

# Bayesian Optimization in Machine Learning

José Jiménez

Master's Thesis

Master's degree in Statistics and Operations Research

supervised by Josep Ginebra<sup>1</sup>

<sup>1</sup> Department of Statistics and Operations Research. ETSEIB.



UNIVERSITAT POLITÈCNICA  
DE CATALUNYA  
BARCELONATECH



UNIVERSITAT DE  
BARCELONA

# Goals of this thesis

This Master's thesis aims to be a multi-objective optimization task:

- Provide an introduction to both Gaussian Process regression and Bayesian optimization.
- Show that the Bayesian Optimization framework works in several real-world machine learning tasks.
- Write a complete software package (pyGPGO) for users to apply Bayesian Optimization in their research.

# Organization of the work

Organized in 5 self-contained chapters.

- **Chapter 2** focuses on an introduction to regression problems using Gaussian Processes. These are surrogate models we will use for Bayesian Optimization.
- **Chapter 3** covers the main topic in this work, Bayesian Optimization.
- **Chapter 4** presents experiments using the Bayesian Optimization framework. Mostly mid-sized supervised-learning problems.
- **Chapter 5** provides technical explanations for pyGPGO, the software developed alongside this manual.

# A brief introduction

Overall, Bayesian Optimization focuses on:

$$\max_{\mathbf{x} \in \mathcal{A}} f(\mathbf{x}) \quad (1)$$

We make almost no assumptions about  $f$ :

- $f$  may not have a closed-form expression.
- Evaluations of  $f$  may be noisy.
- Gradient information is optional.

These situations arise when optimizing the *loss* function of a machine-learning model, depending on its hyperparameters (e.g. log-loss):

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{n} \sum_i (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)) \quad (2)$$

# Gaussian Process Regression: basic definitions

## Definition 1.

A Gaussian Process is a collection of random variables, any finite number of which have a joint Gaussian distribution. This process is defined by two functions. Its *mean function*:

$$m(\mathbf{x}) = \mathbb{E} [f(\mathbf{x})] \quad (3)$$

and its *covariance function*:

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E} [(f(\mathbf{x}) - m(\mathbf{x})) (f(\mathbf{x}') - m(\mathbf{x}'))] \quad (4)$$

We say that  $f$  is a Gaussian Process with mean  $m(\mathbf{x})$  and covariance function  $k(\mathbf{x}, \mathbf{x}')$  and write:

$$f(\mathbf{x}) \sim \mathcal{GP} (m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \quad (5)$$

# Gaussian Process Regression: basic definitions

Define then a covariance function, such as the *squared exponential* kernel:

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}|\mathbf{x} - \mathbf{x}'|^2\right) \quad (6)$$

where  $|\cdot|$  denotes the standard  $L_2$  norm. Drawing samples from a Gaussian Process, assuming zero mean, for given finite inputs  $X_*$  simplifies to sampling from:

$$\mathbf{f}_* \sim \mathcal{N}(\mathbf{0}, K(X_*, X_*)) \quad (7)$$

# Gaussian Process Regression: prediction

Assume training data  $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid i = 1, \dots, n\}$

## Prediction using GP prior

Let  $\mathbf{y}$  and  $\mathbf{f}_*$  be jointly Gaussian:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N} \left( \mathbf{0}, \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix} \right) \quad (8)$$

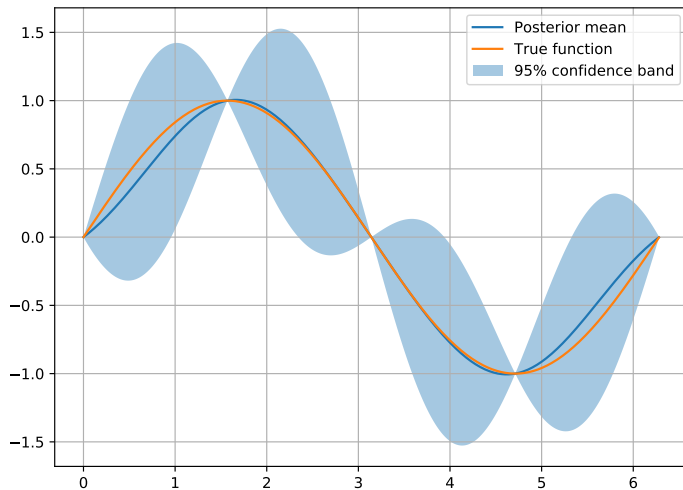
We want to condition  $\mathbf{f}_*$  over  $\mathbf{y}$ .

