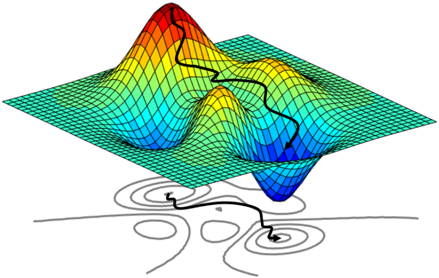
Optimization Techniques for Big Data Analysis

Chapter 9. Introduction to Bayesian Optimisation

Master of Science in Signal Theory and Communications
Dpto. de Señales, Sistemas y Radiocomunicaciones
E.T.S. Ingenieros de Telecomunicación
Universidad Politécnica de Madrid

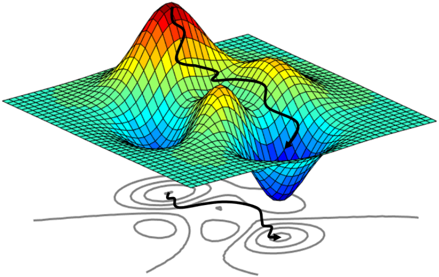
2023

What is mathematical optimization?



“The selection of the best element, with regard to some criterion, from some set of available alternatives” [3].

What is mathematical optimization?



"The selection of the best element, with regard to some criterion, from some set of available alternatives" [3].

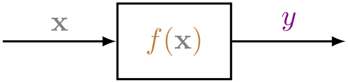
So typically we are given a problem like the following:

$$\hat{z} = \arg\max_{z} f(z) \ \text{ s.t. } g(z) < a \tag{1}$$

- $f(\cdot)$ function subject to optimization.
- $z \in \mathcal{Z}$ variables/parameters that need to be adjusted.
- $\mathcal{Z}$ is the search space. $\hat{z}$ is the optimum.
- $g(\cdot)$ restrictions. $f|_{\mathcal{G}}$, where $\mathcal{G} \subseteq \mathcal{Z}$; $\mathcal{G}$ is the feasible set.

ML set-up

A typical ML task can be seen like:



where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$; so we want to find a model $f(\cdot)$ that performs the mapping $f : \mathcal{X} \to \mathbb{R}$,

ML set-up

A typical ML task can be seen like:

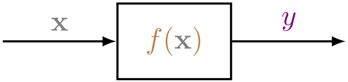
$$\xrightarrow{\mathbf{x}} \boxed{f(\mathbf{x})} \xrightarrow{y}$$

where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$; so we want to find a model $f(\cdot)$ that performs the mapping $f : \mathcal{X} \to \mathbb{R}$, and often we assume that:

$$y = f_{\theta}(\mathbf{x}) + \varepsilon; \;\; \varepsilon \sim \mathcal{N}(0, \sigma^2) \tag{2}$$

ML set-up

A typical ML task can be seen like:



where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$; so we want to find a model $f(\cdot)$ that performs the mapping $f : \mathcal{X} \to \mathbb{R}$, and often we assume that:

$$y = f_{\theta}(\mathbf{x}) + \varepsilon; \;\; \varepsilon \sim \mathcal{N}(0, \sigma^2) \tag{2}$$

Using a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)_{i=1}^N\}$ and some criterion $J(\theta)$ we approximate $f(\cdot)$. This allow us to make predictions of y^* given a new $\mathbf{x}^*$.

ML set-up

A typical ML task can be seen like:

$$\mathbf{x} \longrightarrow \boxed{f(\mathbf{x})} \overset{y}{\longrightarrow}$$

where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$; so we want to find a model $f(\cdot)$ that performs the mapping $f : \mathcal{X} \to \mathbb{R}$, and often we assume that:

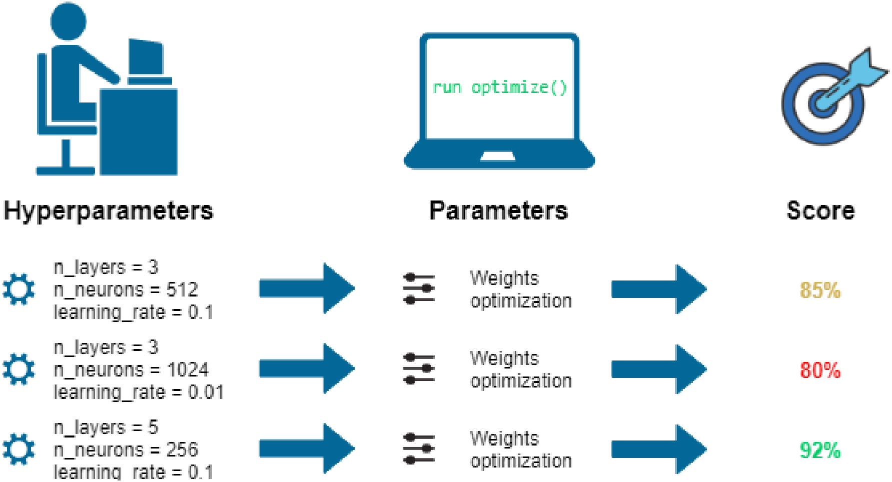
$$y = f_\theta(\mathbf{x}) + \varepsilon; \;\; \varepsilon \sim \mathcal{N}(0, \sigma^2) \tag{2}$$

Using a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)_{i=1}^N\}$ and some criterion $J(\theta)$ we approximate $f(\cdot)$. This allow us to make predictions of y^* given a new $\mathbf{x}^*$.

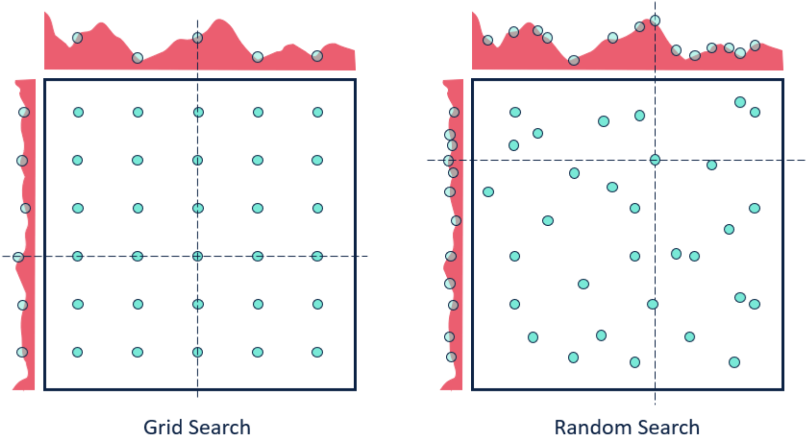
Training $f_\theta(\cdot)$ corresponds to the optimization of $J(\theta)$!

What about the hyperparameters?

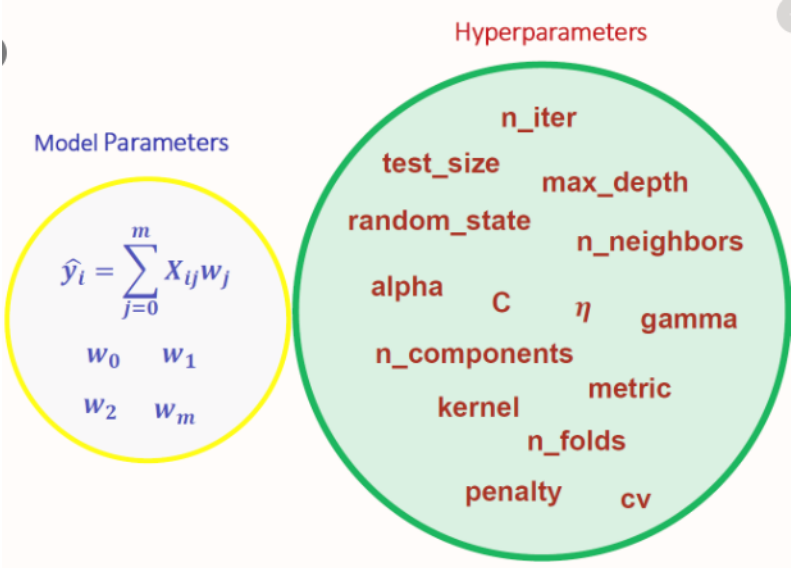
What about the hyperparameters?



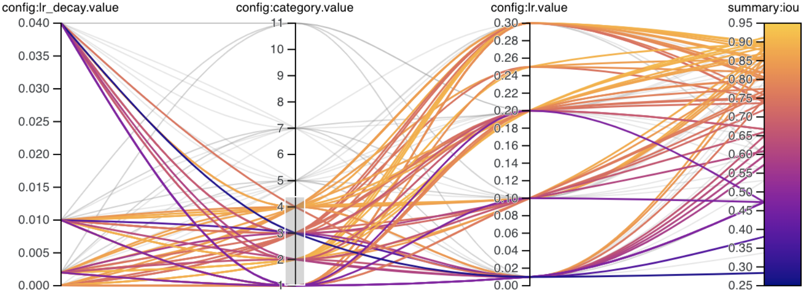
What about the hyperparameters?



What about the hyperparameters?



What about the hyperparameters?



Hyperparameters tuning as optimization problem

Why is it difficult to tune hyperparameters?

Hyperparameters tuning as optimization problem

Why is it difficult to tune hyperparameters?

- **Evaluation cost**: Evaluating the function that we wish to maximize (i.e., the network performance) in hyperparameter search is very expensive.

