Information Theory and Inference

Bayesian Optimization using Gaussian Process: Implementation from Scratch

Eugenio Fella, Theivan Pasupathipillai, Matteo Pedrazzi, Gaetano Ricucci, Carlo Sgorlon Gaiatto

Introduction

Bayesian optimization is a powerful tool for finding the global optimum of an expensive and black-box objective function.

It combines a probabilistic model, usually a **Gaussian process**, and an acquisition function to efficiently optimize the objective function with a small number of function evaluations.

Bayesian optimization has been successfully applied in various domains such as robotics, environmental monitoring, and automatic machine learning.

Bayesian linear regression

Let's consider a linear regression model:

$$\mathbf{y} = \mathbf{X}\mathbf{w} \qquad \text{where} \qquad \mathbf{X} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{pmatrix} \qquad \text{with} \qquad \mathbf{x}_i, \mathbf{w} \in \mathbb{R}^{d+1}$$

The **posterior** probability of the regression coefficients is:

$$p(\mathbf{w} \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{w})\, p(\mathbf{w})}{p(\mathbf{y})}$$

Likelihood and prior

Let's assume that each observation is Gaussian distributed with fixed variance. Therefore the **likelihood** is:

$$p(\mathbf{y} \mid \mathbf{w}) = \mathcal{N}(\mathbf{y} \mid \mathbf{X}\mathbf{w}, \boldsymbol{\Sigma}) \quad \text{where} \quad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_N^2 \end{pmatrix}$$

The **prior** is chosen to be a conjugate prior, namely it is also Gaussian:

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} \mid \mu, \mathbf{V_0})$$

Joint probability

The **joint probability** is the product of two multivariate Gaussian distributions that can be shown to be again Gaussian:

$$p(\mathbf{w}, \mathbf{y}) = p(\mathbf{y} \mid \mathbf{w})\, p(\mathbf{w}) = \mathcal{N}\left(\mathbf{w}, \mathbf{y} \mid \mu', \mathbf{\Sigma}'\right)$$

where

$$\mu' = \begin{pmatrix} \mu \\ \mathbf{X}\mu \end{pmatrix} \qquad \mathbf{\Sigma}' = \begin{pmatrix} \mathbf{V_0} & \mathbf{V_0}\mathbf{X}^T \\ \mathbf{X}\mathbf{V_0} & \mathbf{\Sigma} + \mathbf{X}\mathbf{V_0}\mathbf{X}^T \end{pmatrix}$$

Posterior

The **posterior** can be found by starting with the joint probability and showing that the conditional probability is also Gaussian:

$$p(\mathbf{w} \mid \mathbf{y}) = \mathcal{N}\left(\mathbf{w} \mid \mu_{\mathbf{w}|\mathbf{y}}, \boldsymbol{\Sigma}_{\mathbf{w}|\mathbf{y}}\right)$$

where

$$\mu_{\mathbf{w}|\mathbf{y}} = \mu + \mathbf{V}_0 \mathbf{X}^T \left(\boldsymbol{\Sigma} + \mathbf{X}\mathbf{V}_0\mathbf{X}^T\right)^{-1} (\mathbf{y} - \mathbf{X}\mu)$$

$$\boldsymbol{\Sigma}_{\mathbf{w}|\mathbf{y}} = \mathbf{V}_0 - \mathbf{V}_0 \mathbf{X}^T \left(\boldsymbol{\Sigma} + \mathbf{X}\mathbf{V}_0\mathbf{X}^T\right)^{-1} \mathbf{X}\mathbf{V}_0$$

Bayesian nonparametric linear regression

This approach allows to make predictions at new locations:

$$\mathbf{y}^* = \mathbf{X}^*\mathbf{w}$$

It can be shown that the **distribution of predictions** given the observation is again Gaussian and independent of the regression coefficients:

$$p(\mathbf{y}^* \mid \mathbf{y}) = \mathcal{N}\left(\mathbf{y}^* \mid \mu_{\mathbf{y}^*|\mathbf{y}}, \boldsymbol{\Sigma}_{\mathbf{y}^*|\mathbf{y}},\right)$$

where

$$\mu_{\mathbf{y}^*|\mathbf{y}} = \mathbf{X}^*\mu + \mathbf{X}^*\mathbf{V}_0\mathbf{X}^T\left(\boldsymbol{\Sigma} + \mathbf{X}\mathbf{V}_0\mathbf{X}^T\right)^{-1}(\mathbf{y} - \mathbf{X}\mu)$$

$$\boldsymbol{\Sigma}_{\mathbf{y}^*|\mathbf{y}} = \mathbf{X}^*\mathbf{V}_0\mathbf{X}^{*T} - \mathbf{X}^*\mathbf{V}_0\mathbf{X}^T\left(\boldsymbol{\Sigma} + \mathbf{X}\mathbf{V}_0\mathbf{X}^T\right)^{-1}\mathbf{X}\mathbf{V}_0\mathbf{X}^{*T}$$

Kernel trick

To increase the expressiveness of the model it is common to use a non linear **feature mapping**:

$$\mathbf{\Phi} = \phi(\mathbf{X})$$

The **kernel trick** allows us to specify an intuitive similarity between pairs of points, rather than a feature map, which in practice can be hard to define:

$$\mathbf{K}_{\mathbf{XX}}(i,j) = \mathbf{\Phi}\left(\mathbf{x}_i\right)\mathbf{V}_0\mathbf{\Phi}\left(\mathbf{x}_j\right)^T = \left\langle \mathbf{\Phi}\left(\mathbf{x}_i\right), \mathbf{\Phi}\left(\mathbf{x}_j\right)\right\rangle_{\mathbf{V}_0}$$

