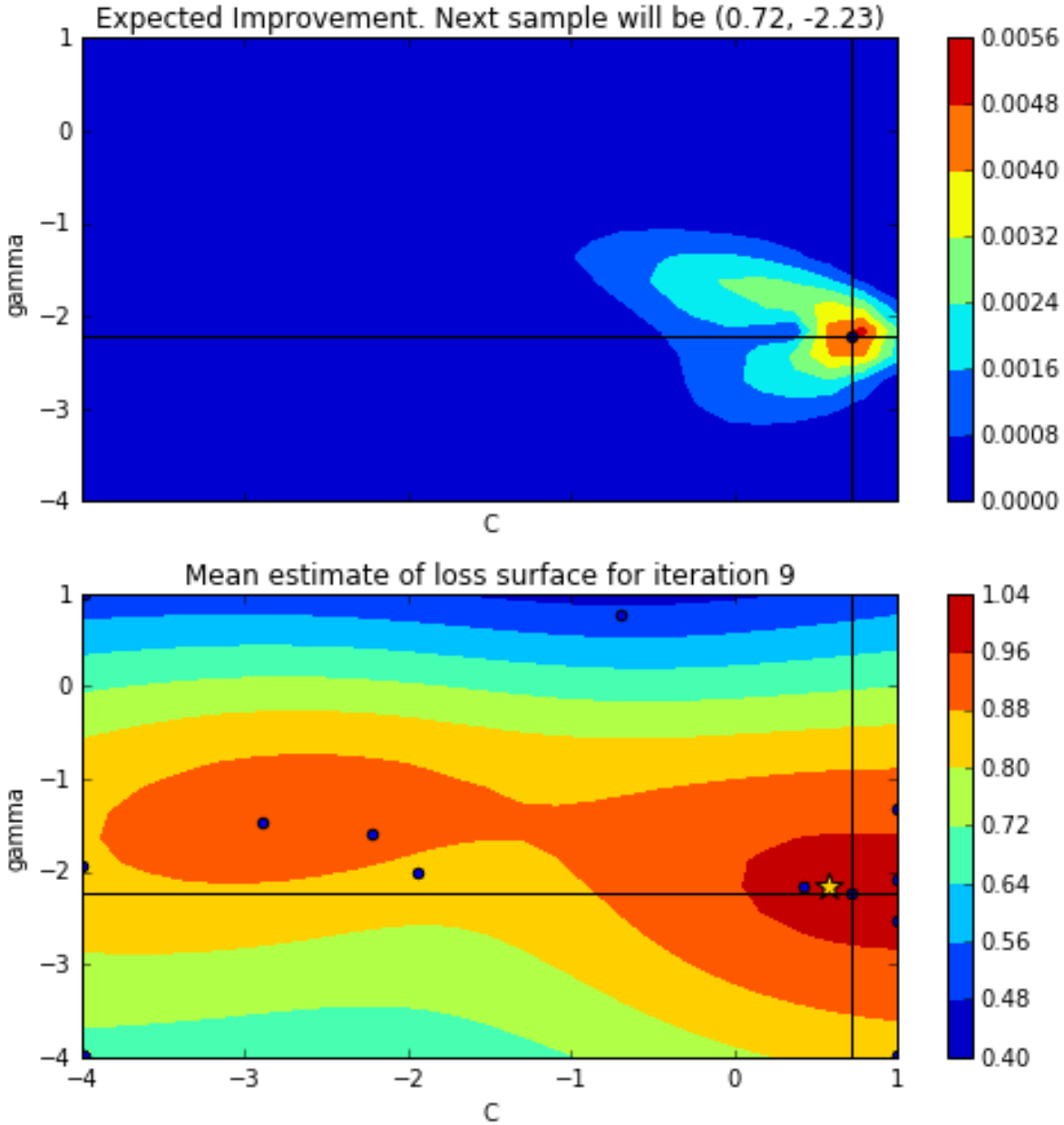
Iterations


Expected Improvement. Next sample will be (1.00, -2.54)
gamma
C
0.0056
0.0048
0.0040
0.0032
0.0024
0.0016
0.0008
0.0000
Mean estimate of loss surface for iteration 7
gamma
C
1.04
0.96
0.88
0.80
0.72
0.64
0.56
0.48
0.40

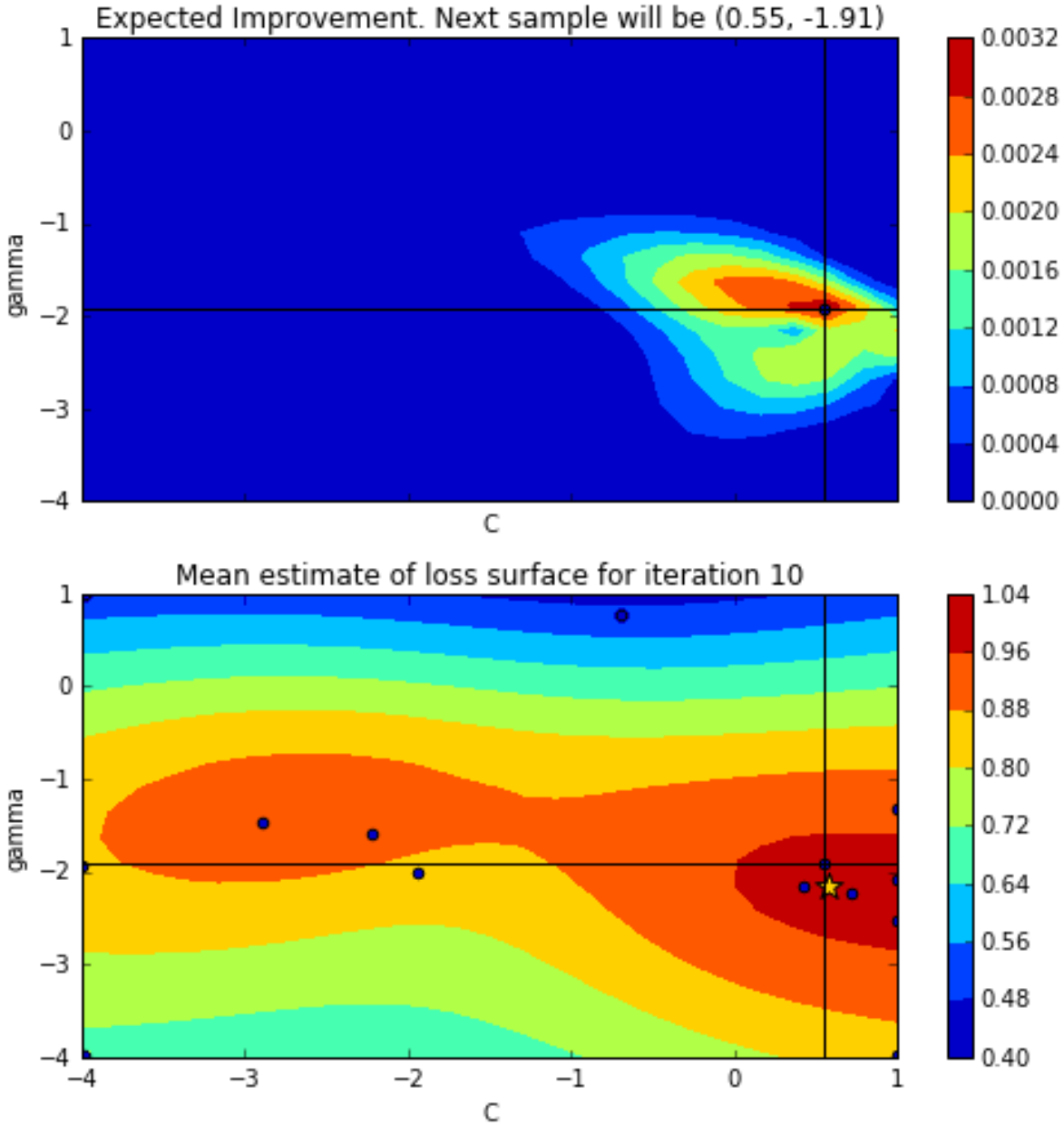
Iterations


Expected Improvement. Next sample will be (0.43, -2.15)
gamma
C
0.0042
0.0036
0.0030
0.0024
0.0018
0.0012
0.0006
0.0000
Mean estimate of loss surface for iteration 8
gamma
C
1.04
0.96
0.88
0.80
0.72
0.64
0.56
0.48
0.40