Hyperparameters tuning as optimization problem

Why is it difficult to tune hyperparameters?

- **Evaluation cost**: Evaluating the function that we wish to maximize (i.e., the network performance) in hyperparameter search is very expensive.
- **Multiple local optima**: The function is not convex.

Hyperparameters tuning as optimization problem

Why is it difficult to tune hyperparameters?

- **Evaluation cost**: Evaluating the function that we wish to maximize (i.e., the network performance) in hyperparameter search is very expensive.
- **Multiple local optima**: The function is not convex.
- **No derivatives**: We do not have access to the function's gradient with respect to the hyperparameters.

Hyperparameters tuning as optimization problem

Why is it difficult to tune hyperparameters?

- **Evaluation cost**: Evaluating the function that we wish to maximize (i.e., the network performance) in hyperparameter search is very expensive.
- **Multiple local optima**: The function is not convex.
- **No derivatives**: We do not have access to the function's gradient with respect to the hyperparameters.
- **Variable types**: There are a mixture of discrete variables and continuous variables.

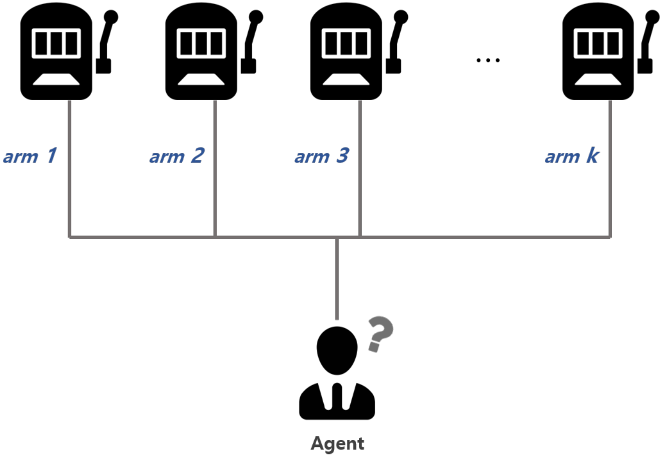
Hyperparameters tuning as optimization problem

Why is it difficult to tune hyperparameters?

- **Evaluation cost**: Evaluating the function that we wish to maximize (i.e., the network performance) in hyperparameter search is very expensive.
- **Multiple local optima**: The function is not convex.
- **No derivatives**: We do not have access to the function's gradient with respect to the hyperparameters.
- **Variable types**: There are a mixture of discrete variables and continuous variables.
- **Noise**: The function may return different values for the same input hyperparameter set.

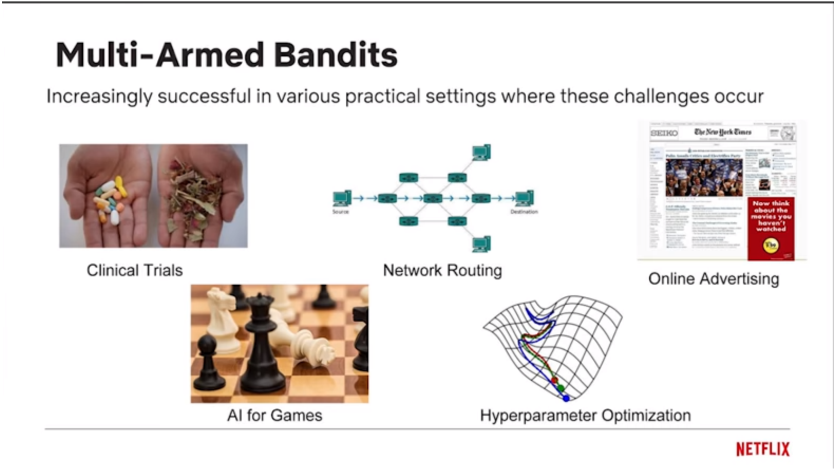
A typical case

Take decisions under uncertainty maximizing rewards.

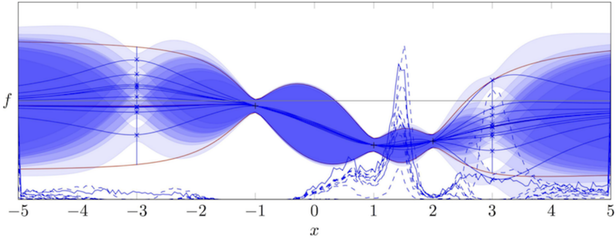


A typical case

Take decisions under uncertainty maximizing rewards.

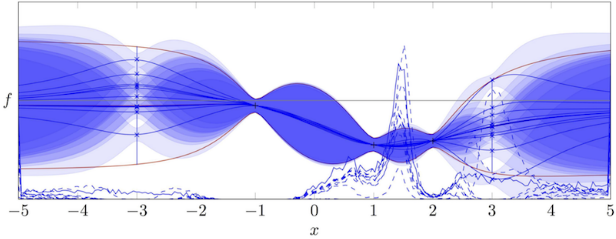


One alternative



Bayesian optimization is a framework that can deal with optimization problems with all the listed challenges.

One alternative



Bayesian optimization is a framework that can deal with optimization problems with all the listed challenges.

BO builds a cheaper **surrogate model** for the true objective; it includes both our current estimate of that function and the **uncertainty** around that estimate. By considering this model, we can choose where next to sample the function.

Active learning

Let's consider the following example:

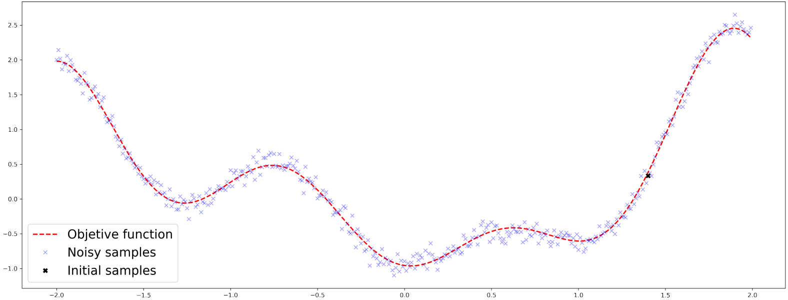


Figure: Example of active learning problem

What locations should $f(\mathbf{x})$ be sampled at in order to approximate it properly?

Active learning

Given a ML problem with a small set of labelled samples and a large set of unlabelled samples, how to guide the labelling process that is usually very time-consuming to enrich the training dataset gradually?

Active learning

Given a ML problem with a small set of labelled samples and a large set of unlabelled samples, how to guide the labelling process that is usually very time-consuming to enrich the training dataset gradually?

The most common criterion for input space exploring in active learning is **uncertainty reduction**.

$$\mathbf{x}_s = \arg\max_{\mathbf{x}\in\mathcal{X}} \mathbb{V}[f_\theta(\mathbf{x})] \tag{3}$$

Active learning

Given a ML problem with a small set of labelled samples and a large set of unlabelled samples, how to guide the labelling process that is usually very time-consuming to enrich the training dataset gradually?

The most common criterion for input space exploring in active learning is **uncertainty reduction**.

$$\mathbf{x}_s = \arg\max_{\mathbf{x}\in\mathcal{X}} \mathbb{V}[f_\theta(\mathbf{x})] \tag{3}$$

The former criterion requires $f_\theta(\cdot)$ to provide not only a prediction value y but an uncertanty estimation over such prediction, let's call it $\sigma(y)$.

Bayesian models

Most common ML approaches follow a frequentist principle, i.e. the learning criteria adjust the model parameters by maximizing the likelihood function (e.g. MSE, cross-entropy).

Bayesian models

Most common ML approaches follow a frequentist principle, i.e. the learning criteria adjust the model parameters by maximizing the likelihood function (e.g. MSE, cross-entropy).

In contrast to maximum likelihood learning, Bayesian learning explicitly models uncertainty over both the observed variables $\mathbf{x}$ and the parameters θ. In other words, the parameters θ are random variables as well.

A prior distribution over the parameters, $p(\theta)$ encodes our initial beliefs.

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \tag{4}$$

Bayesian models

Most common ML approaches follow a frequentist principle, i.e. the learning criteria adjust the model parameters by maximizing the likelihood function (e.g. MSE, cross-entropy).

In contrast to maximum likelihood learning, Bayesian learning explicitly models uncertainty over both the observed variables $\mathbf{x}$ and the parameters θ. In other words, the parameters θ are random variables as well.

A prior distribution over the parameters, $p(\theta)$ encodes our initial beliefs.

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int p(\mathcal{D}|\theta)p(\theta)d\theta} \tag{4}$$

Bayesian models...

If θ is high dimensional then computing integrals could be quite challenging, so when possible $p(\theta)$ is chosen to be a conjugate distribution of $p(\mathcal{D}|\theta)$.

The predictive distribution can be written in the form:

$$p(y^*|\mathbf{x}^*, \mathcal{D}) = \int p(y^*|\mathbf{x}^*, \theta)p(\theta|\mathcal{D})d\theta \tag{5}$$

Thus, we end up with a posterior distribution instead of a point estimate. The most probable output for an input $\mathbf{x}^*$ will be the expected value of the distribution, i.e. its mean, and its variance provides an **uncertainty measure** about the prediction.