$$\mathbf{f}_* | \mathbf{y} \sim \mathcal{N}(\bar{\mathbf{f}}_*, \text{Cov}(\mathbf{f}_*)) \quad (9)$$

where:

$$\begin{aligned} \bar{\mathbf{f}}_* &= K(X_*, X) (K(X, X) + \sigma_n^2 I)^{-1} \mathbf{y} \\ \text{Cov}(\mathbf{f}_*) &= K(X_*, X_*) - K(X_*, X) (K(X, X) + \sigma_n^2 I)^{-1} K(X, X_*) \end{aligned} \quad (10)$$

# Gaussian Process Regression: an example





# Gaussian Process Regression: on covariance functions

## Some common covariance function choices

$$\begin{aligned}
 k_{SE}(r) &= \exp\left(-\frac{r^2}{2l^2}\right) & k_{\text{Matèrn}}(r) &= \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}r}{l}\right)^\nu K_\nu\left(\frac{\sqrt{2\nu}r}{l}\right) \\
 k_{\text{GE}}(r) &= \exp\left(-\left(\frac{r}{l}\right)^\gamma\right) & k_{RQ}(r) &= \left(1 + \frac{r^2}{2\alpha l^2}\right)^{-\alpha}
 \end{aligned}
 \tag{11}$$

where  $r = |\mathbf{x} - \mathbf{x}'|$ . Observations are noisy:

$$k^y(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 k(\mathbf{x}_p, \mathbf{x}_q) + \sigma_n^2 \delta_{pq} \tag{12}$$

# Gaussian Process Regression: Type II Maximum-Likelihood

An empirical Bayes approach to choosing hyperparameters. Noticing that  $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, K + \sigma_n^2 I)$

## Marginal log-likelihood

$$\log p(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^T (K + \sigma_n^2 I)^{-1} \mathbf{y} - \frac{1}{2} \log |K + \sigma_n^2 I| - \frac{n}{2} \log 2\pi \quad (13)$$

## Derivative w.r.t hyperparameters

$$\frac{\partial}{\partial \theta_j} \log p(\mathbf{y}|X, \theta) = \frac{1}{2} \mathbf{y}^T K^{-1} \frac{\partial K}{\partial \theta_j} K^{-1} \mathbf{y} - \frac{1}{2} \text{tr} \left( K^{-1} \frac{\partial K}{\partial \theta_j} \right) \quad (14)$$

# Gaussian Process Regression: incorporating diff. priors

We have assumed  $m(\mathbf{x}) = 0$  for simplicity. If we dispose of prior knowledge, we can use it

## Different prior mean

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}^*)) \quad (15)$$

Posterior mean becomes:

$$\mathbf{f}_* = \mathbf{m}(X_*) + k(X_*, X)K^{-1}(\mathbf{y} - \mathbf{m}(X)) \quad (16)$$

Posterior variance remains unchanged.

# Gaussian Process Regression: marginalizing over hyperparameters

The full Bayesian approach does not optimize the marginal likelihood, but integrates the uncertainty of hyperparameters  $\theta$  into the model, either using MCMC or Variational Inference techniques.

## Full Bayesian approach

$$\theta \sim p_h(\theta) \quad (17)$$

$$\mathbf{f} \sim \mathcal{N}(0, \Sigma_\theta) \quad (18)$$

$$\mathcal{L}(\mathbf{f}) = p(\text{data}|\mathbf{f}) = p(\mathbf{y}|\mathbf{f}) \quad (19)$$

We wish to sample from the joint posterior under unknowns:

$$p(\mathbf{f}, \theta | \text{data}) \propto \mathcal{L}(\mathbf{f})p(\mathbf{f})p_h(\theta) \quad (20)$$

Several strategies for sampling from latent Gaussian models proposed in main text.

# Bayesian Optimization: introduction

Again, assume:

$$\max_{\mathbf{x} \in \mathcal{A}} f(\mathbf{x}) \quad (21)$$

Assume that we have sampled our function  $f$  to optimize a small number of times  $n$ . Providing us with training data:

$$\mathcal{D}_n = \{\mathbf{x}_i, y_i, i = 1, \dots, n\}. \quad (22)$$

Our steps here are:

- 1 Fit a Gaussian Process regression model on  $\mathcal{D}_n$ .
- 2 Choose the next point to sample, according to an *acquisition function*, depending on GP.
- 3 Evaluate said point. Augment training data. Repeat.