Kernels

There are several stationary kernels, which are shift invariant:

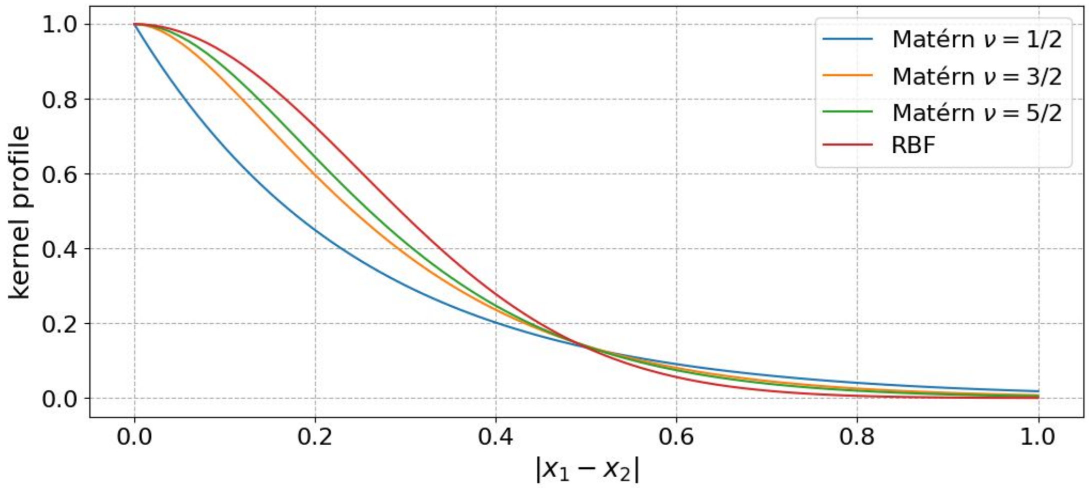
$$
\begin{aligned}
k_{\text{MATERRN 1}}\left(\mathbf{x}, \mathbf{x}'\right) &= \theta_0^2 \exp(-r) \\
k_{\text{MATÉRN3}}\left(\mathbf{x}, \mathbf{x}'\right) &= \theta_0^2 \exp(-\sqrt{3}r)(1 + \sqrt{3}r) \\
k_{\text{MATÉRN5}}\left(\mathbf{x}, \mathbf{x}'\right) &= \theta_0^2 \exp(-\sqrt{5}r)\left(1 + \sqrt{5}r + \frac{5}{3}r^2\right) \\
k_{\text{Sq}-\text{exp}}\left(\mathbf{x}, \mathbf{x}'\right) &= \theta_0^2 \exp\left(-1/2r^2\right),
\end{aligned}
$$

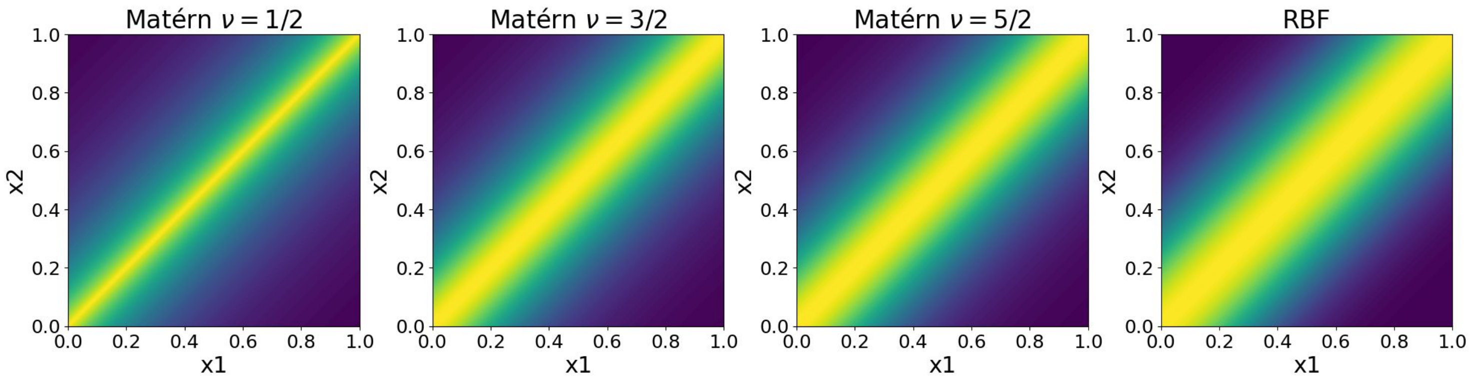
where

$$r^2 = \left(\mathbf{x} - \mathbf{x}'\right)^T \mathbf{\Lambda} \left(\mathbf{x} - \mathbf{x}'\right)$$

Equations taken from this paper.