Gaussian Processes

A Gaussian Processes (GPs) is defined as a probability distribution over functions $f(\mathbf{x})$ such that the set of values of $f(\mathbf{x})$ evaluated at an arbitrary set of points $\mathbf{x}_1, ..., \mathbf{x}_N$ jointly have a Gaussian distribution [1].

Gaussian Processes

A Gaussian Processes (GPs) is defined as a probability distribution over functions $f(\mathbf{x})$ such that the set of values of $f(\mathbf{x})$ evaluated at an arbitrary set of points $\mathbf{x}_1, ..., \mathbf{x}_N$ jointly have a Gaussian distribution [1].

Since we assume that the observations are jointly Gaussian:

> **Partitioned Gaussians**
>
> Given a joint Gaussian distribution $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with $\boldsymbol{\Lambda} \equiv \boldsymbol{\Sigma}^{-1}$ and
>
> $$\mathbf{x} = \begin{pmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{pmatrix}, \quad \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{pmatrix}$$

Gaussian Processes...

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{pmatrix}, \quad \boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{pmatrix}.$$

Conditional distribution:

$$\begin{aligned} p(\mathbf{x}_a|\mathbf{x}_b) &= \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{a|b}, \boldsymbol{\Lambda}_{aa}^{-1}) \\ \boldsymbol{\mu}_{a|b} &= \boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{aa}^{-1}\boldsymbol{\Lambda}_{ab}(\mathbf{x}_b - \boldsymbol{\mu}_b). \end{aligned}$$

Marginal distribution:

$$p(\mathbf{x}_a) = \mathcal{N}(\mathbf{x}_a|\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_{aa}).$$

A GP is completely specified by its mean function and covariance function [5].

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \end{aligned}$$

Gaussian Process...

The GP is written as:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{6}$$

Assuming there is noise in the measurements, the joint distribution for new samples $\mathbf{X}^*$:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(\mathbf{X}, \mathbf{X}) + \sigma^2\mathbf{I} & K(\mathbf{X}, \mathbf{X}^*) \\ K(\mathbf{X}^*, \mathbf{X}) & K(\mathbf{X}^*, \mathbf{X}^*) \end{bmatrix}\right) \tag{7}$$

So, the conditional distribution $\mathbf{f}^*|\mathbf{X}, \mathbf{y}, \mathbf{X}^*$ is straightforward [5]:

$$\begin{aligned} \bar{\mathbf{f}^*} &= K(\mathbf{X}^*, \mathbf{X})[K(\mathbf{X}, \mathbf{X}) + \sigma^2\mathbf{I}]^{-1}\mathbf{y} \qquad (8) \\ \mathrm{cov}(\mathbf{f}^*) &= K(\mathbf{X}^*, \mathbf{X}^*) - K(\mathbf{X}^*, \mathbf{X})[K(\mathbf{X}, \mathbf{X}) + \sigma^2\mathbf{I}]^{-1}K(\mathbf{X}, \mathbf{X}^*) \end{aligned}$$

Gaussian Process hyperparameter learning

The hyperparameters of the model can be estimated by maximizing the marginal likelihood:

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{f}, X)p(\mathbf{f}|X)d\mathbf{f} \tag{9}$$

Under the GP model the prior is Gaussian, $\mathbf{f}|X \sim \mathcal{N}(0, K)$ and the likelihood is a factorized Gaussian $\mathbf{y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}|\sigma^2\mathbf{I})$, therefore:

$$\log p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\mathbf{y}^T(K+\sigma^2\mathbf{I})\mathbf{y} - \frac{1}{2}\log|K+\sigma^2\mathbf{I}| - \frac{N}{2}\log 2\pi \tag{10}$$

Gaussian Process Regression examples

The prior:

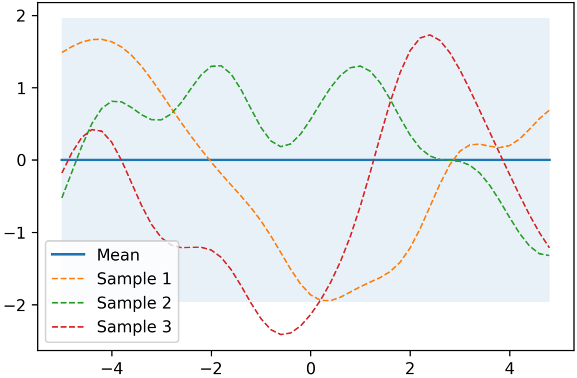


Figure: Samples from the $\mathcal{GP}$

Gaussian Process Regression examples

The prior:

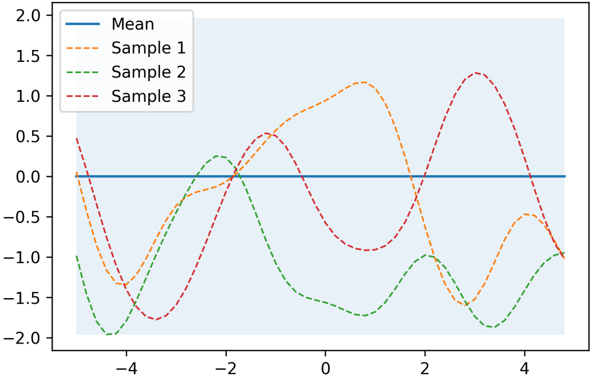


Figure: Samples from the $\mathcal{GP}$

Gaussian Process Regression examples

The prior:

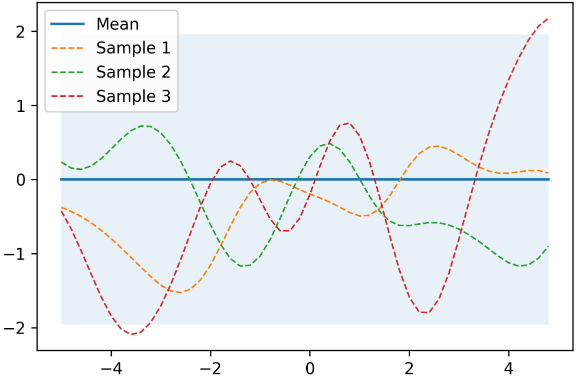


Figure: Samples from the $\mathcal{GP}$

Gaussian Process Regression examples

The posterior predictive distribution (noise free):

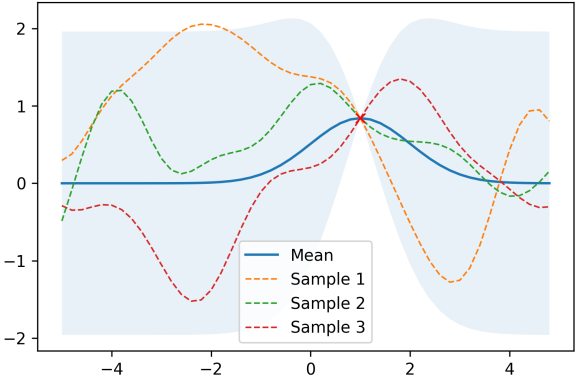


Figure: Predictive distribution of the $\mathcal{GP}$ regressor

Gaussian Process Regression examples

The posterior predictive distribution (noise free):

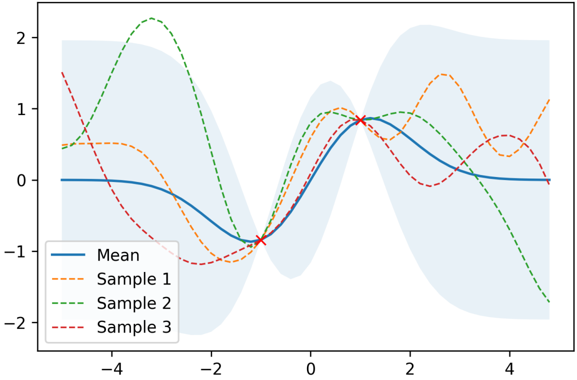


Figure: Predictive distribution of the $\mathcal{GP}$ regressor

Gaussian Process Regression examples

The posterior predictive distribution (noise free):

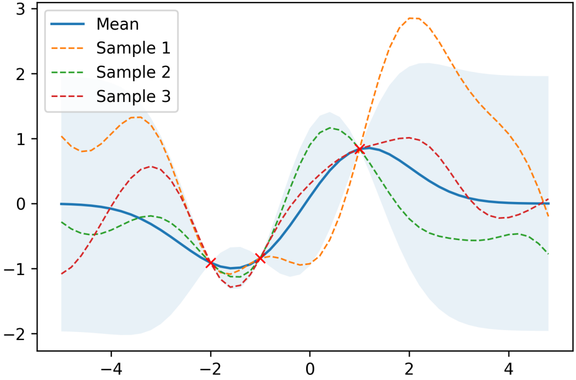


Figure: Predictive distribution of the $\mathcal{GP}$ regressor

Gaussian Process Regression examples

The posterior predictive distribution (noise free):

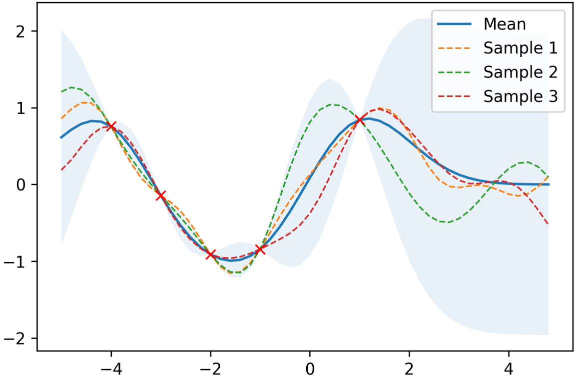


Figure: Predictive distribution of the $\mathcal{GP}$ regressor

Gaussian Process Regression examples

The posterior predictive distribution (with noise):