# Bayesian optimization: algorithm

---

**Algorithm 1** Bayesian optimization framework

---

- 1: Sample a small number of points  $\mathbf{x} \in \mathcal{A}$ . Evaluate  $f(\mathbf{x})$  to get  $\mathcal{D}_n$
  - 2: **for**  $n = 1, 2, \dots$  **do**
  - 3:   Fit a GP regression model on  $\mathcal{D}_n$
  - 4:    $\mathbf{x}_{n+1} \leftarrow \arg \max_{\mathbf{x}} \alpha(\mathbf{x}, \mathcal{D}_n)$
  - 5:   Evaluate  $f(\mathbf{x}_{n+1}) = y_{n+1}$
  - 6:   Augment data  $\mathcal{D}_{n+1} = \{\mathcal{D}_n, (\mathbf{x}_{n+1}, y_{n+1})\}$
  - 7: **end for**
-

# Bayesian optimization: example

# Bayesian optimization: on acquisition functions

Three popular categories: improvement-based, optimistic, and information-based policies.

## Improvement-based

$\tau$  denotes our best evaluation so far.

Probability of improvement:

$$\alpha_{\text{PI}}(\mathbf{x}, \mathcal{D}_n) = P(\nu > \tau) = \Phi\left(\frac{\mu_n(\mathbf{x}) - \tau}{\sigma_n(\mathbf{x})}\right) \quad (23)$$

Expected improvement:

$$\alpha_{\text{EI}}(\mathbf{x}, \mathcal{D}_n) = (\mu_n(\mathbf{x}) - \tau)\Phi\left(\frac{\mu_n(\mathbf{x}) - \tau}{\sigma_n(\mathbf{x})}\right) + \sigma_n(\mathbf{x})\phi\left(\frac{\mu_n(\mathbf{x}) - \tau}{\sigma_n(\mathbf{x})}\right) \quad (24)$$



# Bayesian optimization: on acquisition functions

## Optimistic acquisitions

Upper confidence bound:

$$\alpha_{\text{UCB}}(\mathbf{x}, \mathcal{D}_n) = \mu_n(\mathbf{x}) + \beta_n \sigma_n(\mathbf{x}) \quad (25)$$

## Information-based

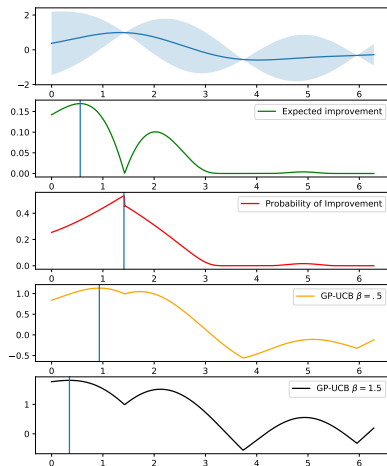
Entropy-based:

$$\alpha_{\text{ES}}(\mathbf{x}|\mathcal{D}_n) = H(\mathbf{x}^*|\mathcal{D}_n) - \mathbb{E}_{y|\mathcal{D}_n, \mathbf{x}} [H(\mathbf{x}^*|\mathcal{D}_n \cup \{(\mathbf{x}, y)\})] \quad (26)$$

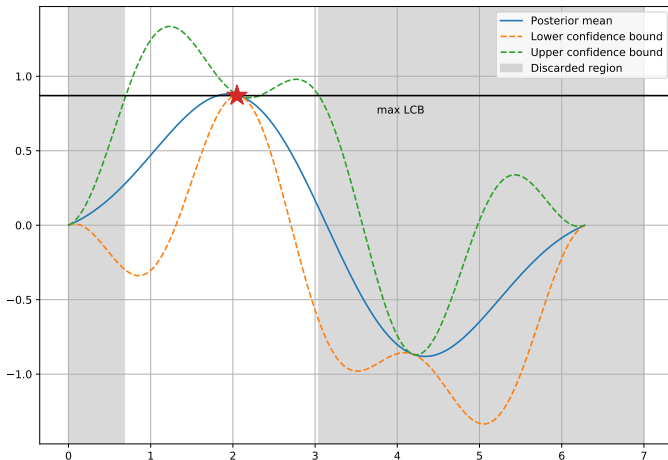
Predictive entropy search:

$$\alpha_{\text{PES}}(\mathbf{x}, \mathcal{D}_n) = H(y|\mathcal{D}_n, \mathbf{x}) - \mathbb{E}_{\mathbf{x}^*|\mathcal{D}_n} [H(y|\mathcal{D}_n, \mathbf{x}, \mathbf{x}^*)] \quad (27)$$

# Bayesian optimization: acquisition function behaviour



# Bayesian optimization: why does it work?



# Bayesian optimization: role of hyperparameters

GP hyperparameters  $\theta$  cause uncertainty in the optimization procedure:

$$\alpha(\mathbf{x}) = \mathbb{E}_{\theta|\mathcal{D}_n} [\alpha(\mathbf{x}, \theta)] = \int_{\Theta} \alpha(\mathbf{x}|\theta) p(\theta|\mathcal{D}_n) d\theta. \quad (28)$$