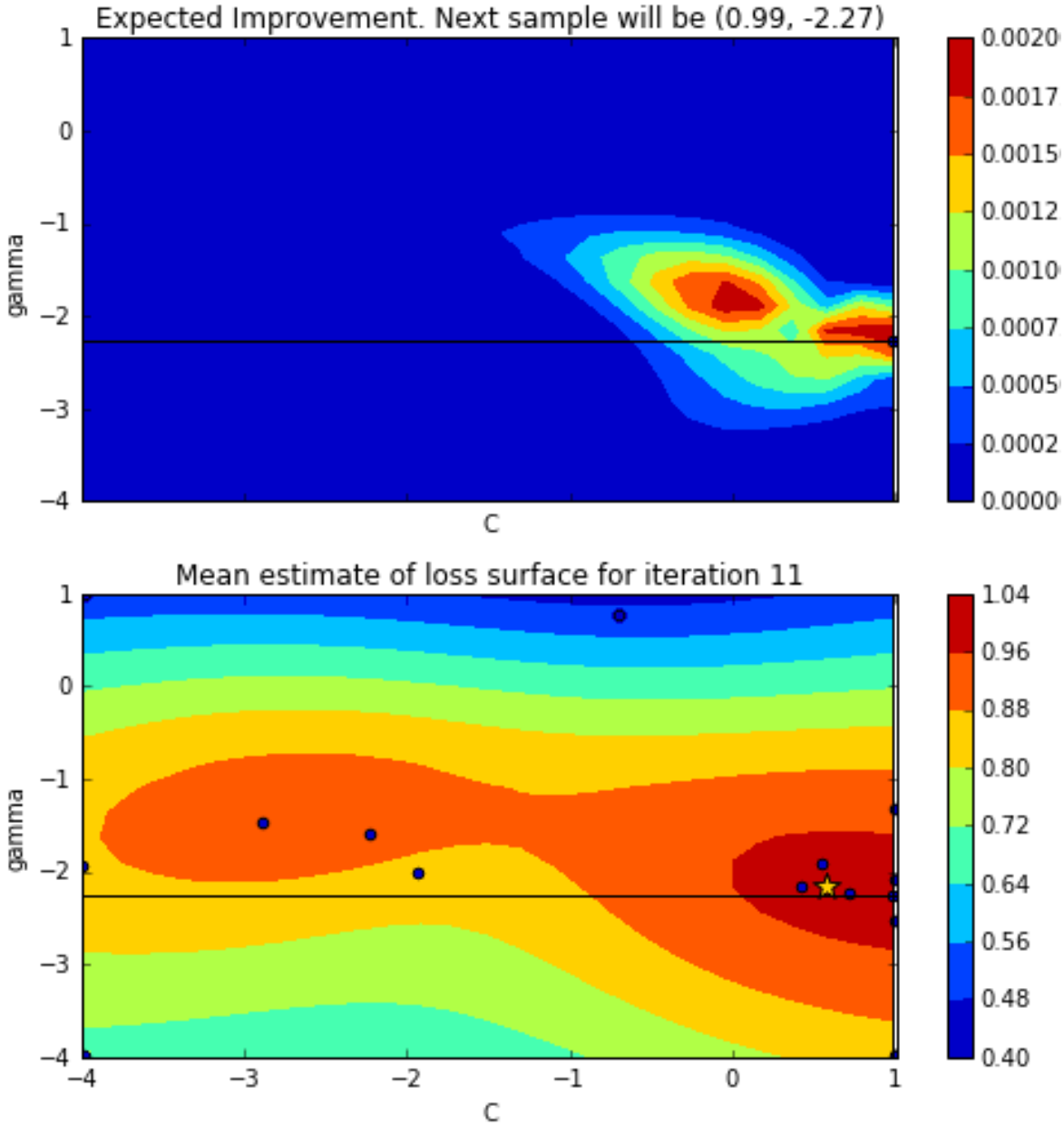
Iterations


Expected Improvement. Next sample will be (0.72, -2.23)
gamma
C
0.0056
0.0048
0.0040
0.0032
0.0024
0.0016
0.0008
0.0000
Mean estimate of loss surface for iteration 9
gamma
C
1.04
0.96
0.88
0.80
0.72
0.64
0.56
0.48
0.40

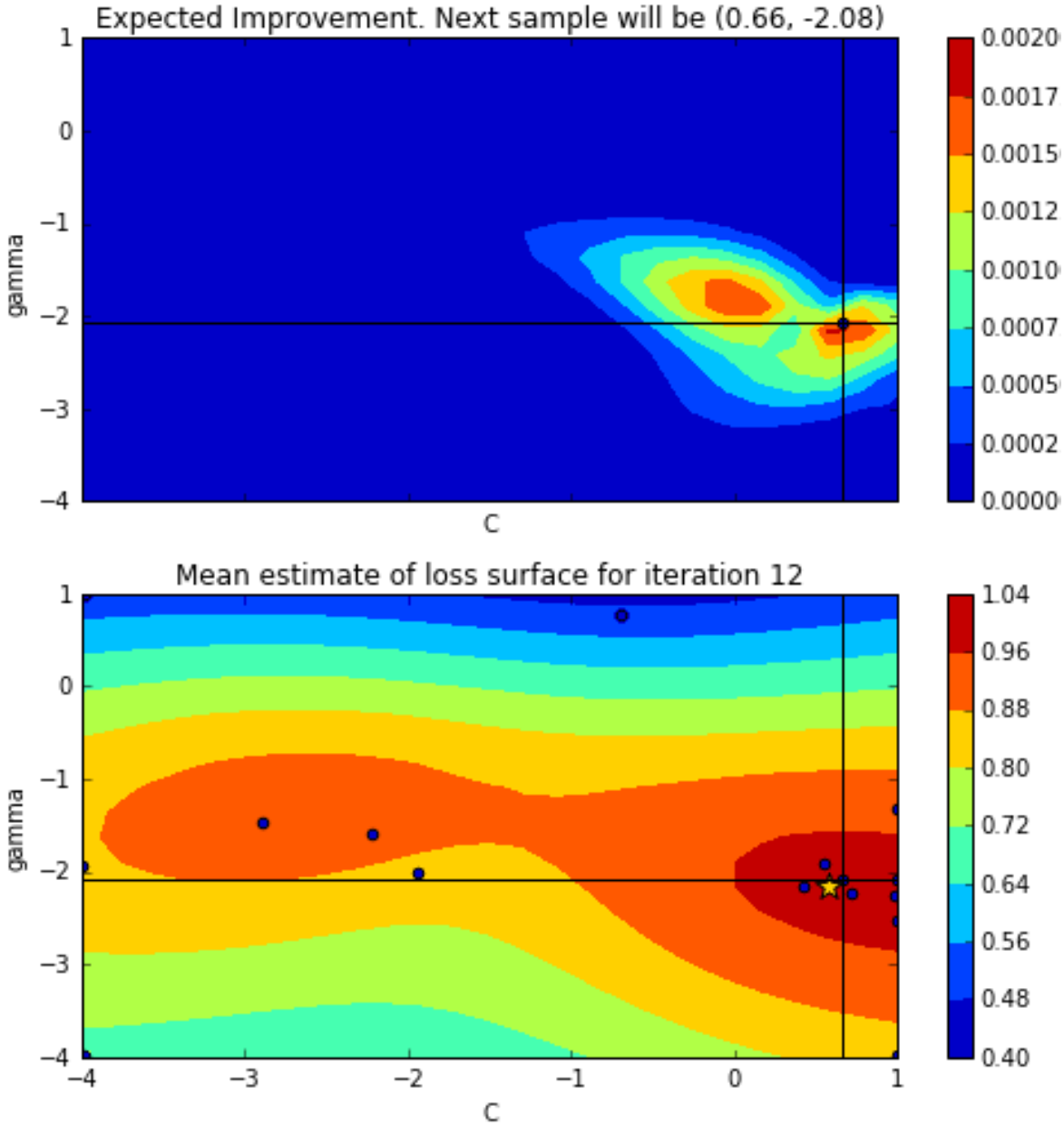
Iterations


Expected Improvement. Next sample will be (0.55, -1.91)
gamma
C
0.0032
0.0028
0.0024
0.0020
0.0016
0.0012
0.0008
0.0004
0.0000
Mean estimate of loss surface for iteration 10
gamma
C
1.04
0.96
0.88
0.80
0.72
0.64
0.56
0.48
0.40

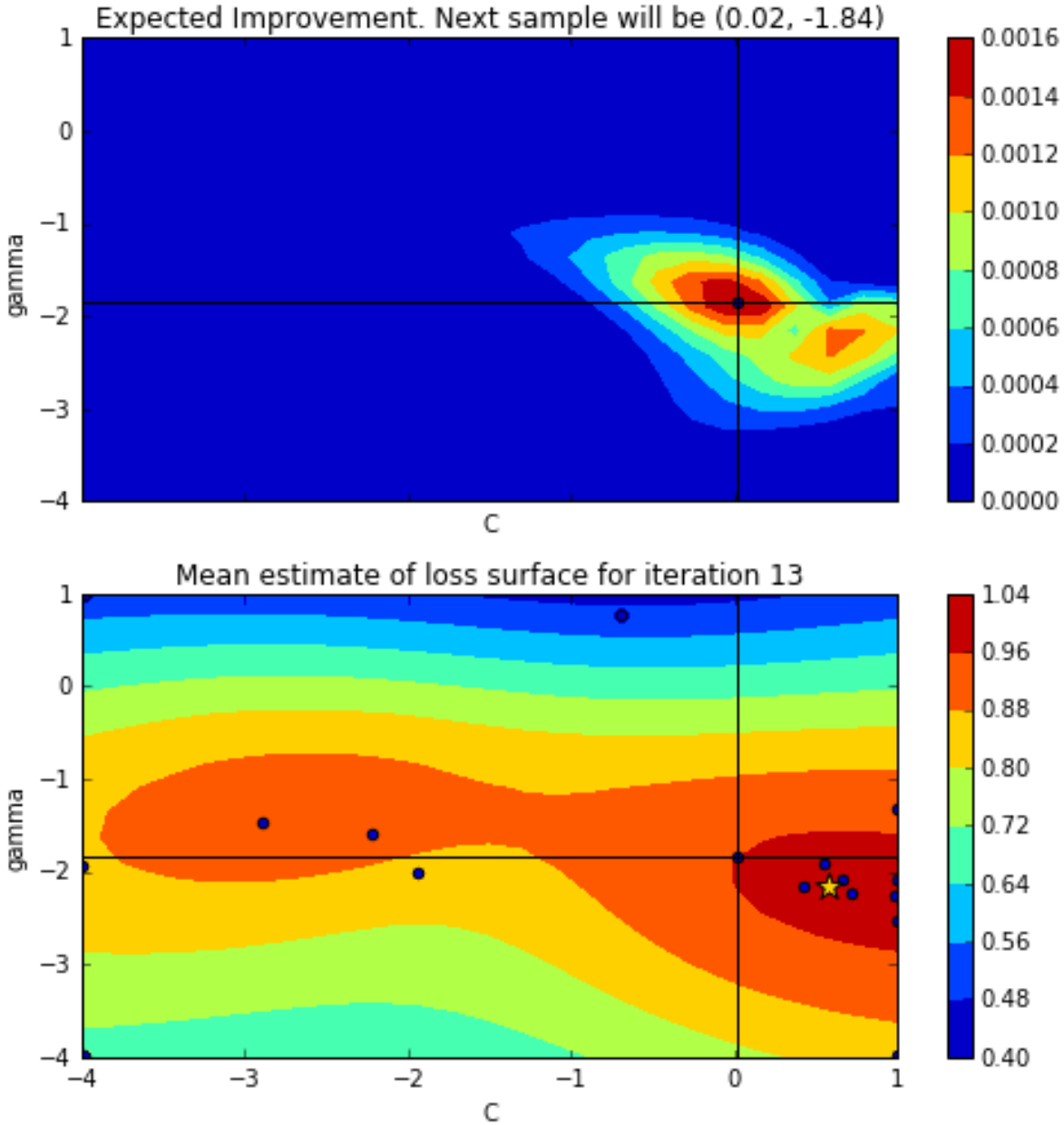
Iterations


Expected Improvement. Next sample will be (0.99, -2.27)
gamma
C
0.0020
0.0017
0.0015
0.0012
0.0010
0.0007
0.0005
0.0002
0.0000
Mean estimate of loss surface for iteration 11
gamma
C
1.04
0.96
0.88
0.80
0.72
0.64
0.56
0.48
0.40