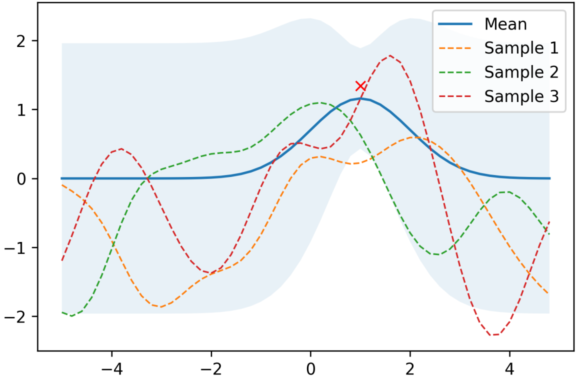


Figure: Predictive distribution of the $\mathcal{GP}$ regressor

Gaussian Process Regression examples

The posterior predictive distribution (with noise):

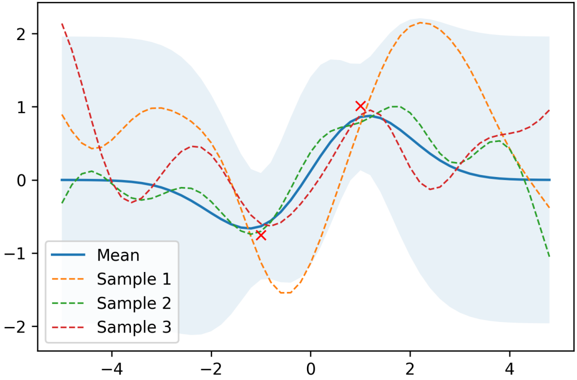


Figure: Predictive distribution of the $\mathcal{GP}$ regressor

Gaussian Process Regression examples

The posterior predictive distribution (with noise):

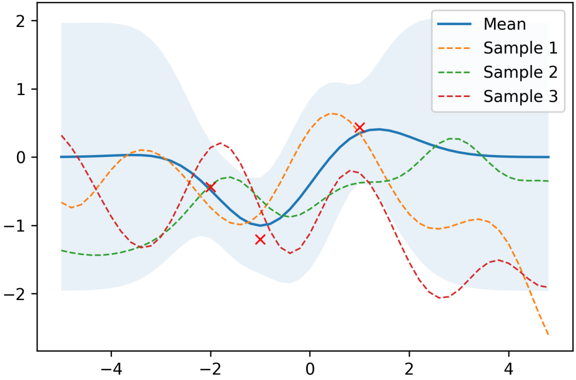


Figure: Predictive distribution of the $\mathcal{GP}$ regressor

Gaussian Process Regression examples

The posterior predictive distribution (with noise):

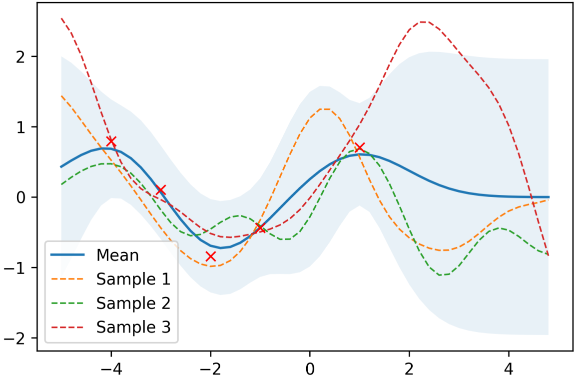


Figure: Predictive distribution of the $\mathcal{GP}$ regressor

Effect of Kernel hyperparameters

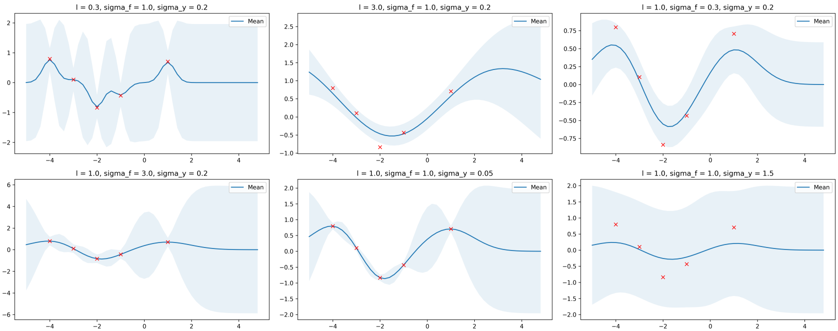


Figure: kernel effects in the posterior distribution
$k(\mathbf{x}_i, \mathbf{x}_j) = {\sigma_f}^2 \exp(-\frac{1}{2l^2}(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j))$.

Effect of Kernel hyperparameters

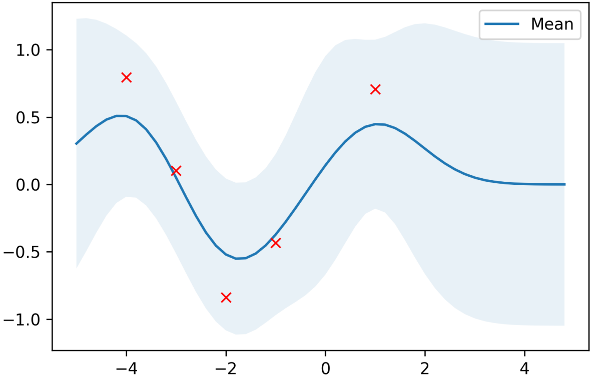


Figure: kernel effects in the posterior distribution
$k(\mathbf{x}_i, \mathbf{x}_j) = {\sigma_f}^2 \exp(-\frac{1}{2l^2}(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j))$.