Kernels


Matérn $\nu = 1/2$
Matérn $\nu = 3/2$
Matérn $\nu = 5/2$
RBF
kernel profile
0.0
0.2
0.4
0.6
0.8
1.0
$|x_1 - x_2|$


Matérn $\nu = 1/2$
Matérn $\nu = 3/2$
Matérn $\nu = 5/2$
RBF
x1
x2
0.0
0.2
0.4
0.6
0.8
1.0

Gaussian process

Finally we get:

$$p(\mathbf{y}^* \mid \mathbf{y}) = \mathcal{N}\left(\mathbf{y}^* \mid \mu_{\mathbf{y}^*|\mathbf{y}}, \mathbf{\Sigma}_{\mathbf{y}^*|\mathbf{y}}\right)$$

where

$$\mu_{\mathbf{y}^*|\mathbf{y}} = \mathbf{\Phi}^*\mu + \mathbf{K}_{\mathbf{X}^*\mathbf{X}}\hat{\mathbf{K}}_{\mathbf{X}\mathbf{X}}^{-1}\left(\mathbf{y} - \mathbf{\Phi}\mu\right)$$

$$\mathbf{\Sigma}_{\mathbf{y}^*|\mathbf{y}} = \mathbf{K}_{\mathbf{X}^*\mathbf{X}^*} - \mathbf{K}_{\mathbf{X}^*\mathbf{X}}\hat{\mathbf{K}}_{\mathbf{X}\mathbf{X}}^{-1}\mathbf{K}_{\mathbf{X}\mathbf{X}^*}$$

Acquisition functions

There are many selection strategies that utilize the posterior model to select the next query point:

Probability of Improvement (PI): it aims to maximize the probability of finding a point that is better than the current best solution.

$$\mathrm{PI}(x) = P(f(x) \geq f(x_{\mathrm{best}}))$$

Acquisition functions

Expected Improvement (EI): it is defined as the expected value of the improvement over the current best solution.

$$\mathrm{EI}(x) = \mathbb{E}\left[\max(f(x) - f(x_{\text{best}}), 0)\right]$$

Upper Confidence Bound (UCB): it is defined as the sum of the mean function value and a measure of the uncertainty in the function value.

$$\mathrm{UCB}(x) = \mu(x) + \kappa\sigma(x)$$

Hyperparameter tuning: Maximum Likelihood Estimation

Maximizing the likelihood of the kernel parameters given the data can give an estimate on the best parameters to choose for the kernel:

$$\hat{\theta} = \arg\max_{\theta} \log p(\mathbf{y} \mid \mathbf{X}, \theta)$$

The typical estimation of the hyperparameters by maximizing the marginal likelihood can easily fall into traps. The optimization problem was tackled by **minimizing the negative log-likelihood**:

$$\log p(\mathbf{y} \mid \mathbf{X}, \theta) = -\frac{1}{2} \log |\mathbf{K}(\theta) + \sigma^2 \mathbf{I}| - \frac{1}{2} \mathbf{y}^T (\mathbf{K}(\theta) + \sigma^2 \mathbf{I})^{-1} \mathbf{y} - \frac{n}{2} \log 2\pi$$

Hyperparameter tuning: gradient descent

Minimize the negative log-likelihood using **gradient descent** is another approach to obtain an estimate of the best hyperparameters to use:

$$\theta_l^{(t+1)} = \theta_l^{(t)} - \lambda \nabla \ell \left(\boldsymbol{\theta}; \mathbf{X}_n, \mathbf{y}_n \right)_l$$

Each gradient component of the negative log-likelihood can be computed with the following formula:

$$\nabla \ell \left(\boldsymbol{\theta}; \mathbf{X}_n, \mathbf{y}_n \right)_l = \frac{1}{2n} \left[-\mathbf{y}_n^\top \mathbf{K}_n^{-1} \frac{\partial \mathbf{K}_n}{\partial \theta_l} \mathbf{K}_n^{-1} \mathbf{y}_n + \operatorname{tr} \left(\mathbf{K}_n^{-1} \frac{\partial \mathbf{K}_n}{\partial \theta_l} \right) \right]$$

Hyperparameter tuning: marginal acquisition function

Point estimation methods like MLE provide a single estimate but do not account for the uncertainty in the estimate.

Marginalization provides a distribution over the hyperparameters that consider this uncertainty:

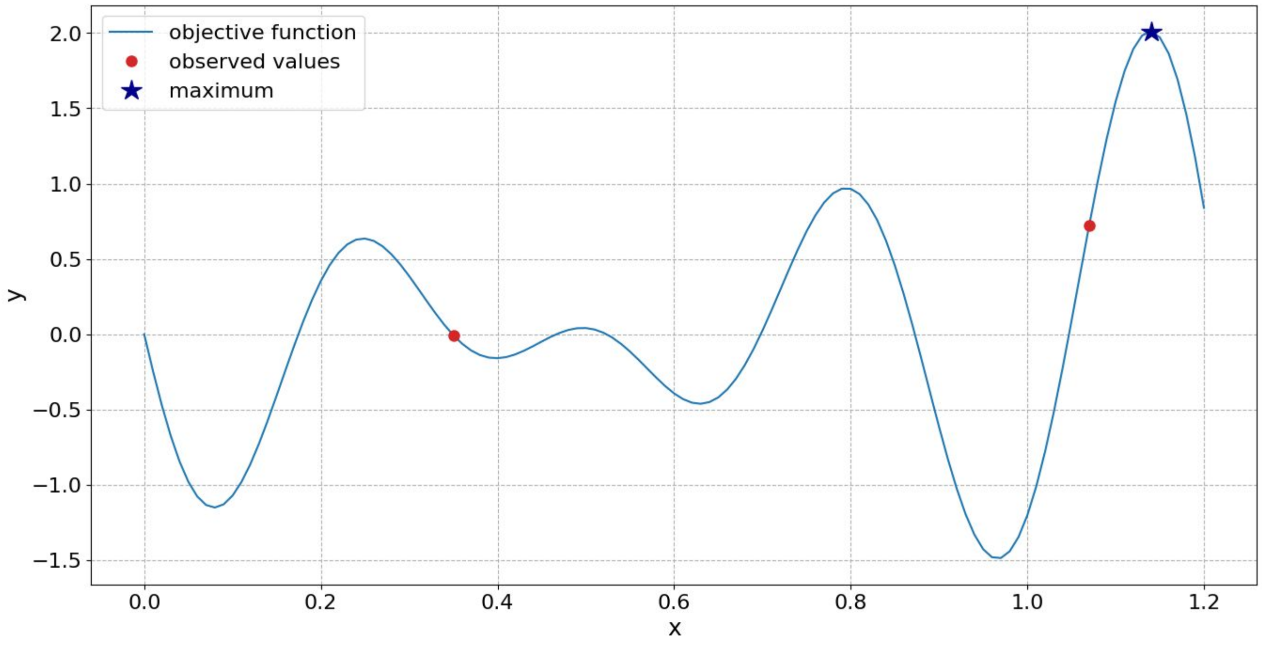
$$\alpha(x) = \mathbb{E}_{\theta|\mathcal{D}_n}[\alpha(x;\theta)] \approx \frac{1}{M}\sum_{i=1}^{M} \alpha\left(x; \theta_n^{(i)}\right)$$