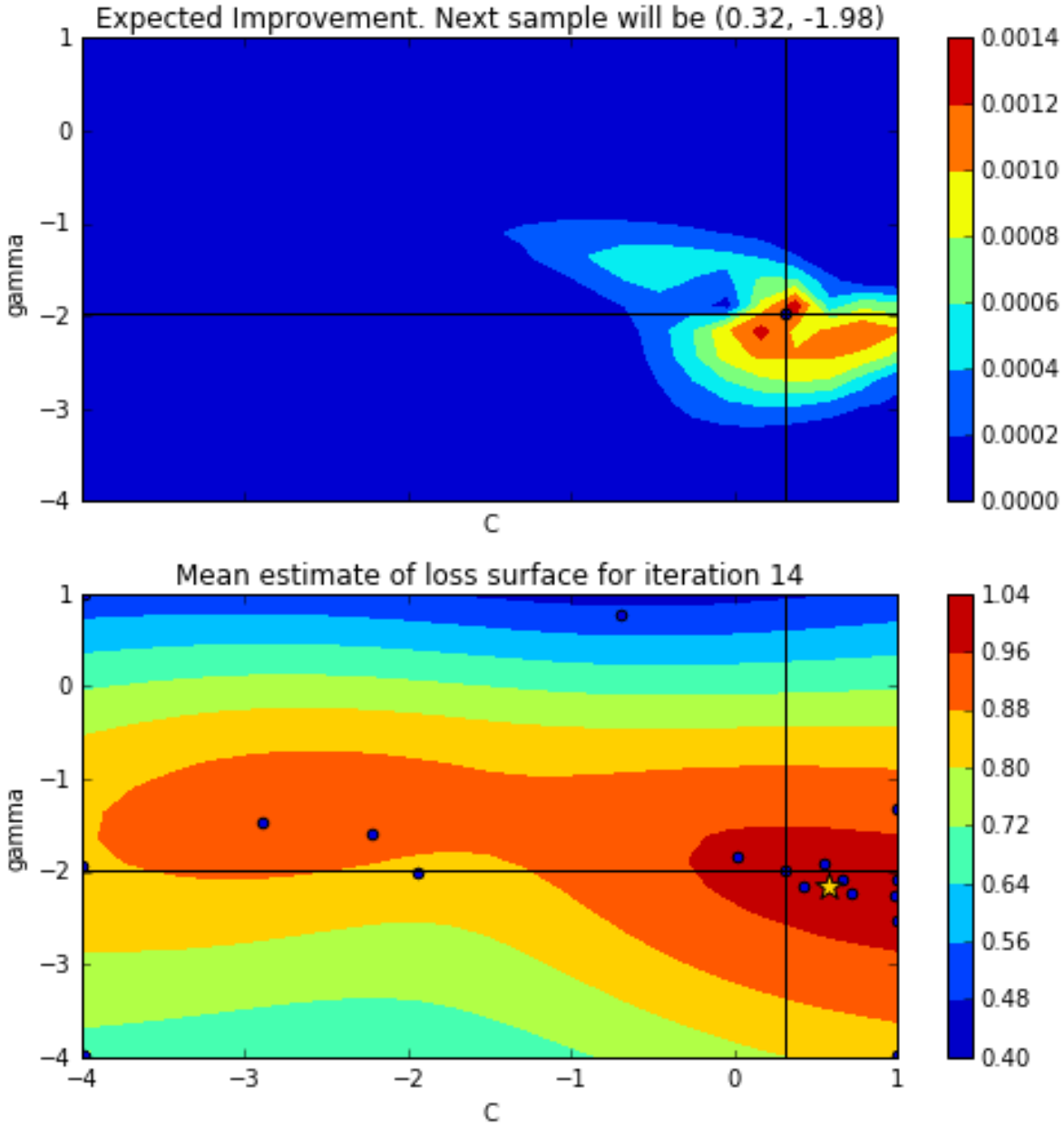
Iterations


Expected Improvement. Next sample will be (0.66, -2.08)
gamma
C
0.0020
0.0017
0.0015
0.0012
0.0010
0.0007
0.0005
0.0002
0.0000
Mean estimate of loss surface for iteration 12
gamma
C
1.04
0.96
0.88
0.80
0.72
0.64
0.56
0.48
0.40

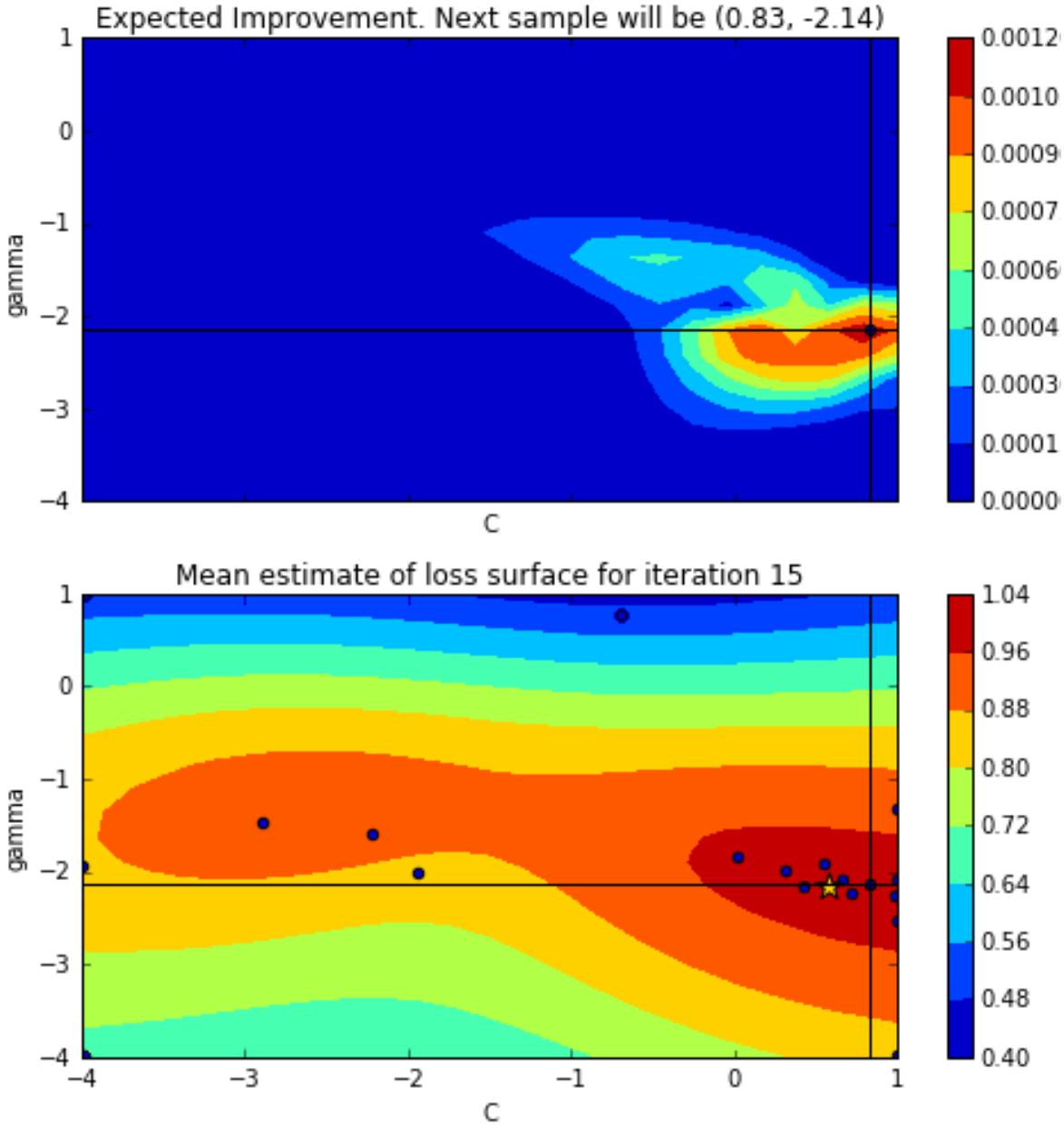
Iterations


Expected Improvement. Next sample will be (0.02, -1.84)
gamma
C
0.0016
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Mean estimate of loss surface for iteration 13
gamma
C
1.04
0.96
0.88
0.80
0.72
0.64
0.56
0.48
0.40