Effect of Kernel hyperparameters in 2D

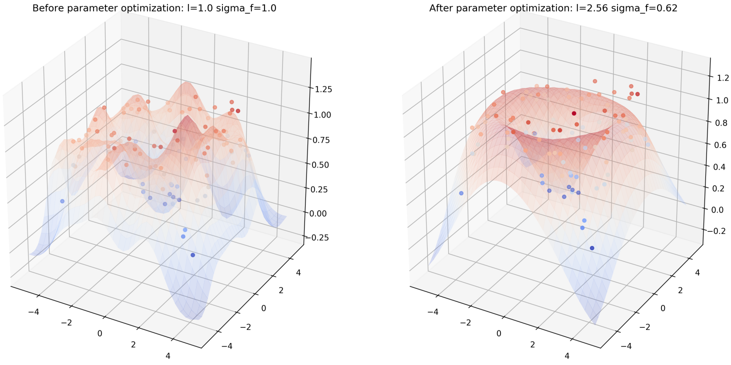


Figure: Mean function in higher dimension

Samples from different kernels

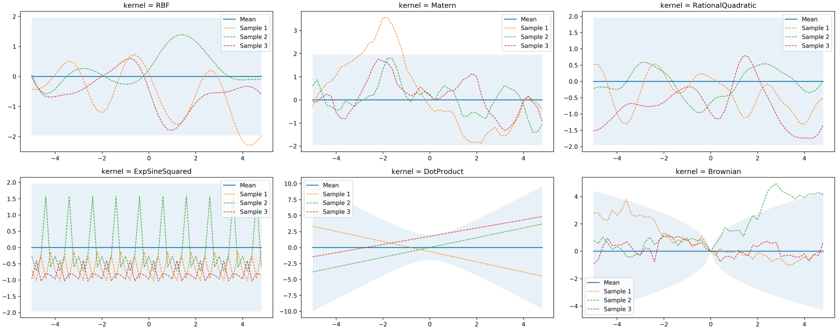


Figure: Samples from GPs with different covariances

Back to the Active learning problem

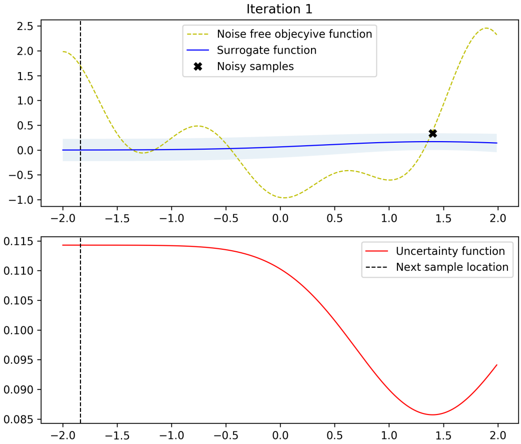


Figure: Active learning iterations

Back to the Active learning problem

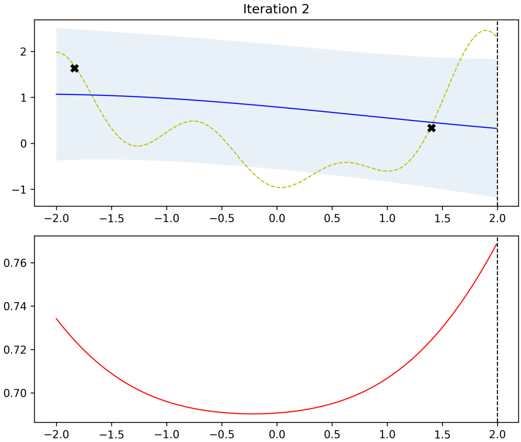


Figure: Active learning iterations

Back to the Active learning problem

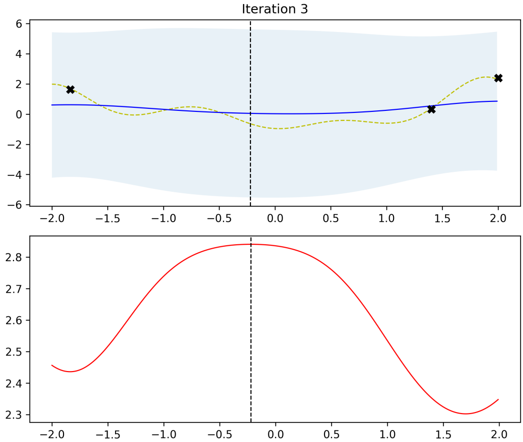


Figure: Active learning iterations

Back to the Active learning problem

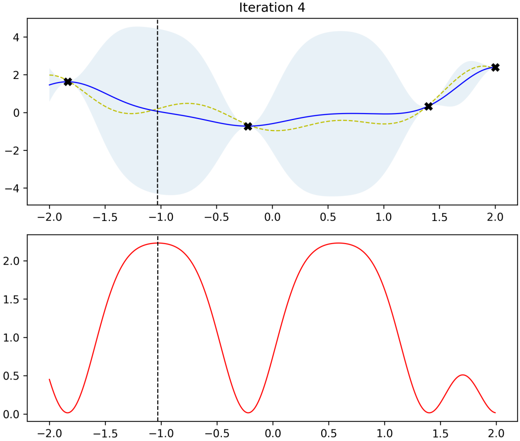


Figure: Active learning iterations

Back to the Active learning problem

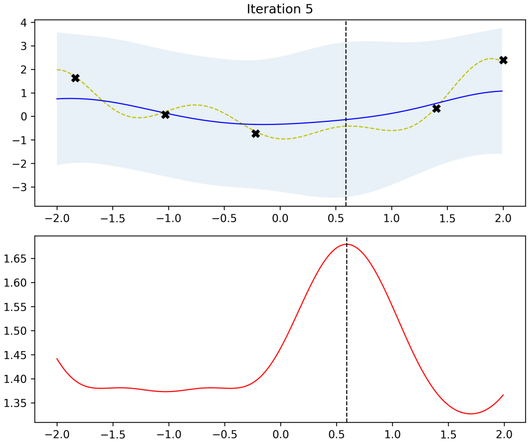


Figure: Active learning iterations

Back to the Active learning problem

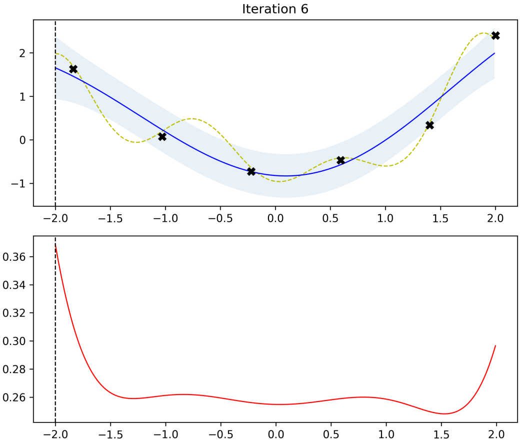


Figure: Active learning iterations

Back to the Active learning problem

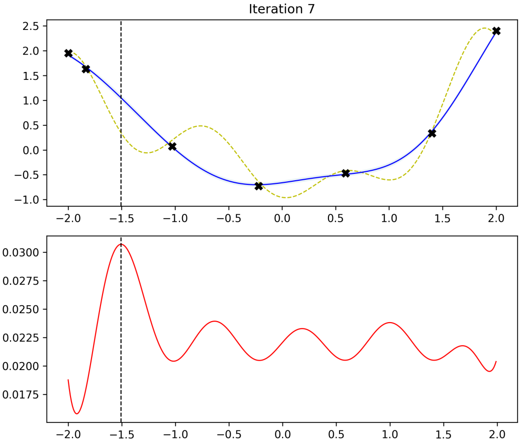


Figure: Active learning iterations

Back to the Active learning problem

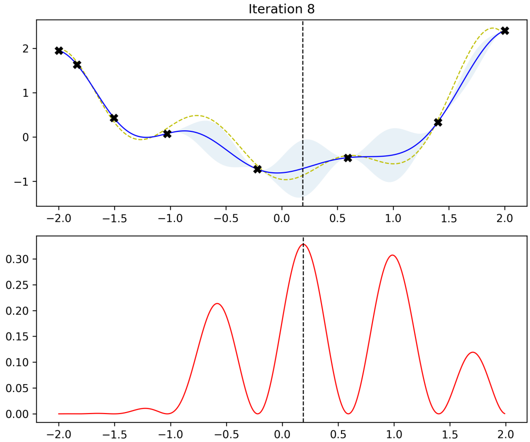


Figure: Active learning iterations

Back to the Active learning problem

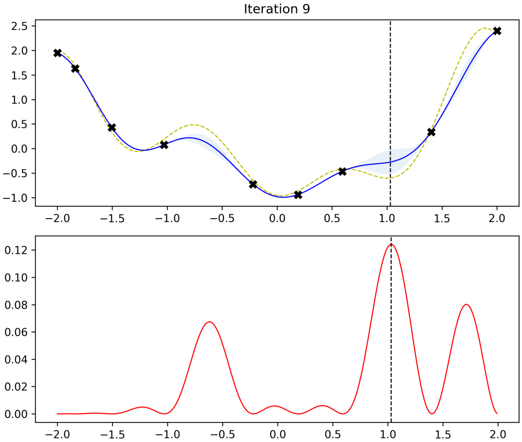


Figure: Active learning iterations

Back to the Active learning problem

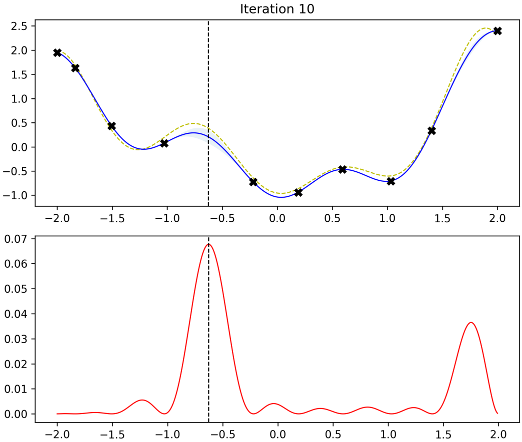


Figure: Active learning iterations

Back to the Active learning problem

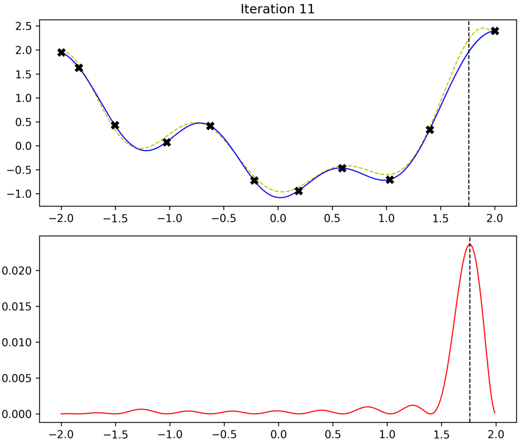


Figure: Active learning iterations

Bayesian optimization

If the function evaluation of $f(\cdot)$ is expensive, how to guide the search for the optimum in a minimum number of steps?

In essence, Bayesian optimization tackle a similar problem than active learning, however, optimization problems require a balance between **exploration** and **exploitation**.

Bayesian optimization

If the function evaluation of $f(\cdot)$ is expensive, how to guide the search for the optimum in a minimum number of steps?

In essence, Bayesian optimization tackle a similar problem than active learning, however, optimization problems require a balance between **exploration** and **exploitation**.

- Exploitation means sampling where the surrogate model predicts a high objective.
- Exploration means sampling at locations where the prediction uncertainty is high.

Bayesian optimization

If the function evaluation of $f(\cdot)$ is expensive, how to guide the search for the optimum in a minimum number of steps?

In essence, Bayesian optimization tackle a similar problem than active learning, however, optimization problems require a balance between **exploration** and **exploitation**.

- Exploitation means sampling where the surrogate model predicts a high objective.
- Exploration means sampling at locations where the prediction uncertainty is high.

In active learning, such a search was controlled only by uncertainty. In Bayesian optimization, instead, proposing sampling points in the search space is done by **acquisition functions**. They trade off exploitation and exploration.

Optimization algorithm

In order to find:

$$\mathbf{x}_{max} = \arg\max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}) \tag{11}$$

The Bayesian optimization procedure is as follows. For $t = 1, 2, \ldots$ repeat [2]:

1. Find the next sampling point $\mathbf{x}_t$ by optimizing the acquisition function over the GP: $\mathbf{x}_t = \text{argmax}_{\mathbf{x}} \alpha(\mathbf{x}; \mathcal{I}_{t-1})$

Optimization algorithm

In order to find:

$$\mathbf{x}_{max} = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \tag{11}$$

The Bayesian optimization procedure is as follows. For $t = 1, 2, ...$ repeat [2]:

1. Find the next sampling point $\mathbf{x}_t$ by optimizing the acquisition function over the GP: $\mathbf{x}_t = \text{argmax}_{\mathbf{x}} \alpha(\mathbf{x}; \mathcal{I}_{t-1})$
2. Obtain a possibly noisy sample $y_t = f(\mathbf{x}_t) + \varepsilon_t$ from the objective function f.