Approach 1. MAP estimate  $\hat{\theta}_{\text{MAP}}$

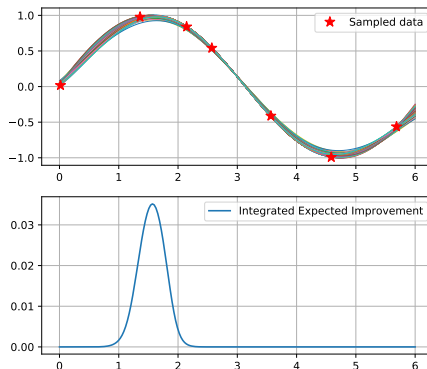
$$\hat{\alpha}(\mathbf{x}) = \alpha(\mathbf{x}, \hat{\theta}) \quad (29)$$

Approach 2. Sample posterior  $p(\theta|\mathcal{D}_n)$ . Integrated acquisition functions

$$\mathbb{E}_{\theta|\mathcal{D}_n} [\alpha(\mathbf{x}, \theta)] \approx \frac{1}{M} \sum_{i=1}^M \alpha(\mathbf{x}, \theta_n^{(i)}) \quad (30)$$

# Bayesian optimization: Integrated acquisition example

**Figure:** Means of 200 posterior predictive distributions, taken from sampled posterior  $p(\theta|\mathcal{D}_n)$



# Bayesian optimization: optimizing acquisitions

We have assumed that the acquisition function is easily optimized. This is not necessarily the case:

- It is often multimodal.
- Or non-convex.
- Furthermore theoretical convergence is guaranteed when finding true maximum  $\mathbf{x}^*$ .

One may be thinking that we have switched optimization problems! ( $f$  by  $\alpha$ ). However,  $\alpha$  is very cheap to compute.

- In practice  $\alpha$  is optimized using multi-start quasi-Newton methods (e.g. L-BFGS-B) or evolutionary approaches (CMA-ES).

# Bayesian optimization: Other surrogates

- Random Forests: empirical estimate of variance provided by predictions of trees.
- Gradient Boosting Machines: using quantile regression for variance estimate.
- Sparse pseudo-input GPs
- Student-t Processes
- ...

## Student-t Process example

$$\mathbf{f}^* | \mathbf{f} \sim \mathcal{T} \left( \nu + n_1, \phi_2, \frac{\nu + \beta_1 - 2}{\nu + n_1 - 2} \tilde{K}_{**} \right) \quad (31)$$

# Experiments: benchmarking rules

Loss functions used:

## Binary classification

Logarithmic loss:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{n} \sum_i (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)) \quad (32)$$

## Regression

Mean Squared Error (MSE)

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \sum_i (y_i - \hat{y}_i)^2 \quad (33)$$



# Experiments: benchmarking rules

- Shuffled  $k = 5$  fold cross-validation scheme for loss evaluation.
- Features scaled: zero mean, unit variance.
- $n = 50$  (+3 to account GP fitting) total function evaluations.
- Gaussian Process surrogate, with squared exponential covariance function.
- Type II ML estimation of all hyperparameters at each stage ( $l$ ,  $\sigma_n^2$ ,  $\sigma_f^2$ ).
- Expected Improvement and GP-UCB ( $\beta = 0.5$ ,  $\beta = 1.5$ ) acquisitions.
- Machine-learning models used: Support Vector Machines, K-Nearest-Neighbours, Gradient Boosting Machines and One-Hidden Layer Neural Networks.

# Experiments: benchmarking rules

Table: SVM parameters.

Parameter	Type	Bounds
$C$	$\mathbb{R}^+$	$[10^{-5}, 10^5]$ (log-scaled)
$\gamma$	$\mathbb{R}^+$	$[10^{-5}, 10^5]$ (log-scaled)

Table: KNN parameters.

Parameter	Type	Bounds
$k$	Integer	$\{10, \dots, 50\}$

# Experiments: benchmarking rules

Table: GBM parameters.

Parameter	Type	Bounds
learning_rate	$\mathbb{R}^+$	$[10^{-5}, 10^{-2}]$
n_estimators	Integer	$\{10, \dots, 100\}$
max_depth	Integer	$\{2, \dots, 100\}$
min_samples_split	Integer	$\{2, \dots, 100\}$

Table: MLP parameters.

Parameter	Type	Bounds
hidden_layer_size	Integer	$[5, 50]$
alpha	$\mathbb{R}^+$	$[0, 0.9]$

# Experiments: Binding affinity prediction