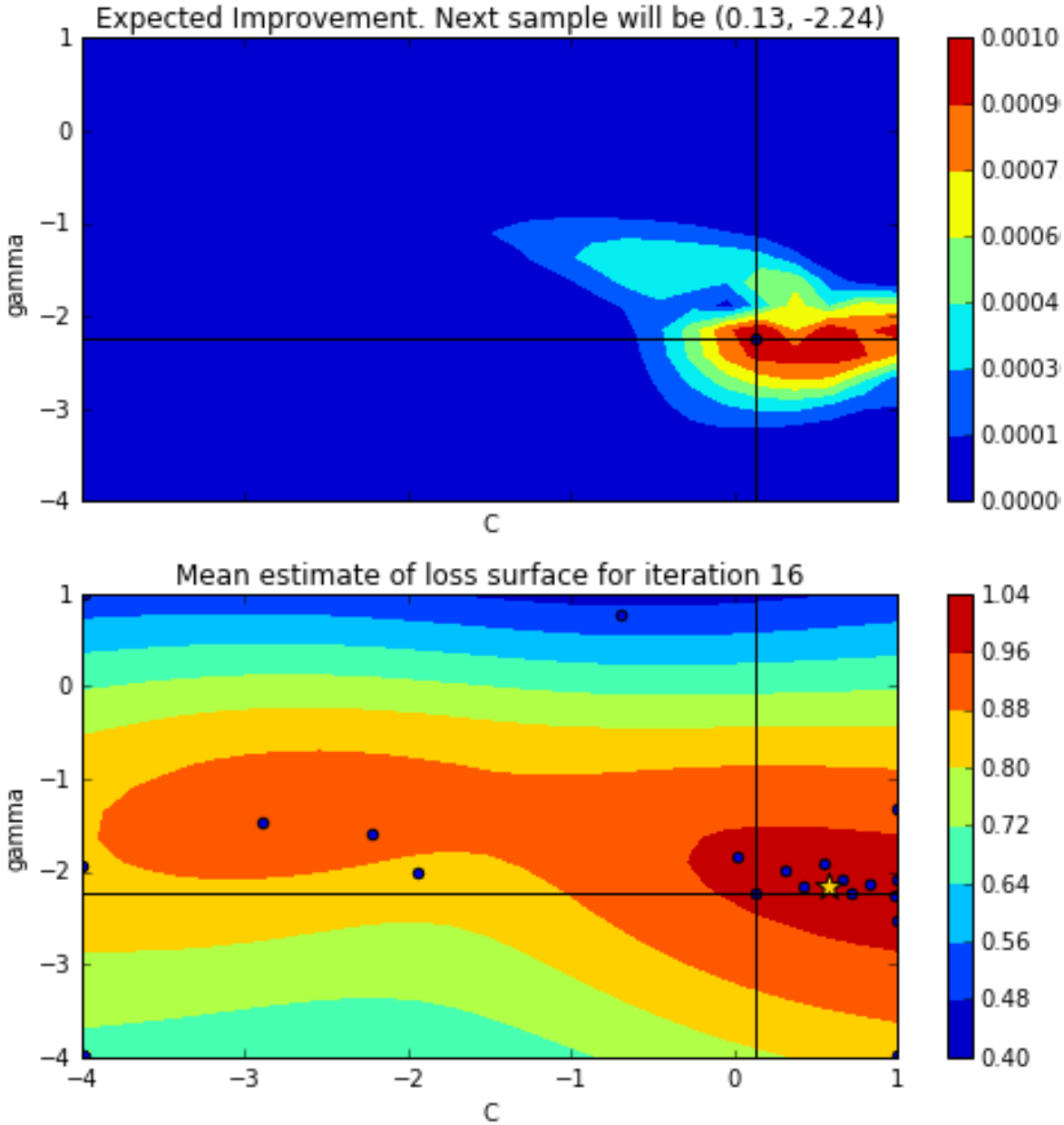
Iterations


Expected Improvement. Next sample will be (0.32, -1.98)
gamma
C
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Mean estimate of loss surface for iteration 14
gamma
C
1.04
0.96
0.88
0.80
0.72
0.64
0.56
0.48
0.40

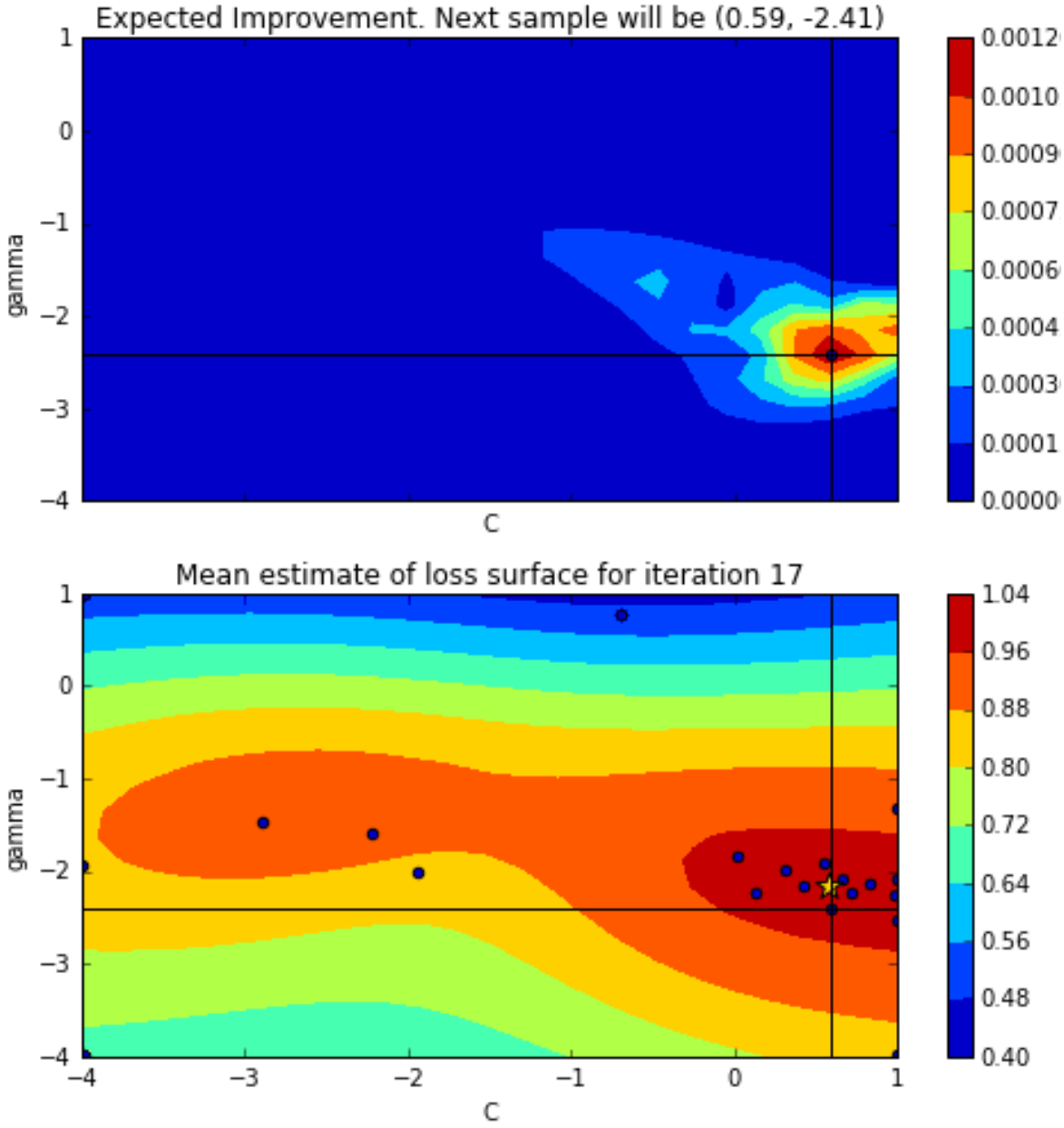
Iterations


Expected Improvement. Next sample will be (0.83, -2.14)
gamma
C
0.0012
0.0010
0.0009
0.0007
0.0006
0.0004
0.0003
0.0001
0.0000
Mean estimate of loss surface for iteration 15
gamma
C
1.04
0.96
0.88
0.80
0.72
0.64
0.56
0.48
0.40