Optimization algorithm

In order to find:

$$\mathbf{x}_{max} = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \tag{11}$$

The Bayesian optimization procedure is as follows. For $t = 1, 2, ...$ repeat [2]:

1. Find the next sampling point $\mathbf{x}_t$ by optimizing the acquisition function over the GP: $\mathbf{x}_t = \text{argmax}_{\mathbf{x}} \alpha(\mathbf{x}; \mathcal{I}_{t-1})$
2. Obtain a possibly noisy sample $y_t = f(\mathbf{x}_t) + \varepsilon_t$ from the objective function f.
3. Add the sample to previous samples $\mathcal{D}_{1:t} = \{\mathcal{D}_{1:t-1}, (\mathbf{x}_t, y_t)\}$ and update the GP.

Optimization algorithm

In order to find:

$$\mathbf{x}_{max} = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \tag{11}$$

The Bayesian optimization procedure is as follows. For $t = 1, 2, ...$ repeat [2]:

1. Find the next sampling point $\mathbf{x}_t$ by optimizing the acquisition function over the GP: $\mathbf{x}_t = \text{argmax}_{\mathbf{x}} \alpha(\mathbf{x}; \mathcal{I}_{t-1})$
2. Obtain a possibly noisy sample $y_t = f(\mathbf{x}_t) + \varepsilon_t$ from the objective function f.
3. Add the sample to previous samples $\mathcal{D}_{1:t} = \{\mathcal{D}_{1:t-1}, (\mathbf{x}_t, y_t)\}$ and update the GP.

Optimization algorithm

In order to find:

$$\mathbf{x}_{max} = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \tag{11}$$

The Bayesian optimization procedure is as follows. For $t = 1, 2, \ldots$ repeat [2]:

1. Find the next sampling point $\mathbf{x}_t$ by optimizing the acquisition function over the GP: $\mathbf{x}_t = \text{argmax}_{\mathbf{x}} \alpha(\mathbf{x}; \mathcal{I}_{t-1})$
2. Obtain a possibly noisy sample $y_t = f(\mathbf{x}_t) + \varepsilon_t$ from the objective function f.
3. Add the sample to previous samples $\mathcal{D}_{1:t} = \{\mathcal{D}_{1:t-1}, (\mathbf{x}_t, y_t)\}$ and update the GP.

where $\mathcal{I}_{t-1}$ represents the available data set $\mathcal{D}_{1:t-1}$ and the GP structure (kernel, likelihood and parameter values) at $t-1$ step.

Acquisition functions

There are several proposed acquisition functions, a simple one is *probability of improvement*, which chooses the next query point as the one which has the highest probability of improvement over the current max $f(\mathbf{x}^+)$. Formally:

$$\mathbf{x}_t = \arg\max_{\mathbf{x}} P(f(\mathbf{x}) \geq (f(\mathbf{x}^+) + \epsilon)) \tag{12}$$

If we are using a GP as a surrogate the expression above converts to,

$$\mathbf{x}_t = \arg\max_{\mathbf{x}} \Phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \epsilon}{\sigma(\mathbf{x})}\right) \tag{13}$$

where $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ are the mean and variance of the posterior distribution at $\mathbf{x}$ respectively, and $\Phi(\cdot)$ indicates the CDF.

PI example

Let's consider the same objective function than before:

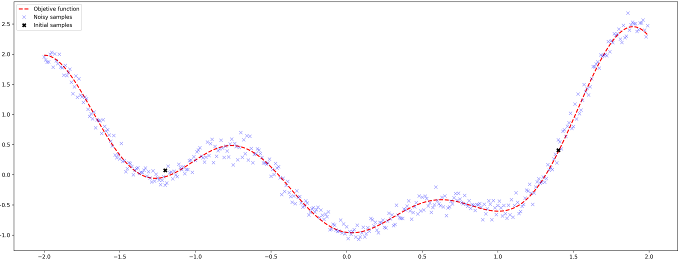


Figure: Example of a bayesian optimization problem

Take into account that in this case, the aim is to find the minimum of the function.

PI example...

$$\mathbf{x}_t = \arg\max_{\mathbf{x}} \Phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \epsilon}{\sigma(\mathbf{x})}\right)$$

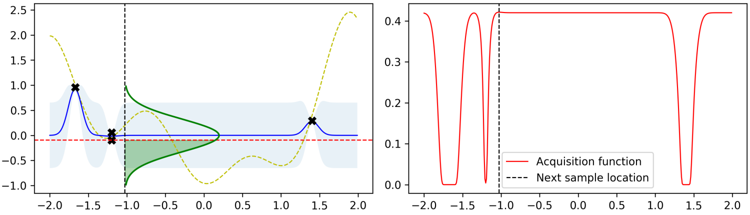


Figure: Interpretation of PI acquisition function

PI example...

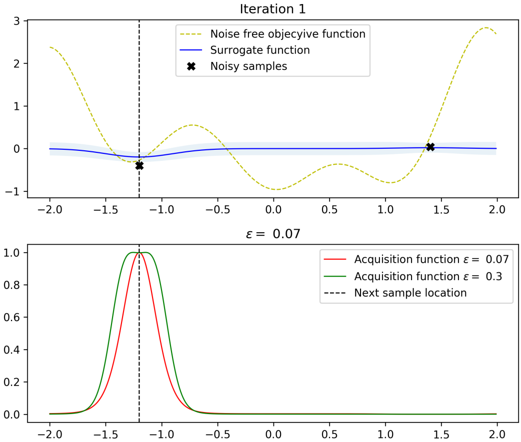


Figure: Bayesian optimization iterations using PI

PI example...

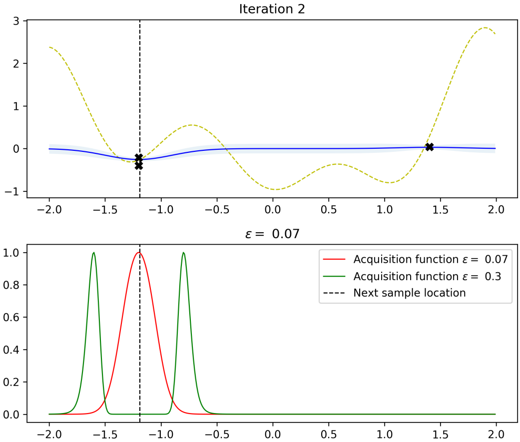


Figure: Bayesian optimization iterations using PI

PI example...

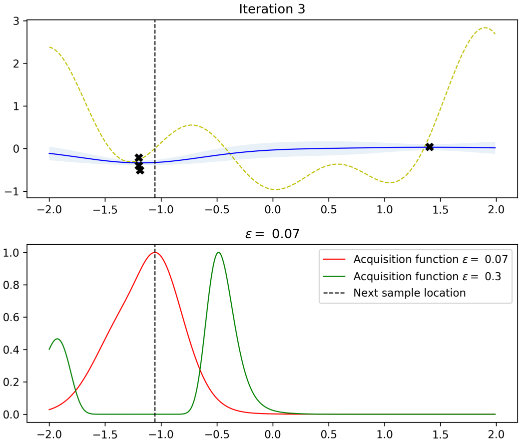


Figure: Bayesian optimization iterations using PI

PI example...

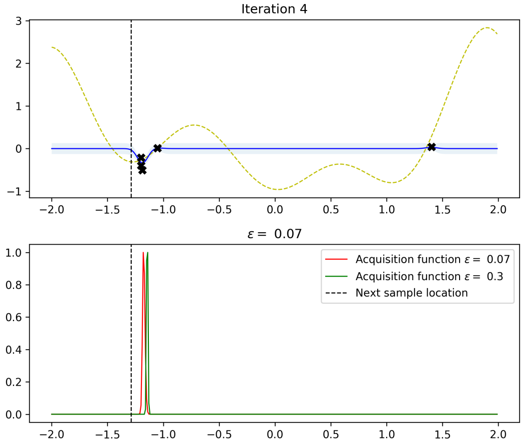


Figure: Bayesian optimization iterations using PI

PI example...

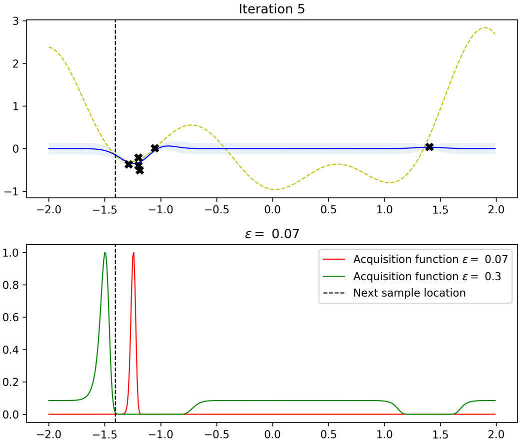


Figure: Bayesian optimization iterations using PI

PI example...