The sampling procedure is not trivial given that it requires to tune the Metropolis-Hasting hyperparameters.

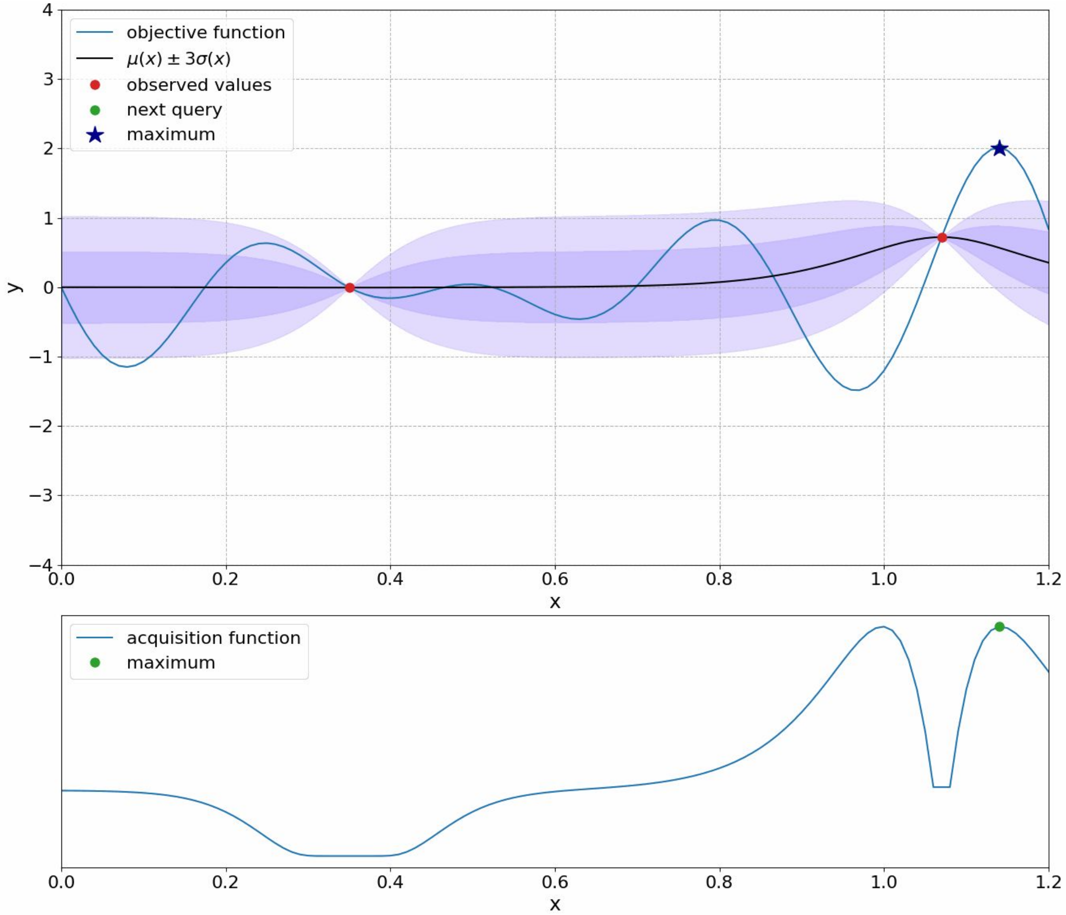
1D analytic function

$$y = -(a - bx)\sin(cx)$$

$$[a = 1.3, b = 3, c = 18]$$