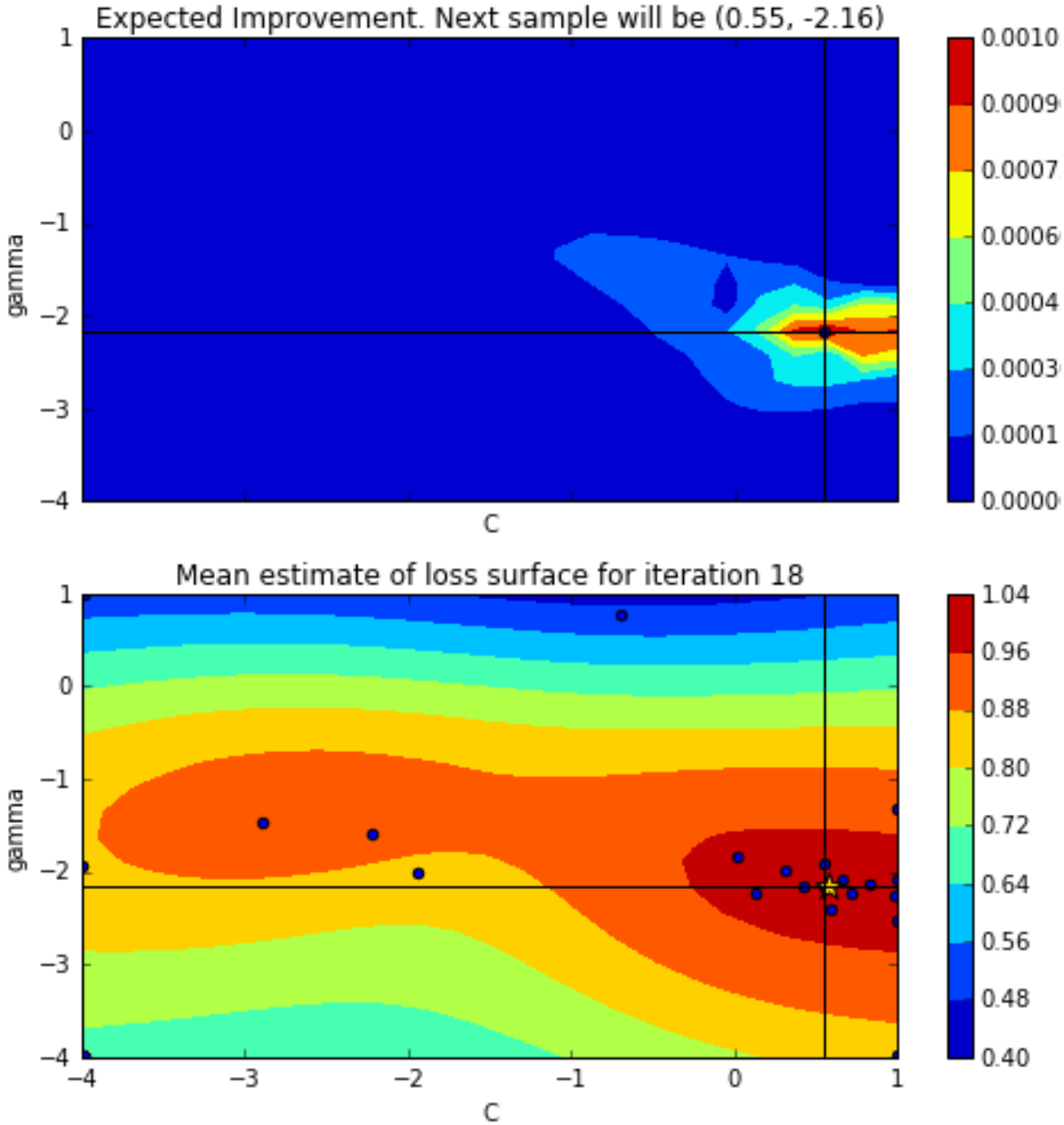
Iterations


Expected Improvement. Next sample will be (0.13, -2.24)
gamma
C
0.0010
0.0009
0.0007
0.0006
0.0004
0.0003
0.0001
0.0000
Mean estimate of loss surface for iteration 16
gamma
C
1.04
0.96
0.88
0.80
0.72
0.64
0.56
0.48
0.40

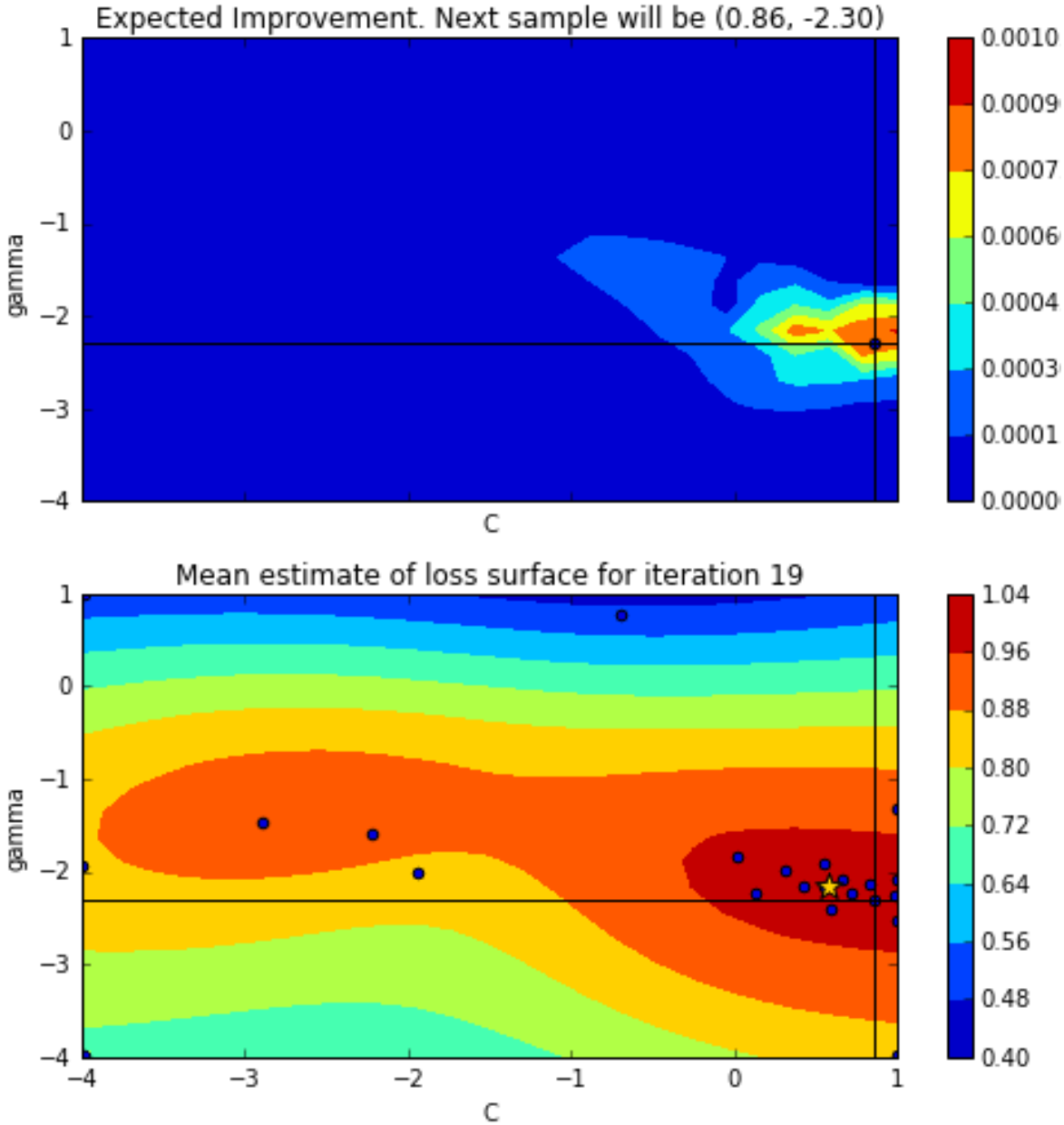
Iterations


Expected Improvement. Next sample will be (0.59, -2.41)
gamma
C
0.0012
0.0010
0.0009
0.0007
0.0006
0.0004
0.0003
0.0001
0.0000
Mean estimate of loss surface for iteration 17
gamma
C
1.04
0.96
0.88
0.80
0.72
0.64
0.56
0.48
0.40

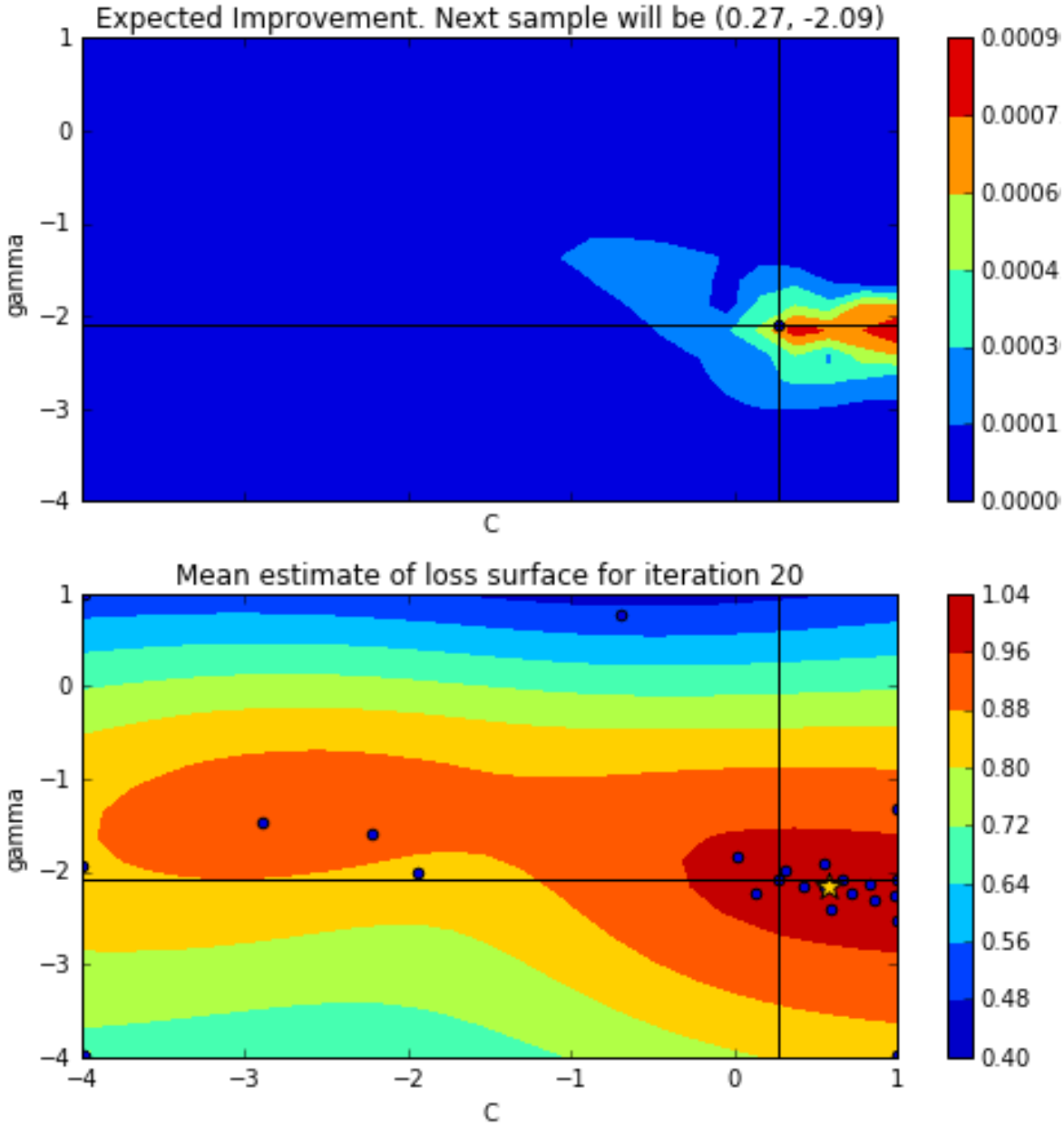
Iterations


Expected Improvement. Next sample will be (0.55, -2.16)
gamma
C
0.0010
0.0009
0.0007
0.0006
0.0004
0.0003
0.0001
0.0000
Mean estimate of loss surface for iteration 18
gamma
C
1.04
0.96
0.88
0.80
0.72
0.64
0.56
0.48
0.40

Iterations


Expected Improvement. Next sample will be (0.86, -2.30)
gamma
C
0.0010
0.0009
0.0007
0.0006
0.0004
0.0003
0.0001
0.0000
Mean estimate of loss surface for iteration 19
gamma
C
1.04
0.96
0.88
0.80
0.72
0.64
0.56
0.48
0.40

Iterations


Expected Improvement. Next sample will be (0.27, -2.09)
gamma
C
0.0009
0.0007
0.0006
0.0004
0.0003
0.0001
0.0000
Mean estimate of loss surface for iteration 20
gamma
C
1.04
0.96
0.88
0.80
0.72
0.64
0.56
0.48
0.40

My message...

- **Simple to use**
  - Just matrix operations (if likelihoods are Gaussian)
  - Few parameters: relatively easy to set or sample over
  - Predictions are often very good
  - Handle uncertainty in unknown function $\mathbf{f}$
  - Can learn kernel parameters automatically from data, no matter how flexible we wish to make the kernel
  - Can incorporate interpretable noise models and priors over functions
  - Can combine automatic feature selection with learning using ARD
- **No magic bullet**: best results need (at least) careful data scaling, which could be modelled or done by hand
- **The need for approximate inference:**
  - Sometimes Gaussian likelihoods arent enough
  - $O(m^3)$ and $O(m^2)$ costs are bad news for big problems

What's next?

- Multi-task and structured GP learning
- Large scale GP based on Random Fourier features
- Deep GP based on random Fourier features
- Deep GP with Deep NN input features
- Multi-fidelity and multi-criteria Bayesian optimization
- Learning to optimize
- Other probabilistic models, e.g. modeling and optimization based on point processes

What else?

- Mercer's theorem: can always decompose covariance function into eigenfunctions and eigenvalues

$$K(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \psi_i(\mathbf{x}) \psi_i(\mathbf{x}')$$

- If sum is finite — back to linear regression. Often sum is infinite, an no analytical expression for eigenfunctions
- However, a lot of kernels can be represented as

$$K(\mathbf{x}, \mathbf{x}') = \int_{\mathbb{R}^d} \phi(\mathbf{x}, \mathbf{w}) \phi(\mathbf{x}', \mathbf{w}) p(\mathbf{w}) d\mathbf{w}$$

Thus we can approximate kernels using random sampling and get approximate non-linear kernel models in linear time $O(m)$

Let us consider the squared exponential covariance function

$$K(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\sum_{i=1}^{d} \theta_i^2 (x_i - x_i')^2\right)$$

When optimizing the likelihood wrt covariance function parameters $\left\{\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d), \sigma_f, \sigma\right\}$, the following weird behavior can occur

- MLE optimum is located in the area where $\|\boldsymbol{\theta}\| \sim 0$, and the conditional number of $\mathbf{K}$ is big
- For $\|\boldsymbol{\theta}\| \to \infty$ matrix $\mathbf{K} \to \mathrm{I}_m$. Thus, we have a degenerate approximation
- Dependence of the likelihood $\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$ on $\boldsymbol{\theta}$ can be weak

Degenerate approximation

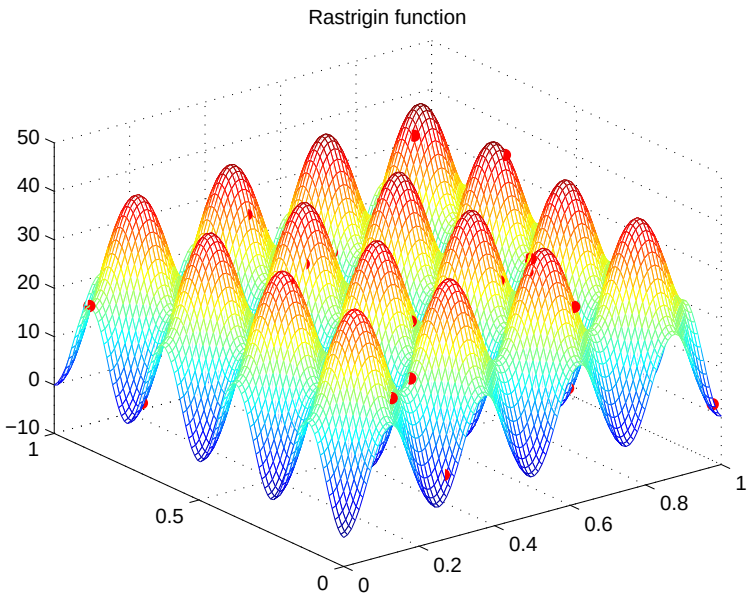


Figure – Rastrigin function

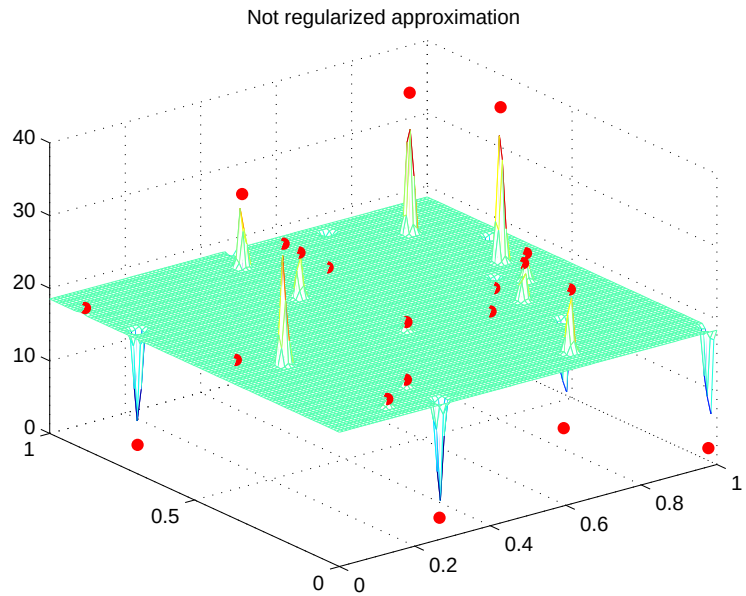
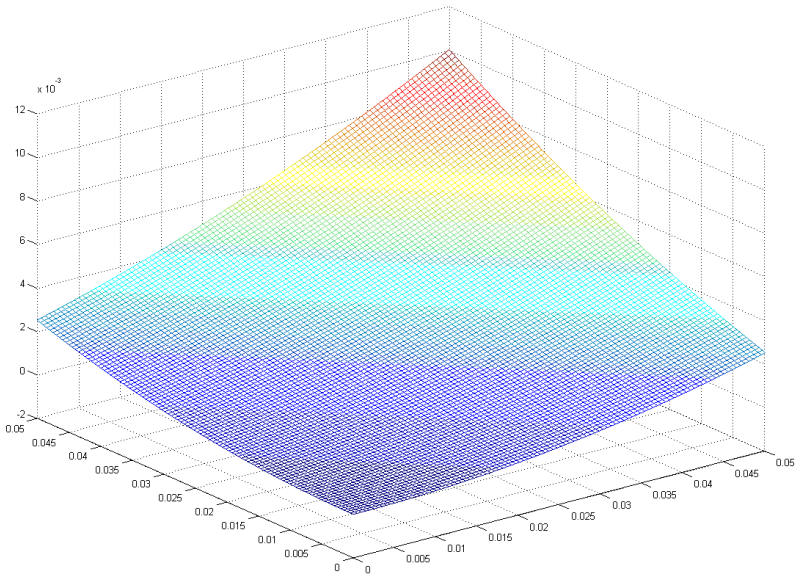


Figure – An approximation of Rastrigin function

Numerical noise



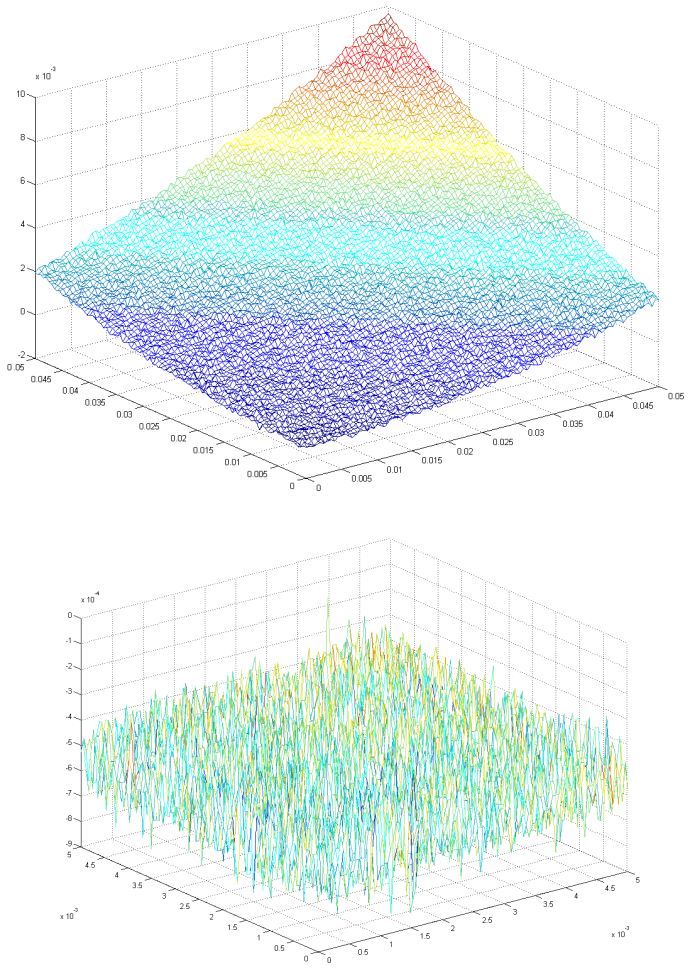


Figure – $y(\mathbf{x}) = (x_1 + x_2)^2$

Let us introduce the following priors on covariance function parameters:

$$\theta_i^2 \in \Gamma(\alpha, \beta), i = 1, \ldots, d$$
$$\sigma^2 \in \Gamma(\alpha_\sigma, \beta_\sigma)$$

Thus log-prior of parameters has the form

$$\log p(\{\boldsymbol{\theta}, \sigma\}) = m(\alpha \log \beta - \log \Gamma(\alpha)) + 2(\alpha - 1) \sum_{i=1}^{m} \log \theta_i +$$
$$- \beta \sum_{i=1}^{m} \theta_i^2 + (\alpha_\sigma \log \beta_\sigma - \log \Gamma(\alpha_\sigma)) + 2(\alpha_\sigma - 1) \log \sigma - \beta_\sigma \sigma^2$$

We obtain MAP estimates for $\{\boldsymbol{\theta}, \sigma\}$:

$$\log p(\mathbf{y}|\mathbf{X}, \{\boldsymbol{\theta}, \sigma\}) + \log p_{\alpha,\beta,\alpha_\sigma,\beta_\sigma}(\{\boldsymbol{\theta}, \sigma\})$$
$$+ \log p(\alpha, \beta, \alpha_\sigma, \beta_\sigma) \to \max_{\{\boldsymbol{\theta},\sigma\},\{\alpha,\beta,\alpha_\sigma,\beta_\sigma\}}$$

Here parameters $\{\alpha, \beta, \alpha_\sigma, \beta_\sigma\}$ are also optimized along with imposed hyper-prior $p(\alpha, \beta, \alpha_\sigma, \beta_\sigma)$, parameters of which are fixed to some constants

Regularized approximation vs. degeneracy

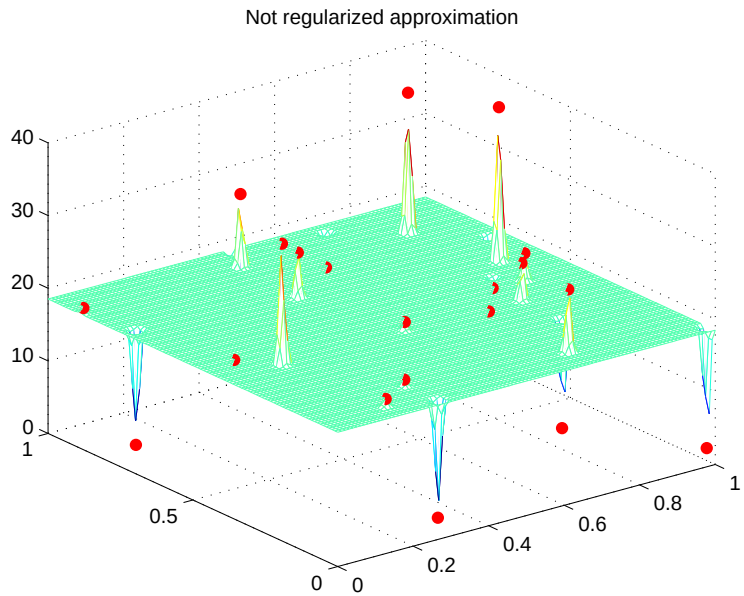


Figure – Degenerate approximation

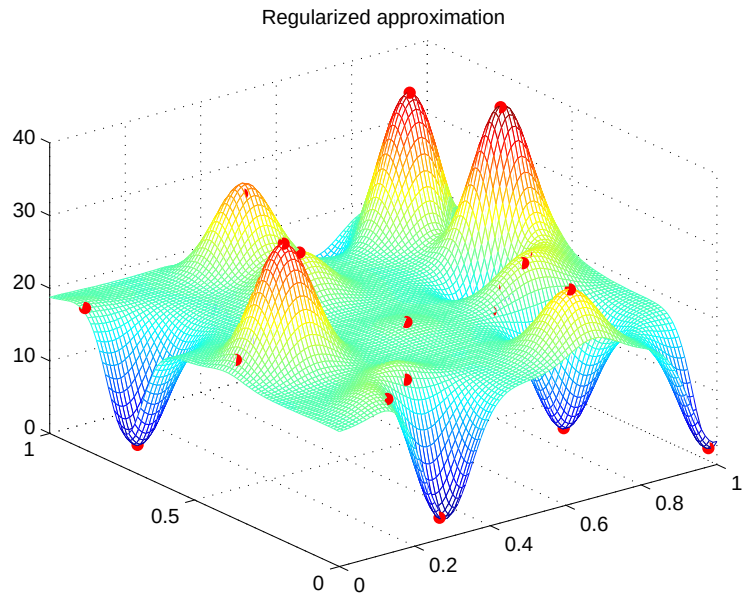
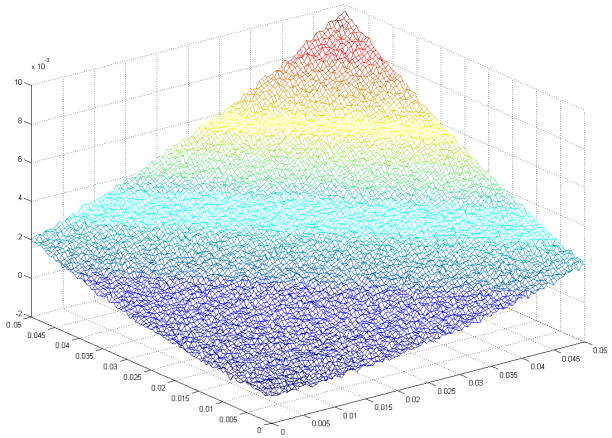
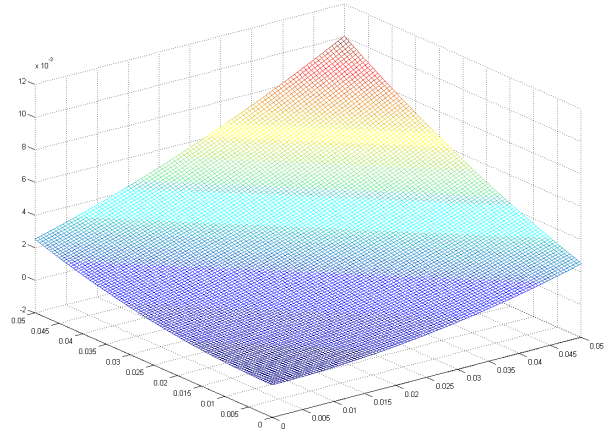
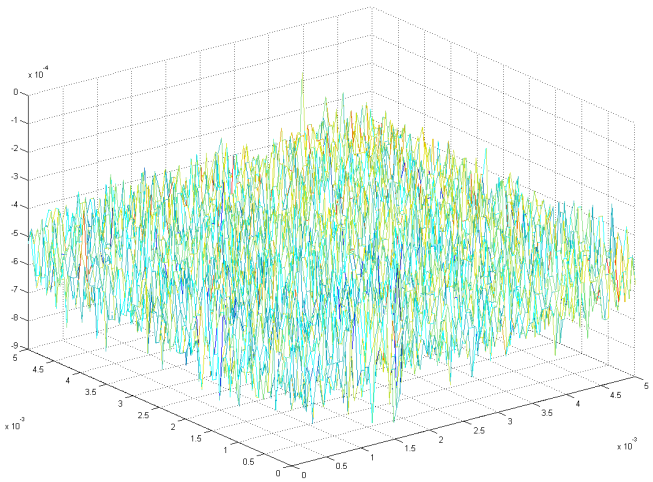


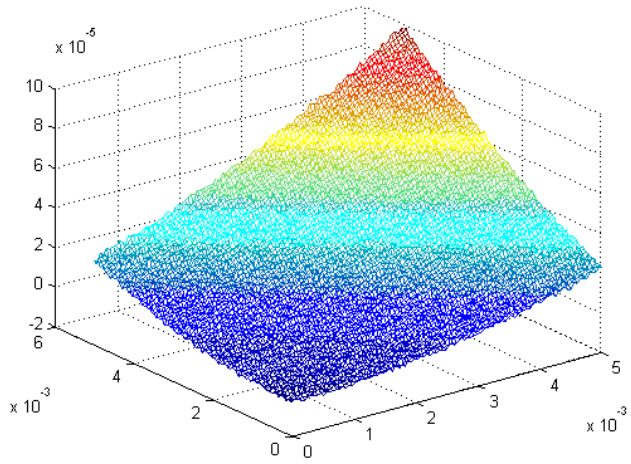
Figure – Approximation with imposed Bayesian regularization

Regularized approximation vs. numerical noise

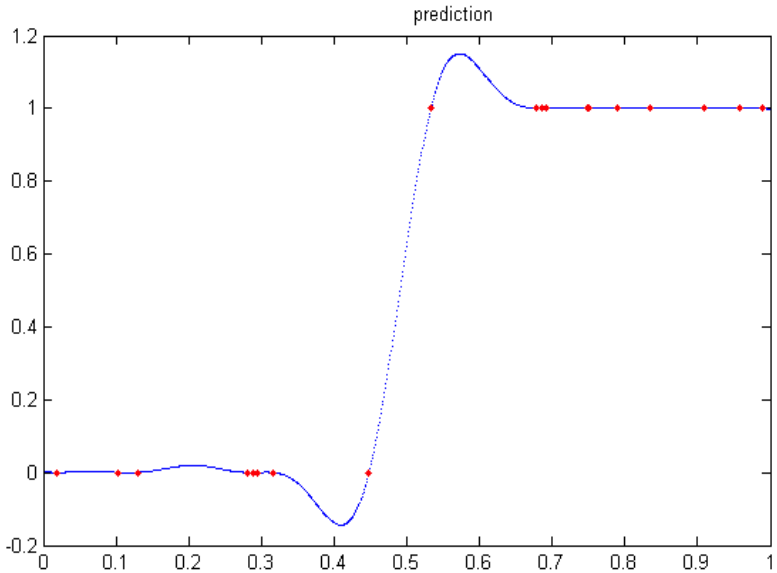








Approximation of Heaviside function for $x \in [0, 1]$, training sample size $m = 20$



Let us assume that

$$y(\mathbf{x}) = \sum_{i=1}^{Q} \alpha_i \psi_i(\mathbf{x}) + f(\mathbf{x}) + \varepsilon(\mathbf{x}),$$

where

- $f(\mathbf{x})$ is a GP,
- $\varepsilon(\mathbf{x})$ is a Gaussian white noise,
- $\{\alpha_i\}_{i=1}^{Q}$ are i.i.d. r.v. with zero mean and variance $\frac{\sigma_0^2}{Q}$,
- $\left\{\psi_i(\mathbf{x})\right\}_{i=1}^{Q}$ is a set of functions

Thus the covariance function of $y(\mathbf{x})$ has the form:

$$\overline{K}(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') + \sigma_0^2 \boldsymbol{\psi}^{\mathrm{T}}(\mathbf{x}) \boldsymbol{\psi}(\mathbf{x}') + \sigma^2 \delta(\mathbf{x}, \mathbf{x}'),$$

where $\boldsymbol{\psi}(\mathbf{x}) = \{\psi_i(\mathbf{x}), i = 1, \ldots, Q\}$

Posterior mean of $y(\mathbf{x})$ has the form:

$$\mu_*(\mathbf{x}) = \overline{\mathbf{k}}(\mathbf{x})\overline{\mathbf{K}}^{-1}\mathbf{y},$$

where

- $\Psi = \{\boldsymbol{\psi}(\mathbf{x}_i), i = 1, \ldots, m\}$,
- $\overline{\mathbf{k}}(\mathbf{x}) = \mathbf{k}(\mathbf{x}) + \sigma_0^2\psi^{\mathrm{T}}(\mathbf{x})\Psi$,
- $\overline{\mathbf{K}} = \mathbf{K} + \sigma_0^2\Psi^{\mathrm{T}}\Psi + \sigma^2\mathrm{I}_m$

Posterior variance has the form:

$$\sigma_*^2(\mathbf{x}) = K(\mathbf{x}, \mathbf{x}) + \sigma^2 + \sigma_0^2\psi^{\mathrm{T}}(\mathbf{x})\psi(\mathbf{x}) - \overline{\mathbf{k}}^{\mathrm{T}}(\mathbf{x})\overline{\mathbf{K}}^{-1}\overline{\mathbf{k}}(\mathbf{x})$$

Minus log-likelihood has a similar structure compared to a stationary process, but we should replace $\mathbf{K}$ by $\overline{\mathbf{K}}$:

$$-\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \frac{1}{2}\mathbf{y}^{\mathrm{T}}\overline{\mathbf{K}}^{-1}\mathbf{y} + \frac{1}{2}\log|\overline{\mathbf{K}}| - \frac{m}{2}\log 2\pi,$$

where $\{\boldsymbol{\theta}, \sigma_0, \sigma\}$ are parameters

If the set of functions is fixed, then estimates of parameters can be obtained via usual maximization of the log-likelihood

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) \to \max_{\boldsymbol{\theta}}$$

Let us consider three families of functions:

1. Sigmoid functions

$$\psi_j(\mathbf{x}) = \sigma\left(\sum_{i=1}^{d} \beta_{j,i} x_i\right),$$

where $\sigma(x) = \frac{e^x - 1}{e^x + 1}$, $\beta_{j,i} \in \mathbb{R}$ are parameters

2. Radial basis functions (RBF)

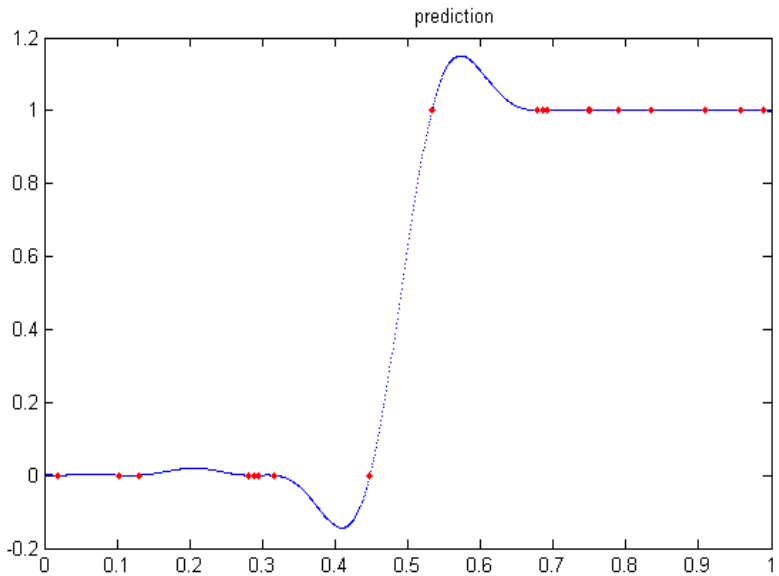
$$\psi_j(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|_2^2}{r_j^2}\right),$$

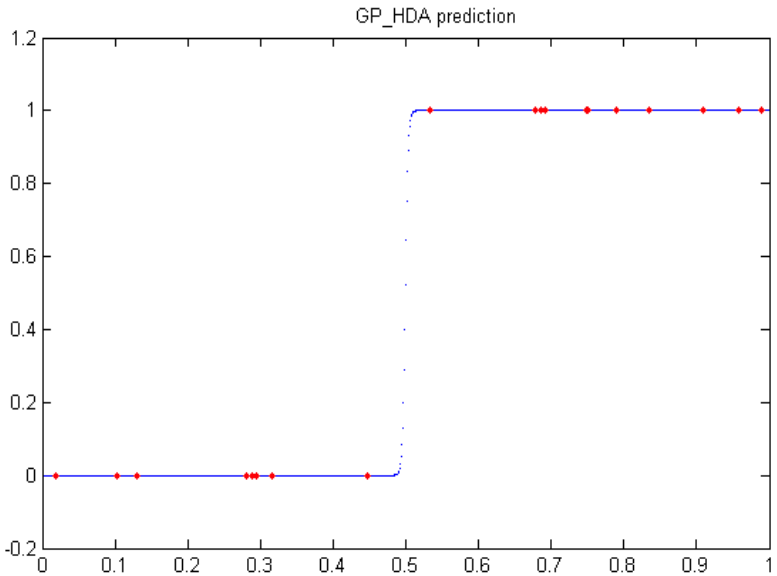
where $\mathbf{c}_j \in \mathbb{R}^d, r_j \in \mathbb{R}$ are parameters of the function.

3. Linear basis functions

$$\psi_j(\mathbf{x}) = x_j, j = 1, 2, \ldots, d$$

Approximation of Heaviside function for $x \in [0, 1]$, training sample size $m = 20$





Bibliography: books, papers

- Forrester, A., Sobester, A., and Keane, A. (2008). Engineering design via surrogate modelling: a practical guide. Wiley, West Sussex, UK.
- Rasmussen, C. and Williams, C. (2006). Gaussian Processes for Machine Learning. MIT Press, Cambridge, MA.
- Cressie, N. (1993). Statistics for Spatial Data, revised edition. Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics. Wiley Interscience, New York
- Jones, D., Schonlau, M., and Welch, W. (1998). Efficient Global Optimization of Expensive Black-Box Functions. Journal of Global Optimization, 13(4):455-492.
- Villemonteix, J., Vazquez, E., and Walter, E. (2009). An informational approach to the global optimization of expensive-to-evaluate functions. Journal of Global Optimization, 44(4):509534.
- B. Scholkopf and A.J. Smola. Learning with Kernels, Support Vector Machines, Regularization, Optimization, and Beyond. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, MA, USA, 2002.

Bibliography: tutorials

- Javier Gonzalez, Gaussian Processes for Global Optimization, 2015
- Novi Quadrianto, Gaussian Processes, 2017
- Ryan P. Adams, Tutorial on! Bayesian Optimization for Machine Learning
- Peter I. Frazier, Tutorial: Bayesian Methods for Global and Simulation Optimization, 2011
- Marc Deisenroth, Gaussian Processes for Big Data Problems, 2015
- Ed Snelson, Tutorial: Gaussian process models for machine learning, 2006
- Zoubin Ghahramani, A Tutorial on Gaussian Processes (or why I don't use SVMs), 2011
- Iain Murray, Introduction to Gaussian Process, 2008
- Cedric Archambeau, Bayesian Optimization, 2016

Bibliography: additional reading

- Burnaev E., Zaytsev A. Minimax approach to variable fidelity data interpolation. Proceedings of Machine Learning Research 54:652-661, Volume 54: Artificial Intelligence and Statistics, 20-22 April 2017, Fort Lauderdale, FL, USA.
- Burnaev E., Zaytsev A. Large Scale Variable Fidelity Surrogate Modeling. Ann Math Artif Intell (2017), pp. 1-20. doi:10.1007-s10472-017-9545-y
- E. Burnaev, M. Belyaev, E. Kapushev. Computationally efficient algorithm for Gaussian Processes based regression in case of structured samples. Computational Mathematics and Mathematical Physics, 2016, Vol. 56, No. 4, pp. 499-513, 2016.
- E. Burnaev, M. Panov, A. Zaytsev. Regression on the Basis of Nonstationary Gaussian Processes with Bayesian Regularization, Journal of Communications Technology and Electronics, 2016, Vol. 61, No. 6, pp. 661-671.
- Burnaev E, Nazarov I. Conformalized Kernel Ridge Regression. 15th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE Conference Publications, pp. 45 - 52, 2016. DOI: 10.1109-ICMLA.2016.0017
- E. Burnaev and M. Panov. Adaptive design of experiments based on gaussian processes. In Lecture Notes in Artificial Intelligence. Proceedings of SLDS 2015. A. Gammerman et al. (Eds.), volume 9047, pages 116-126, London, UK, April 20-23 2015. Springer.
- E. Burnaev, V. Vovk. Efficiency of conformalized ridge regression. JMLR 35:605-622, 2014
- A. Zaytsev, E. Burnaev, V. Spokoiny. Properties of the Bayesian Parameter Estimation of a Regression Based on Gaussian Processes. Journal of Mathematical Sciences, Volume 203, Issue 6, pp. 789-798, 15 Nov 2014.