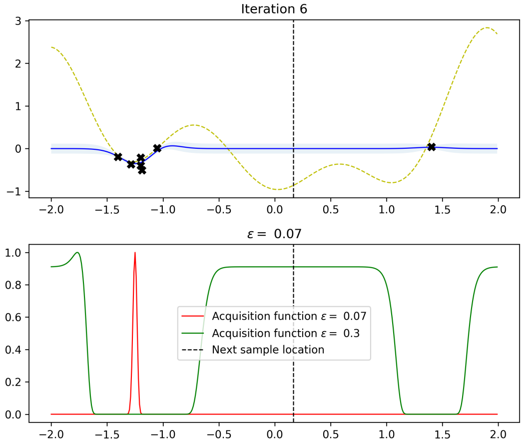


Figure: Bayesian optimization iterations using PI

PI example...

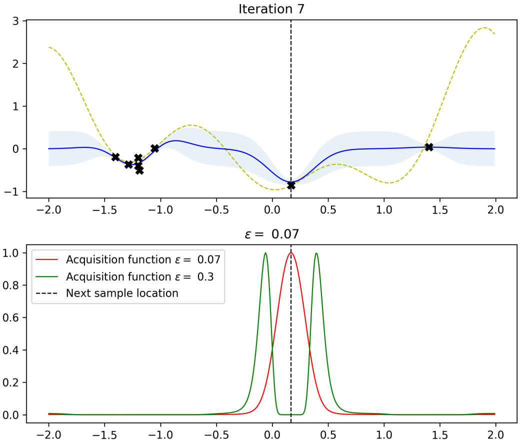


Figure: Bayesian optimization iterations using PI

PI example...

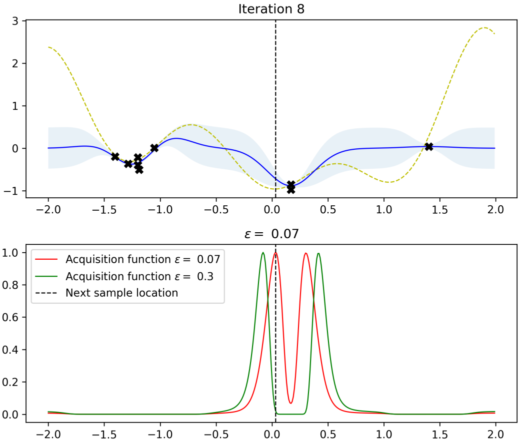


Figure: Bayesian optimization iterations using PI

PI example...

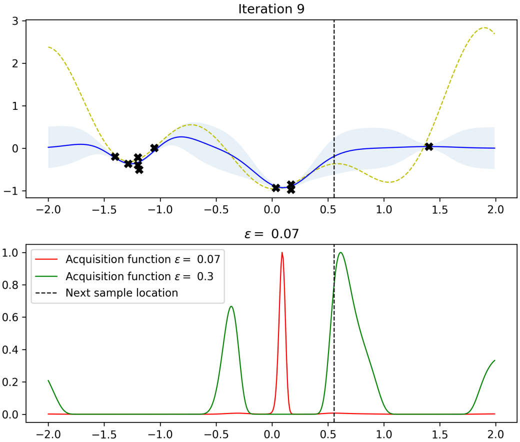


Figure: Bayesian optimization iterations using PI

PI example...

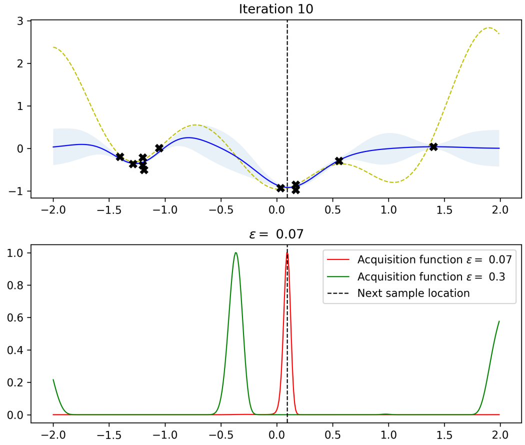


Figure: Bayesian optimization iterations using PI

Expected improvement

The probability of improvement only looked at how likely an improvement is but did not consider how much we can improve. The next criterion, called Expected Improvement (EI), does exactly that!

Expected improvement

The probability of improvement only looked at how likely an improvement is but did not consider how much we can improve. The next criterion, called Expected Improvement (EI), does exactly that!

The idea is to choose the next query point as the one which has the highest expected improvement over the current max $f(\mathbf{x}^+)$, where $\mathbf{x}^+ = \arg\max_{\mathbf{x}_i \in \mathbf{x}_{1:t-1}} f(\mathbf{x}_i)$ and $\mathbf{x}_i$ is the location queried at i^{th} time step.

Expected improvement is defined as

$$\mathrm{EI}(\mathbf{x}) = \mathbb{E}[\max((f(\mathbf{x}) - f(\mathbf{x}^+), 0)]$$

$$\mathrm{EI}(\mathbf{x}) = \int_{-\infty}^{\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma(\mathbf{x})}} \left(\mu(\mathbf{x}) - f(\mathbf{x}^+) - \sigma(\mathbf{x})\varepsilon\right) \phi(\varepsilon) d\varepsilon$$

Expected improvement...

The expected improvement can be evaluated analytically under the GP [4]:

$$\mathrm{EI}(\mathbf{x}) = \begin{cases} (\mu(\mathbf{x}) - f(\mathbf{x}^+) - \epsilon)\Phi(Z) + \sigma(\mathbf{x})\phi(Z) & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases} \tag{14}$$

where

$$Z = \begin{cases} \frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \epsilon}{\sigma(\mathbf{x})} & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases} \tag{15}$$

Φ and ϕ are the CDF and PDF of the standard normal distribution, respectively.

EI example

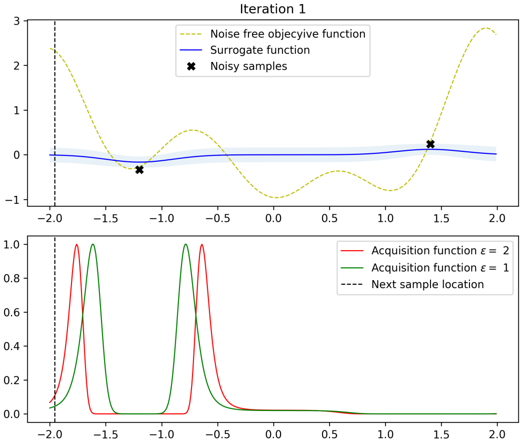


Figure: Bayesian optimization iterations using EI

EI example

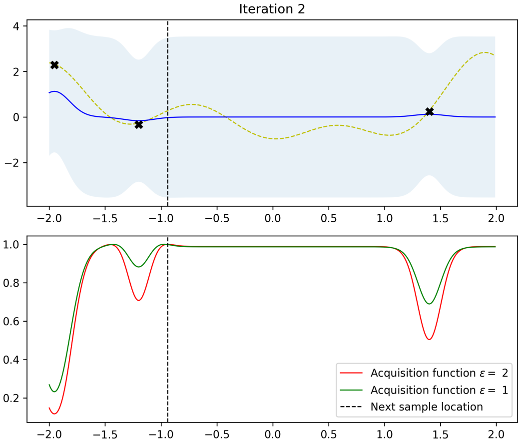


Figure: Bayesian optimization iterations using EI

EI example

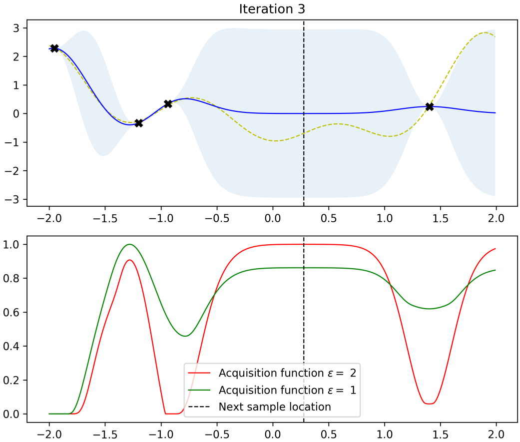


Figure: Bayesian optimization iterations using EI

EI example

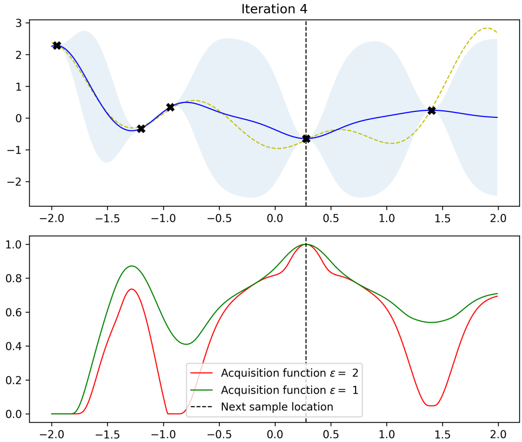


Figure: Bayesian optimization iterations using EI

EI example

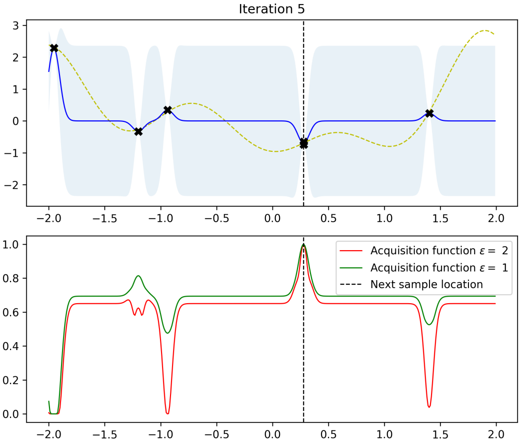


Figure: Bayesian optimization iterations using EI

EI example

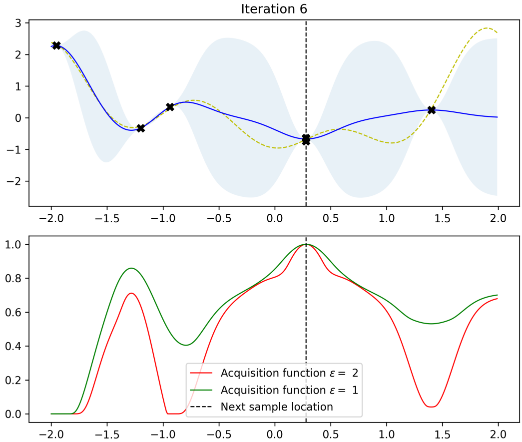


Figure: Bayesian optimization iterations using EI

EI example

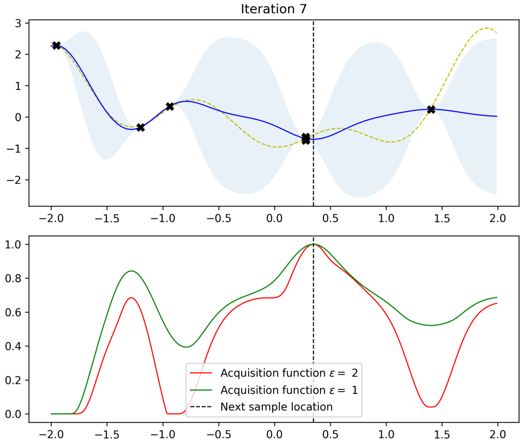


Figure: Bayesian optimization iterations using EI

EI example

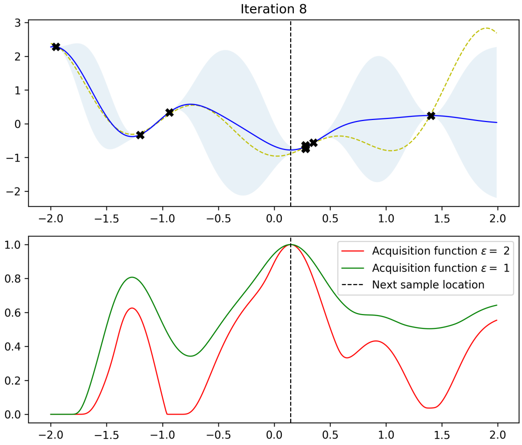


Figure: Bayesian optimization iterations using EI

EI example

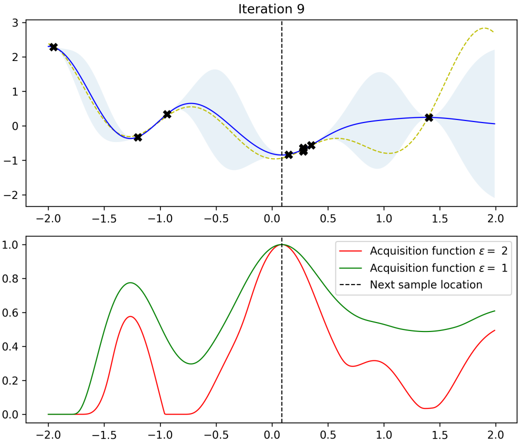


Figure: Bayesian optimization iterations using EI

EI example

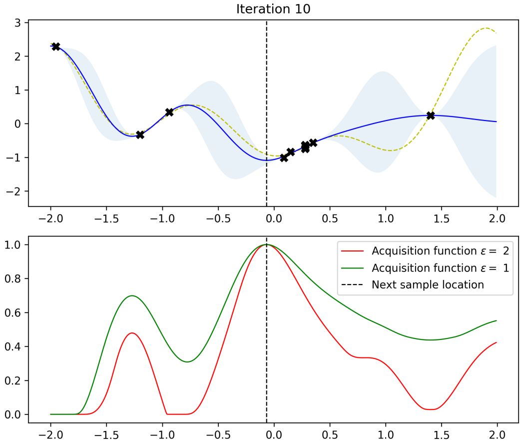


Figure: Bayesian optimization iterations using EI

Hyperparameter tuning with Bayesian optimization

Let's consider the problem of tuning the hyperparameters of a XGBoost model to predict diabetes.

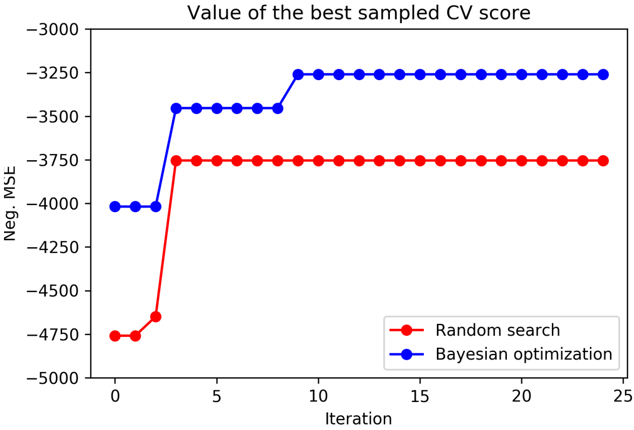


Figure: Hyperparameter tuning using Bayesian Optimization

Financial and trading applications

BO tackles the same problem as in Multi-Armed Bandits.

The GP_UCB (Upper Confidence Bound) is equivalent to the minimisation of the cumulative regret.

It could also be extended to contexts where the objective function is time-dependent.
$k(\{\mathbf{x}_i, t_i\}, \{\mathbf{x}_j, t_j\}) = k_{\text{Gabor}}(t_i, t_j) \times k_{\text{Linear}}(\mathbf{x}_i, \mathbf{x}_j)$.

Adaptive Bayesian Optimisation for Online Portfolio Selection

Favour M. Nyikosa, Michael A. Osborne and Stephen J. Roberts
Department of Engineering Science
University of Oxford
{favour,mosb,sjrob}@robots.ox.ac.uk

Figure: Bayesian optimization in finance

Agro-industrial applications

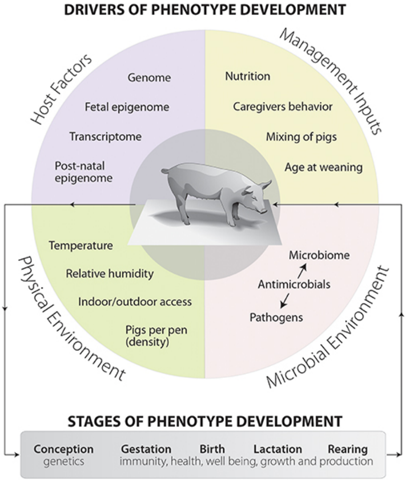


Figure: Bayesian optimization in precision agriculture. First image taken from [6]

Agro-industrial applications

Nikitin *et al. Plant Methods* (2019) 15:43
https://doi.org/10.1186/s13007-019-0422-z

Plant Methods

RESEARCH **Open Access**

Bayesian optimization for seed germination

Check for updates

Artyom Nikitin[1*], Ilia Fastovets[1,2], Dmitrii Shadrin[1], Mariia Pukalchik[1] and Ivan Oseledets[1]

Figure: Bayesian optimization in precision agriculture. First image taken from [6]

SSR

Bayesian optimization libraries

- Scikit-optimize.
- pyGPGO
- KerasTuner
- RAY Tune
- GPflowOpt
- Trieste
- GPyTorch
- Botorch
- Dragonfly
- Vizier
- GPyOpt is a Bayesian optimization library based on GPy. No longer supported.

Further readings

- Other acquisition functions: Entropy search, Predictive entropy search, Thomson sampling, Monte Carlo acquisition functions, etc.
- Parallel (Batch) Bayesian optimization.
- Multi-objective Bayesian optimization.
- Optimization over non-Euclidean spaces.
- Restrictions: known and unknown
- Causal Bayesian optimization
- Several parts of this presentation were based on the outstanding tutorial by Martin Krasser available on: http://krasserm.github.io/2018/03/21/bayesian-optimization/

Questions?

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Eric Brochu, Vlad M Cora, and Nando De Freitas. "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning". In: *arXiv preprint arXiv:1012.2599* (2010).

[3] Melvyn W Jeter. *Mathematical programming: an introduction to optimization*. Routledge, 2018.

[4] Donald R Jones, Matthias Schonlau, and William J Welch. "Efficient global optimization of expensive black-box functions". In: *Journal of Global optimization* 13.4 (1998), pp. 455–492.

[5] Carl Edward Rasmussen and Christopher K Williams. *Gaussian processes for machine learning*. Vol. 2. 3. MIT press Cambridge, MA, 2006.

[6] Mohamed Zeineldin, Brian Aldridge, and James Lowe. "Antimicrobial effects on swine gastrointestinal microbiota and their accompanying antibiotic resistome". In: *Frontiers in microbiology* 10 (2019), p. 1035.

Thank You

Julián D. Arias-Londoño
julian.arias@upm.es