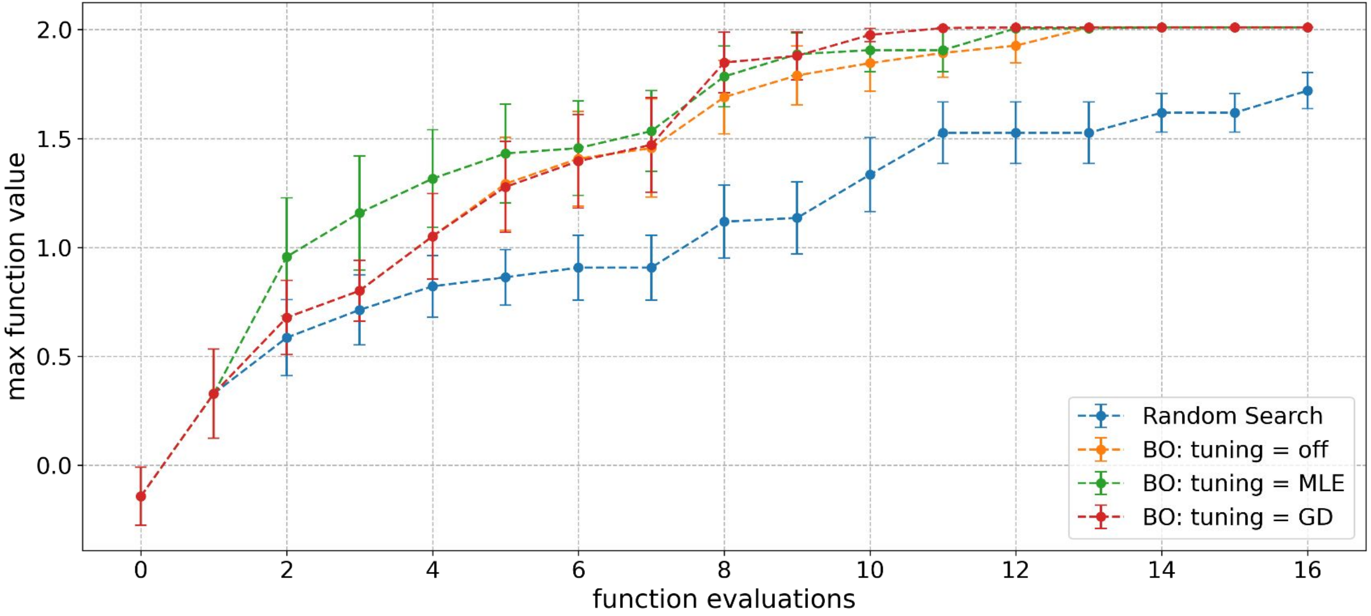


1D analytic function: simulation



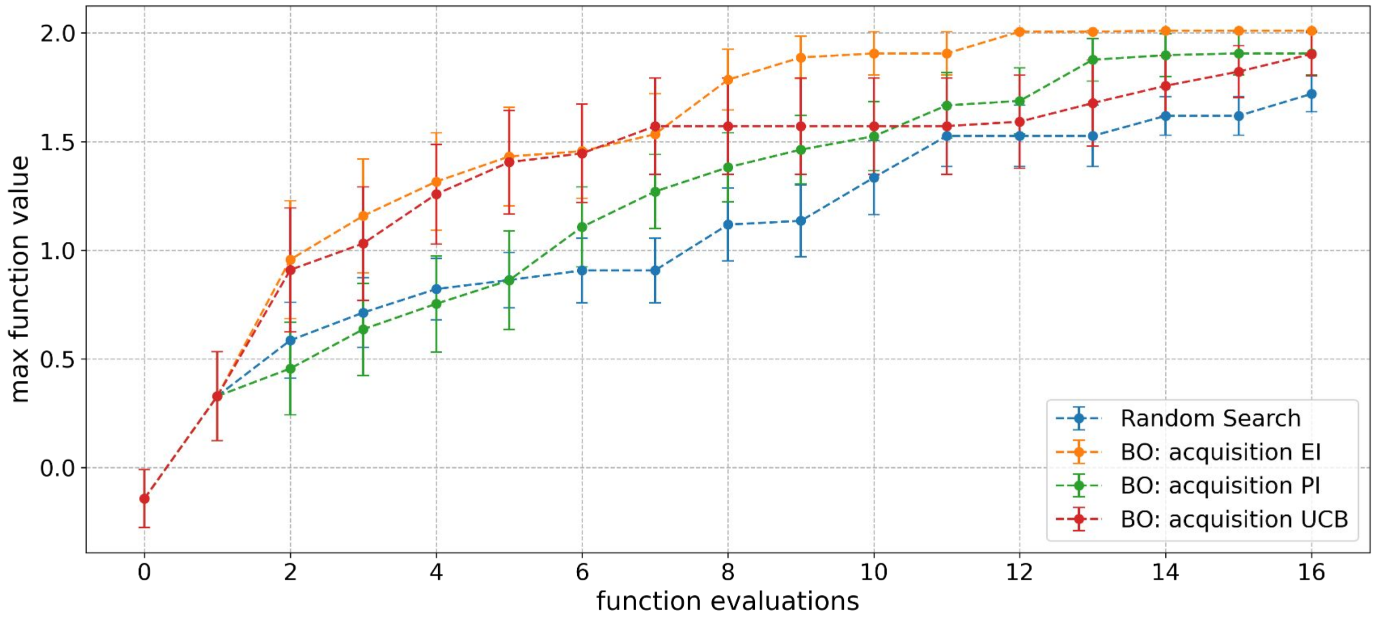
1D analytic function: hyperparameter tuning

Comparison of hyperparameter tuning methods:



1D analytic function: acquisition functions

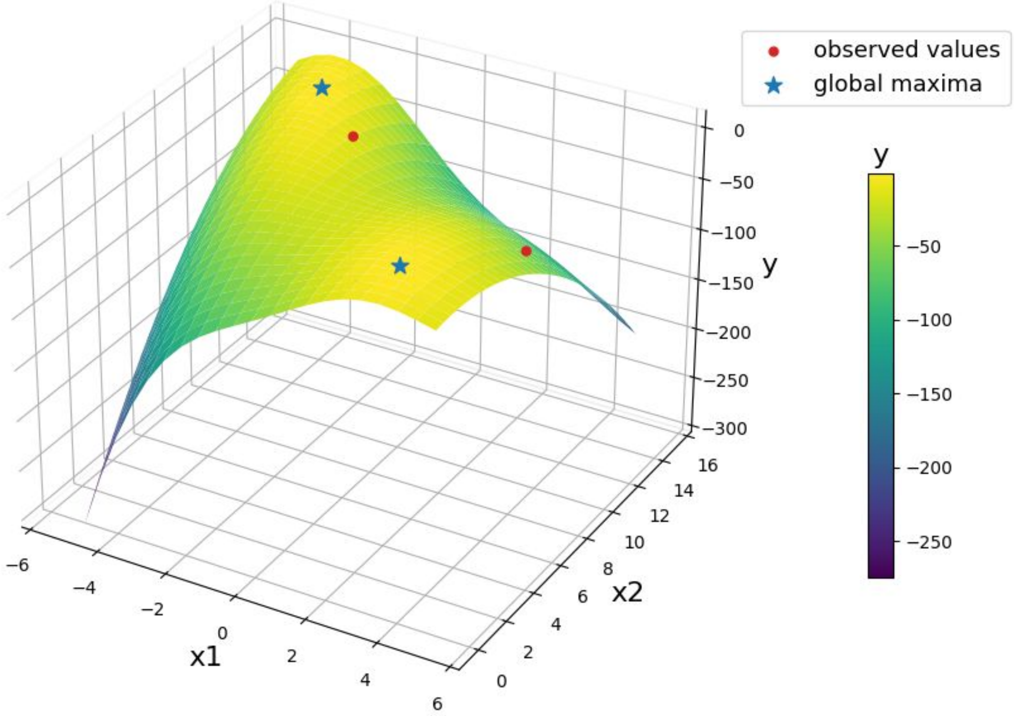
Comparison of acquisition functions:



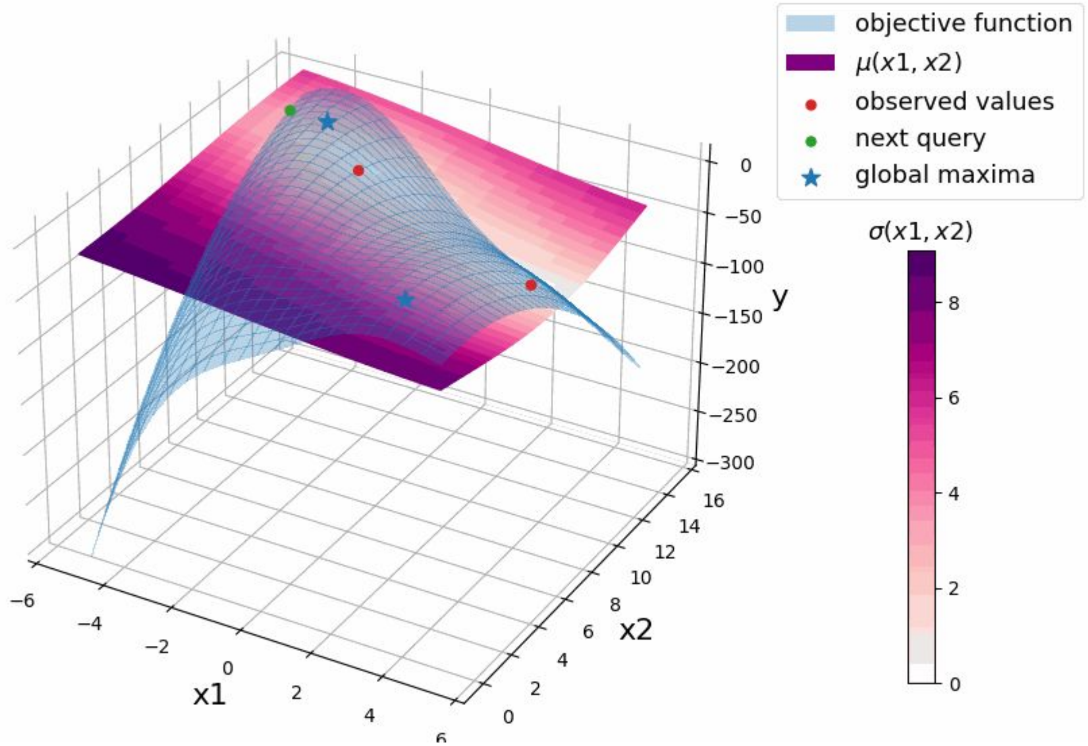
2D analytic function : Branin-Hoo

$$f(x_1, x_2) = -a\left(x_2 - bx_1^2 + cx_1 - r\right)^2 - s(1-t)\cos\left(x_1\right) - s$$

$$[a = 1, b = 5.1/(4\pi 2), c = 5/\pi, r = 6, s = 10, t = 1/(8\pi)]$$

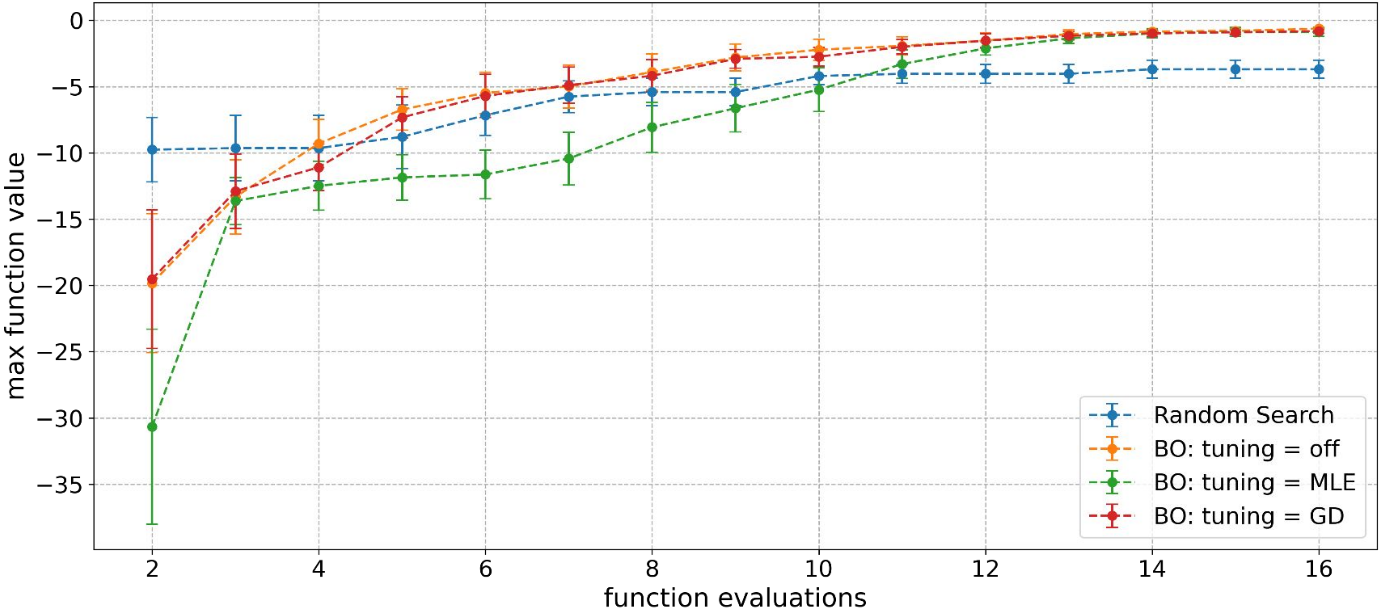


2D analytic function: simulation



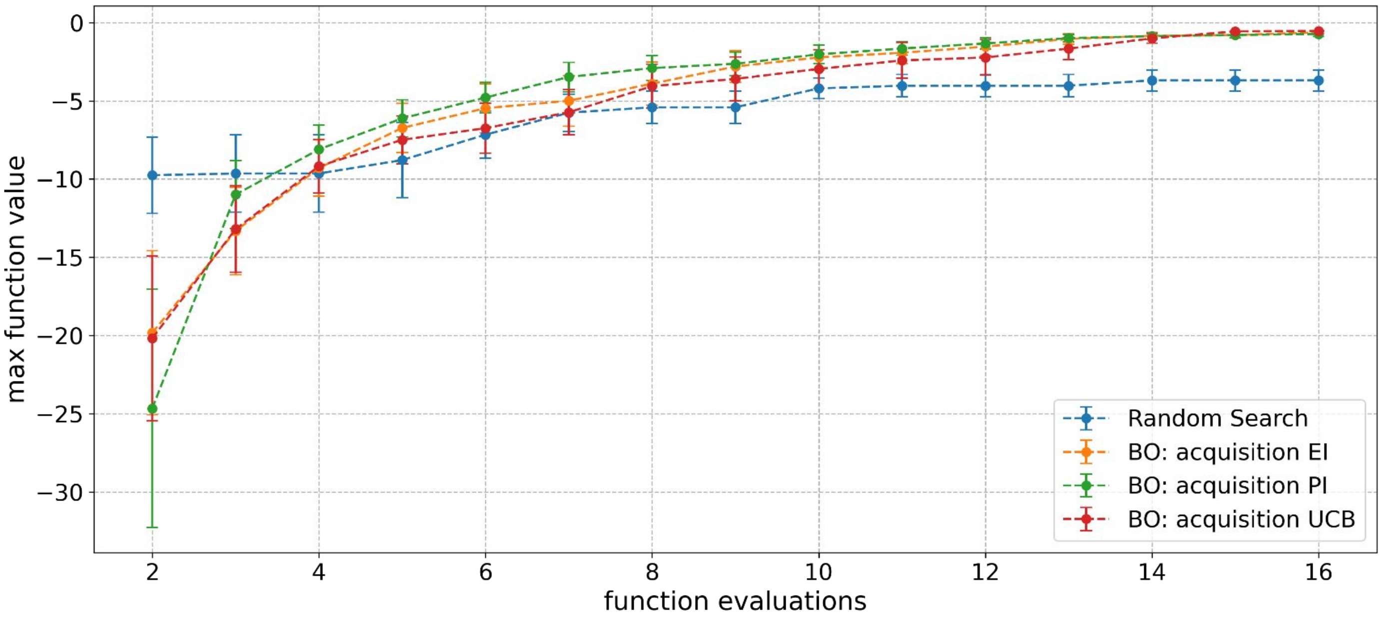
2D analytic function: hyperparameter tuning

Comparison of hyperparameter tuning methods:



2D analytic function: acquisition functions

Comparison of acquisition functions:



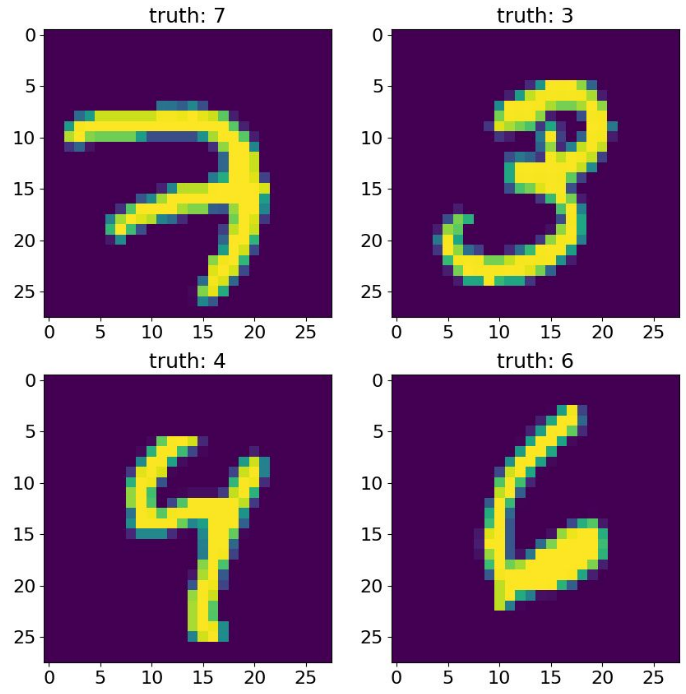
Multilayer perceptron

We trained a simple fully connected feedforward **neural network** (NN) on the *mnist-784* dataset from OpenML.

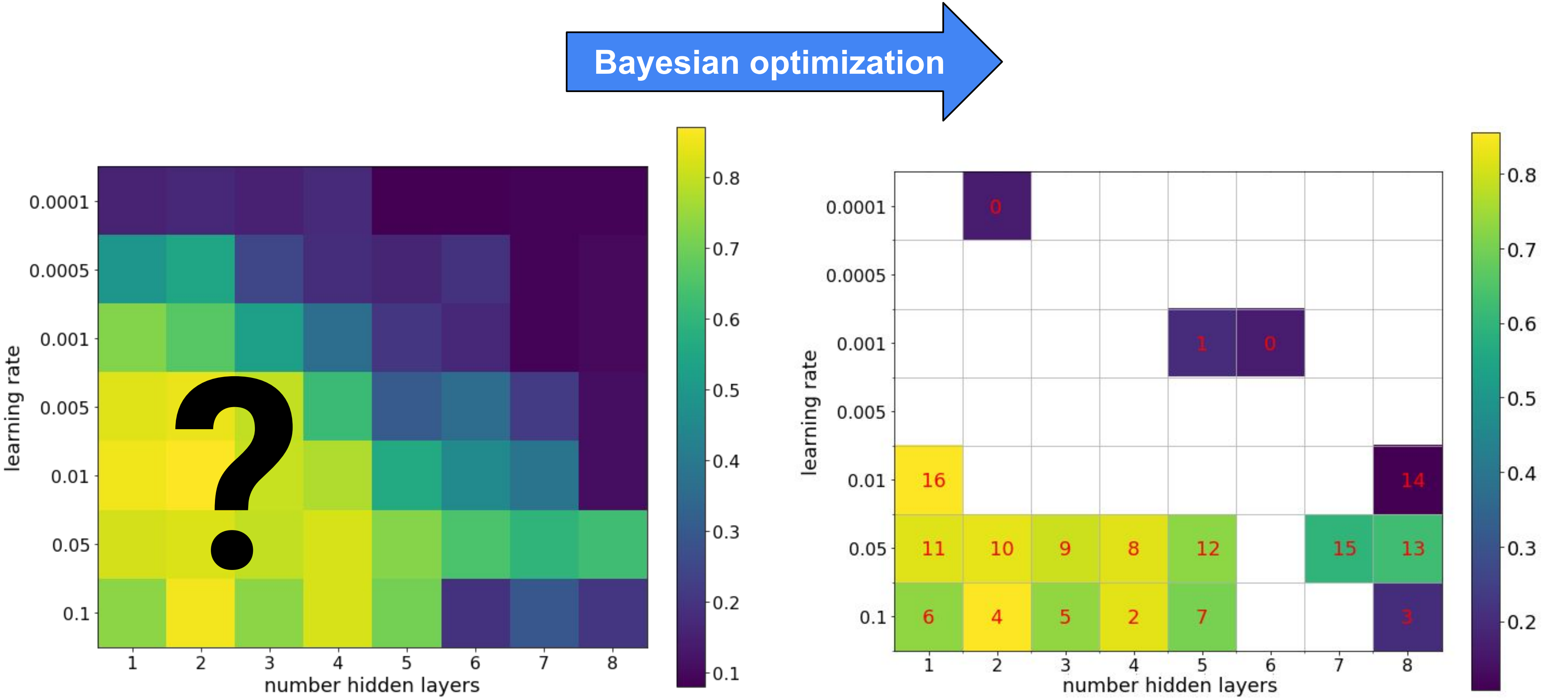
The number of neurons in each hidden layer is set to 5 and we look for the maximum scores obtained by the NN, exploring two hyperparameters:

- number of hidden layers
- learning rate

mnist-784 dataset



Multilayer perceptron: simulation



Multilayer perceptron: hyperparameter tuning

Comparison of hyperparameter tuning methods: