Financial Applications of Gaussian Processes and Bayesian Optimization*

Joan Gonzalvez
Quantitative Research
Amundi Asset Management, Paris
joan.gonzalvez@amundi.com

Edmond Lezmi
Quantitative Research
Amundi Asset Management, Paris
edmond.lezmi@amundi.com

Thierry Roncalli
Quantitative Research
Amundi Asset Management, Paris
thierry.roncalli@amundi.com

Jiali Xu
Quantitative Research
Amundi Asset Management, Paris
jiali.xu@amundi.com

February 2019

Abstract

In the last five years, the financial industry has been impacted by the emergence of digitalization and machine learning. In this article, we explore two methods that have undergone rapid development in recent years: Gaussian processes and Bayesian optimization. Gaussian processes can be seen as a generalization of Gaussian random vectors and are associated with the development of kernel methods. Bayesian optimization is an approach for performing derivative-free global optimization in a small dimension, and uses Gaussian processes to locate the global maximum of a black-box function. The first part of the article reviews these two tools and shows how they are connected. In particular, we focus on the Gaussian process regression, which is the core of Bayesian machine learning, and the issue of hyperparameter selection. The second part is dedicated to two financial applications. We first consider the modeling of the term structure of interest rates. More precisely, we test the fitting method and compare the GP prediction and the random walk model. The second application is the construction of trend-following strategies, in particular the online estimation of trend and covariance windows.

Keywords: Gaussian process, Bayesian optimization, machine learning, kernel function, hyperparameter selection, regularization, time-series prediction, asset allocation, portfolio optimization, trend-following strategy, moving-average estimator, ADMM, Cholesky trick.

JEL classification: C61, C63, G11.

*We would like to thank Elisa Baku and Thibault Bourgeron for their helpful comments.

1 Introduction

This article explores the use of Gaussian processes and Bayesian optimization in finance. These two tools have been successful in the machine learning community. In recent years, machine learning algorithms have been applied in risk management, asset management, option trading and market making. Despite the skepticism about earlier implementations, we must today recognize that machine learning is changing the world of finance. Banks, asset managers, hedge funds and robo-advisors have invested a lot of money in such technologies, and all the reports agree that it is just the beginning (McKinsey, 2015; OECD, 2017; Oliver Wyman, 2018). Even supervisory bodies are closely monitoring this development and its impact on the financial industry (FSB, 2017). Certainly, the most impressive indicator is the evolution of the financial job market (BCG, 2018). Today, applicants in the quant finance must have a certification in machine learning or at least knowledge of this technology and experience in the Python programming language.

In our two previous works, we focus on asset allocation and portfolio construction. In Bourgeron *et al.* (2018), we discuss how to design a comprehensive and automated portfolio optimization model for robo-advisors. In Richard and Roncalli (2019), we extend this approach when we consider risk budgeting portfolios in place of mean-variance portfolios. This third paper continues the '*tour d'horizon*' of machine learning techniques that can be useful for asset management challenges. However, we are significantly changing the direction since we move away from portfolio optimization and we are interested here in estimation and forecasting problems.

A Gaussian process (GP) is generalization of a Gaussian random vector, and can be seen as a stochastic process on general continuous functions. This can be done because it replaces the traditional covariance matrix by a kernel function, and benefits from the power of kernel methods. The core of this approach is the computation of the conditional distribution. In a Bayesian framework, this is equivalent to computing the posterior distribution from the prior distribution. Since linear regression is the solution of the conditional expectation problem when the random variables are Gaussian, it is then straightforward to define Gaussian process regression, which is a powerful semi-parametric machine learning model (Rasmussen and Williams, 2006) that has been successfully used in geostatistics (Cressie, 1993), multi-task learning (Alvarez *et al.*, 2012) or robotics and reinforcement learning (Deisenroth *et al.*, 2015). Bayesian optimization is an approach used to solve black-box optimization problems (Bochu *et al.*, 2009; Frazier, 2018) where the objective function is not explicitly known and costly to evaluate. Without access to the gradient vector of the objective function, usual quasi-Newton or gradient-descent methods are unusable. Bayesian optimization models the unknown function as a random Gaussian process surrogate and replaces the intractable original problem by a sequence of simpler optimization problems. In this case, GPs appear as a tool in Bayesian optimization. Generally, a financial model depends on some external parameters that have to be fixed before running the model. Bayesian optimization mainly concerns the estimation of these external parameters, which are called hyperparameters. Examples are the length of a moving-average estimator, the risk aversion of the investor or the window of a covariance matrix of asset returns.

This paper is organized as follows. Section Two reviews the mathematics of Gaussian processes and Bayesian optimization. In particular, we present the technique of Gaussian process regression and discuss the issue of hyperparameter selection. In Section Three, we use Gaussian processes in order to fit the term structure of the interest rates, and show how they can be used for forecasting the yield curve. The second application of Section Three concerns the online estimation of the hyperparameters of the trend-following strategy. Finally, Section Four offers some concluding remarks.

2 A primer on Gaussian processes and Bayesian optimization

In this section, we define the main concepts and techniques used in machine learning with Gaussian processes. In regression and classification problems, they are used for interpolation, extrapolation and pattern discovery for which the choice of the kernel function is central. Contrary to many supervised learning algorithms, GPs have the distinctive property of estimating the variance of the prediction or the confidence region for a test sample. This feature is useful for global optimization and helps to improve the objective function, because validation samples can be evaluated according to the confidence of the model.

Bayesian optimization is a statistical approach used to solve black-box optimization problems, where the objective function is not explicitly known or costly to evaluate. Since the gradient of the objective function is difficult to evaluate or unknown, descent methods are unusable. In this case, Bayesian optimization replaces the unknown objective function by a random Gaussian process and the intractable original problem by a sequence of simpler optimization problems. This explains why Gaussian processes and Bayesian optimization are closely related.

2.1 Gaussian processes

2.1.1 Definition

Let $\mathcal{X}$ be a set[1] in $\mathbb{R}^d$. A Gaussian process is a collection $\{f(x),\ x \in \mathcal{X}\}$ such that for any $n \in \mathbb{N}$ and $x_1, \ldots, x_n \in \mathcal{X}$, the random vector $(f(x_1), \ldots, f(x_n))$ has a joint multivariate Gaussian distribution (Rasmussen and Williams, 2006). Therefore, we can characterize the GP by its mean function:

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

and its covariance function:

$$\begin{aligned} \mathcal{K}(x, x') &= \operatorname{cov}(f(x), f(x')) \\ &= \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))] \end{aligned}$$

The covariance function $\mathcal{K}(x, x')$ is central in the analysis of Gaussian processes, and is called the '*kernel*' function. In machine learning, the most popular kernel function is the squared exponential kernel[2], which is given by:

$$\mathcal{K}_{\mathrm{SE}}(x, x') = \exp\left(-\frac{1}{2}\|x - x'\|_2^2\right) \tag{1}$$

for $x \in \mathbb{R}^d$ and $x' \in \mathbb{R}^d$.

Remark 1. *In what follows, we assume that* $m(x) = \mathbf{0}$ *without loss of generality.*

2.1.2 Gaussian process regression

Given a training set $\{(x_i, y_i)\}_{i=1}^n$ of features, the goal of Gaussian process regression (GPR) is to forecast $f(x^\star)$ for some new inputs $x^\star \in \mathbb{R}^{n^\star \times d}$. For that, we adopt the Bayesian framework, and we compute the posterior distribution of the GP conditionally on $x = (x_1, \ldots, x_n) \in \mathbb{R}^{n \times d}$.

[1] The set $\mathcal{X}$ of inputs can be multi-dimensional (linear regression modeling), time-dimensional (time series forecasting), etc.

[2] It is also known as the radial basis function (RBF) kernel or the Gaussian kernel.

The noise-free case Let us first assume that the observations are noise-less, meaning that $y = f(x) = (f(x_1), \dots, f(x_n))$ where f is the GP. Let $x^\star$ be $n^\star$ new inputs and $\hat{y}^\star = \mathbb{E}[f(x^\star \mid x, y)]$ be the conditional prediction. Then we have:

$$f(x, x^\star) \sim \mathcal{N}\left(\mathbf{0}_{n+n^\star}, \begin{pmatrix} \mathcal{K}(x,x) & \mathcal{K}(x,x^\star) \\ \mathcal{K}(x^\star,x) & \mathcal{K}(x^\star,x^\star) \end{pmatrix}\right) \tag{2}$$

where $\mathcal{K}(x^\star, x)$ is the $n^\star \times n$ matrix[3] with entries $\mathcal{K}_{i,j}(x^\star, x) = \mathcal{K}(x_i^\star, x_j)$. Using Appendix A.2 on page 34, it follows that the random vector $y^\star \mid x, y = f(x^\star \mid x, y)$ is also Gaussian:

$$f(x^\star \mid x, y) \sim \mathcal{N}(m(x^\star \mid x, y), \mathcal{K}(x^\star, x^\star \mid x, y))$$

where $m(x^\star \mid x, y)$ is the mean vector of the posterior distribution[4]:

$$m(x^\star \mid x, y) = \mathcal{K}(x^\star, x)\,\mathcal{K}(x, x)^{-1}\, y$$

and the covariance matrix $\mathcal{K}(x^\star, x^\star \mid x, y)$ is the *Schur's complement* of the prior:

$$\mathcal{K}(x^\star, x^\star \mid x, y) = \mathcal{K}(x^\star, x^\star) - \mathcal{K}(x^\star, x)\,\mathcal{K}(x, x)^{-1}\,\mathcal{K}(x, x^\star)$$

We deduce that the prediction is the conditional expectation:

$$\hat{y}^\star = m(x^\star \mid x, y)$$

We notice that computing the posterior distribution requires us to invert the $n \times n$ matrix $\mathcal{K}(x, x)$. Since it is a covariance matrix, it is a symmetric positive semi-definite matrix and the Cholesky decomposition can be applied leading to $O(n^3)$ operations.

Remark 2. *In order to reduce the notation complexity, we introduce the hat notation for writing conditional quantities. We have* $\hat{f}(x^\star) = f(x^\star \mid x, y)$, $\hat{m}(x^\star) = m(x^\star \mid x, y)$ *and* $\hat{\mathcal{K}}(x^\star, x^\star) = \mathcal{K}(x^\star, x^\star \mid x, y)$.

Gaussian noise In order to take into account noise in the data, we assume that $y = f(x) + \varepsilon$ where $\varepsilon \sim \mathcal{N}(\mathbf{0}_n, \sigma_\varepsilon^2 I_n)$. In this case, Equation (2) becomes:

$$f(x, x^\star) \sim \mathcal{N}\left(\mathbf{0}_{n+n^\star}, \begin{pmatrix} \mathcal{K}(x,x) + \sigma_\varepsilon^2 I_n & \mathcal{K}(x,x^\star) \\ \mathcal{K}(x^\star,x) & \mathcal{K}(x^\star,x^\star) \end{pmatrix}\right) \tag{3}$$

Again, the posterior distribution is Gaussian and we have:

$$\hat{f}(x^\star) \sim \mathcal{N}\left(\hat{m}(x^\star), \hat{\mathcal{K}}(x^\star, x^\star)\right)$$

where:

$$\hat{m}(x^\star) = \mathcal{K}(x^\star, x)\left(\mathcal{K}(x, x) + \sigma_\varepsilon^2 I_n\right)^{-1} y$$

and:

$$\hat{\mathcal{K}}(x^\star, x^\star) = \mathcal{K}(x^\star, x^\star) - \mathcal{K}(x^\star, x)\left(\mathcal{K}(x, x) + \sigma_\varepsilon^2 I_n\right)^{-1}\mathcal{K}(x, x^\star)$$

[3] We must not confuse the kernel function $\mathcal{K}(x, x')$ where $x \in \mathbb{R}^d$ and $x' \in \mathbb{R}^d$ that returns a scalar, and the kernel matrix $\mathcal{K}(x^\star, x)$ where $x \in \mathbb{R}^{n \times d}$ and $x^\star \in \mathbb{R}^{n^\star \times d}$ that returns a $n^\star \times n$ matrix.

[4] We reiterate that $m(x) = \mathbf{0}_n$ and $m(x^\star) = \mathbf{0}_{n^\star}$.

Scalability issues For large datasets, inverting the kernel matrix $\mathcal{K}(x,x)$ leads to a $O\left(n^3\right)$ complexity and may be prohibitive. Therefore, several methods have been proposed to adapt naive GPR to such problems (Quiñonero-Candela and Rasmussen, 2015; Quiñonero-Candela *et al.*, 2007). For instance, the subsets of regressors (SoR) algorithm uses a low-rank approximation of the matrix $\mathcal{K}(x,x)$. If we select $m < n$ samples x_m from the training set x, the approximation of $\mathcal{K}(x,x')$ is given by[5]:

$$\mathcal{K}(x,x') \approx \mathcal{K}(x,x_m)\,\mathcal{K}(x_m,x_m)^{-1}\,\mathcal{K}(x_m,x')$$

In Appendix A.3 on page 34, we show that:

$$\hat{m}(x^\star) \approx \mathcal{K}(x^\star,x_m)\,\tilde{\mathcal{K}}(x_m,x_m)^{-1}\,\mathcal{K}(x_m,x)\,y$$

and:

$$\hat{\mathcal{K}}(x^\star,x^\star) \approx \sigma_\varepsilon^2\mathcal{K}(x^\star,x_m)\,\tilde{\mathcal{K}}(x_m,x_m)^{-1}\,\mathcal{K}(x_m,x^\star)$$

where:

$$\tilde{\mathcal{K}}(x_m,x_m) = \mathcal{K}(x_m,x)\,\mathcal{K}(x,x_m) + \sigma_\varepsilon^2\mathcal{K}(x_m,x_m)$$

Gaussian process regression can then be done by inverting the $m \times m$ matrix $\tilde{\mathcal{K}}(x_m,x_m)$ instead of the $n \times n$ matrix $\mathcal{K}(x,x) + \sigma_\varepsilon^2 I_n$.

Remark 3. *Other methods to reduce the computational cost of GPR include Bayesian committee machines (BCM) introduced by Tresp (2000). The underlying idea is to train several Gaussian processes (or other kernel machines) on subsets of data and mix them according to their prediction confidence. This kind of ensemble methods can be particularly useful for large-scale problems and have been designed for parallel computation (Deisenroth and Ng, 2015; Liu et al., 2018).*

2.1.3 Covariance functions

Gaussian processes can be seen as probability distributions over functions. Therefore, the covariance function of the GP determines the properties of the function $f(x)$. For instance, $\mathcal{K}(x,x') = \min(x,x')$ is the covariance function of the Brownian motion, thus samples from a GP with this kernel function will be nowhere differentiable. Regularity, periodicity and monotonicity of the samples can all be controlled by choosing the appropriate kernel. Moreover, operations on kernels allow us to extract more complex patterns and structures in the data (Duvenaud *et al.*, 2013). Kernels can be summed, multiplied and convoluted, yielding another valid covariance function (Bishop, 2006). In what follows, we introduce the most used covariance kernels and their properties.

Usual covariance kernels A simple covariance function is the linear kernel, which is given by:

$$\mathcal{K}_{\text{Linear}}(x,x') = x^\top x'$$

GP regression using the linear kernel is equivalent to Bayesian linear regression and multiplying it by itself several times yields Bayesian polynomial regression.

One of the most used covariance functions is the SE kernel mentioned above, which can be generalized in the following way:

$$\mathcal{K}_{\text{SE}}(x,x') = \sigma^2 \exp\left(-\frac{1}{2}(x-x')^\top \Sigma\,(x-x')\right)$$

[5] x_m are called the '*inducing points*'.

where Σ is the $d \times d$ matrix that parameterizes the length scales of the inputs. Taking $\Sigma = \text{diag}\left(\ell_1^2, \ldots, \ell_d^2\right)$ allows us to scale each dimension of the inputs separately. Setting $\ell_j = 0$ will eliminate the j^{th} dimension of the input, which can be useful when constructing complex kernels. This kernel is sometimes called the automatic relevance determination (ARD) kernel since it can be used to discover relevant dimensions of the inputs when optimizing the hyperparameters $(\ell_1, \ldots, \ell_d)$. Sample functions with this covariance kernel have infinitely many derivatives.

Adding together SE kernels with different length scales gives the rational quadratic (RQ) kernel. Let us consider the Gamma distribution parameterized by shape α and rate β, whose density function is equal to:

$$g_{\alpha,\beta}(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$$

If we consider a Bayesian prior on the inverse squared length scale τ using this Gamma distribution, we obtain:

$$\mathcal{K}_{\text{SE}}(x, x' \mid \tau) = \sigma^2 e^{-\frac{1}{2}\tau r^2}$$

where $r = \|x - x'\|_2$. Then, we have:

$$\begin{aligned}
\mathcal{K}_{\text{RQ}}(x, x') &= \int_0^{+\infty} \mathcal{K}_{\text{SE}}(x, x' \mid \tau)\, g_{\alpha,\beta}(\tau)\, \mathrm{d}\tau \\
&= \sigma^2 \frac{\beta^\alpha}{\Gamma(\alpha)} \int_0^{+\infty} \tau^{\alpha-1} e^{-\left(\beta + \frac{1}{2}r^2\right)\tau}\, \mathrm{d}\tau \\
&= \sigma^2 \frac{\beta^\alpha}{\Gamma(\alpha)} \left[-\left(\beta + \frac{1}{2}r^2\right)^{-\alpha} \Gamma\left(\alpha, \left(\beta + \frac{1}{2}r^2\right)\tau\right) \right]_0^{\infty} \\
&\propto \frac{1}{\left(\beta + \frac{1}{2}r^2\right)^\alpha} \\
&\propto \left(1 + \frac{\|x - x'\|_2^2}{2\alpha\ell^2}\right)^{-\alpha}
\end{aligned}$$

where $\Gamma(\alpha, x) = \int_x^{+\infty} x^{\alpha-1} e^{-x}\, \mathrm{d}x$ is the upper incomplete gamma function and $\beta = \ell^2\alpha$ for a given ℓ. We deduce that the RQ kernel is isotropic, because it only depends on the Euclidean norm $r = \|x - x'\|_2$.

Another class of popular kernels is the Matérn family given by:

$$\mathcal{K}_{\text{Matern}}(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\, \|x - x'\|_2}{\ell} \right)^\nu K_\nu \left(\frac{\sqrt{2\nu}\, \|x - x'\|_2}{\ell} \right)$$

where ν and ℓ are two positive parameters and K_ν is the modified Bessel function of the second kind. The normalization constant is such that $\lim_{\nu \to \infty} \mathcal{K}_{\text{Matern}}(x, x') = \mathcal{K}_{\text{SE}}(x, x')$. This kernel appears quite complex but attractive, because its expression can be simplified when ν is a half-integer (Rasmussen and Williams, 2006). For instance, we have:

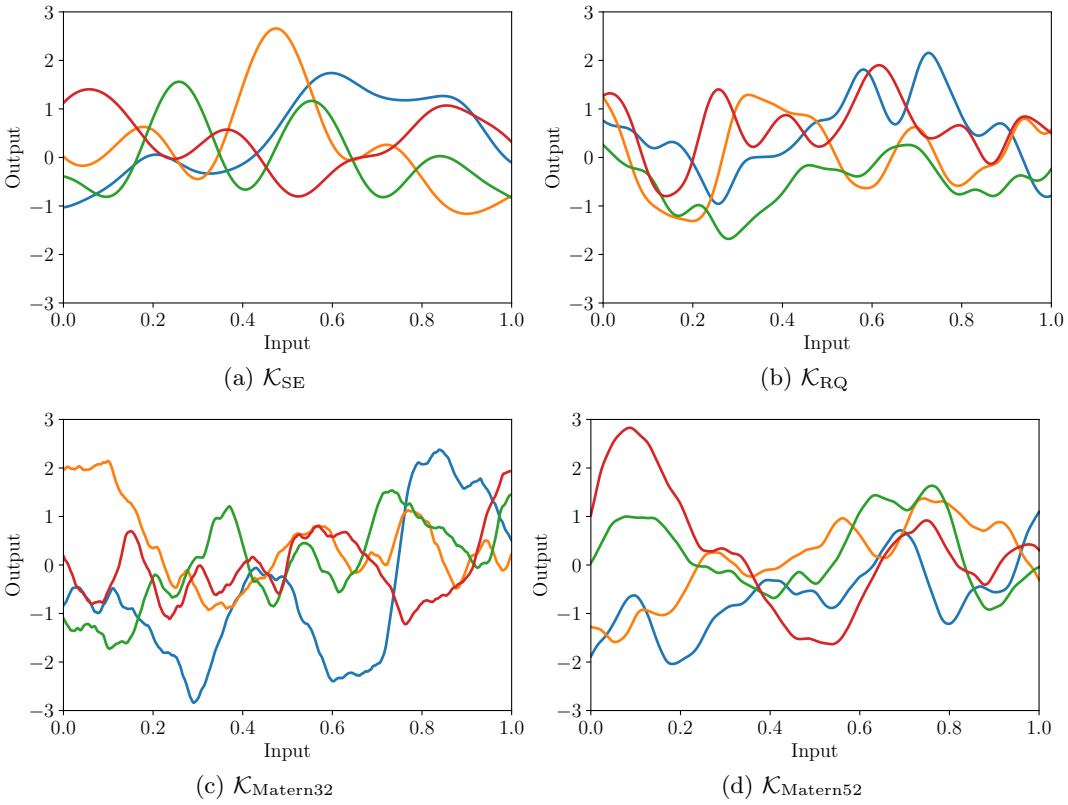
$$\begin{aligned}
\mathcal{K}_{\text{Matern32}}(x, x') &= \mathcal{K}_{\text{Matern}}\left(x, x'; \frac{3}{2}\right) \\
&= \left(1 + \frac{\sqrt{3}r}{\ell}\right) \exp\left(-\frac{\sqrt{3}r}{\ell}\right)
\end{aligned}$$

and:

$$\begin{aligned}\mathcal{K}_{\text{Matern52}}(x, x') &= \mathcal{K}_{\text{Matern}}\left(x, x'; \frac{5}{2}\right) \\ &= \left(1 + \frac{\sqrt{5}r}{\ell} + \frac{5r^2}{3\ell^2}\right) \exp\left(-\frac{\sqrt{5}r}{\ell}\right)\end{aligned}$$

Figure 1 shows four sample paths of a one-dimensional Gaussian process with different covariance functions. They are generated from a multivariate Gaussian random vector and the Cholesky factorization of the covariance matrix[6].

Figure 1: Sample path of Gaussian processes with different kernel functions



(a) $\mathcal{K}_{\text{SE}}$ (b) $\mathcal{K}_{\text{RQ}}$

(c) $\mathcal{K}_{\text{Matern32}}$ (d) $\mathcal{K}_{\text{Matern52}}$

In Figures 2 and 3, we consider a training set of 7 patterns measured without noise from the cardinal sine function $\text{sinc}(2x)$ and we report the corresponding posterior distribution $\hat{f}(x^\star)$ for two different kernels. The blue solid line corresponds to the prediction, whereas the shaded area shows the 95% confidence interval of the forecast[7]. We notice that the square exponential kernel produces a lower interpolation variance (Figure 2) than the Matern32 kernel (Figure 3), whereas the extrapolation variance is similar for the two kernel functions.

[6]We have $X = PU$ where $U \sim \mathcal{N}(\mathbf{0}_n, I_n)$, P is the Cholesky decomposition of $\mathcal{K}(x, x)$ such that $PP^\intercal = \mathcal{K}(x, x)$, $\mathcal{K}(x, x)$ is the $n \times n$ covariance matrix and x is the uniform vector on the range $[-1, 1]$.

[7]It is equal to ± 2 times the posterior standard deviation.

Figure 2: Posterior distribution of $\operatorname{sinc}(2x)$ ($\mathcal{K}_{\text{SE}}$ kernel)

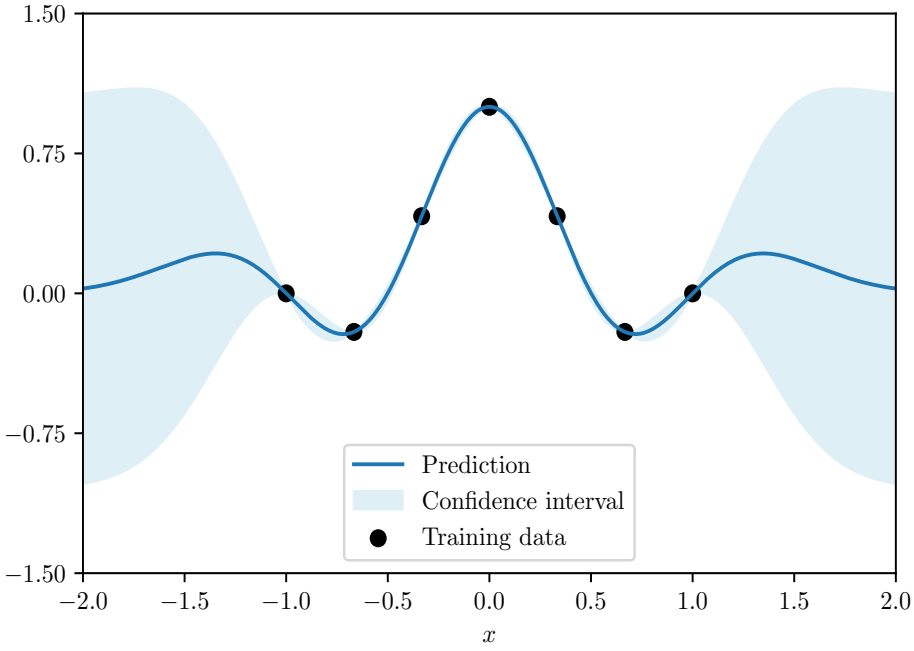
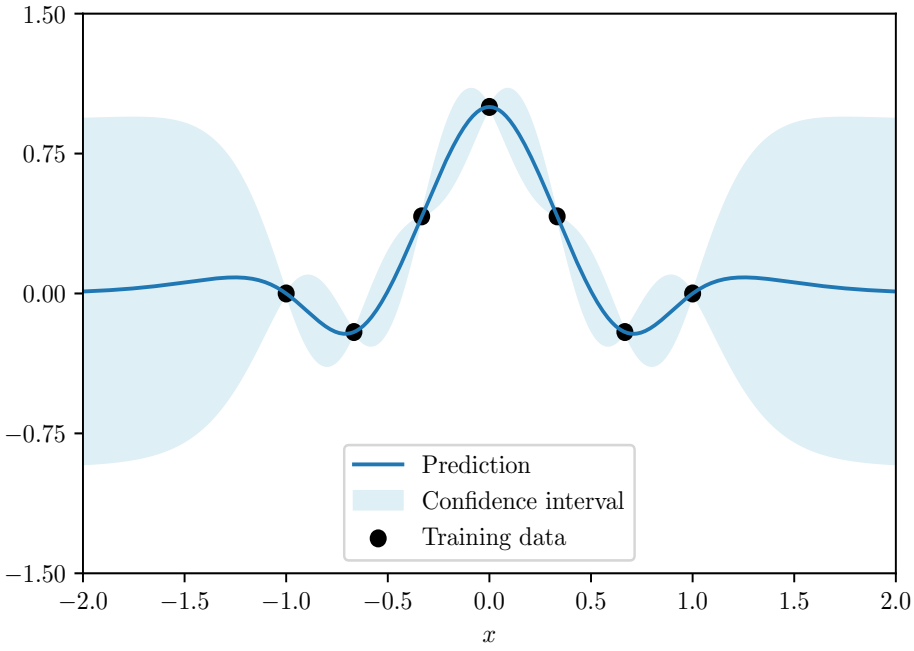


Figure 3: Posterior distribution of $\operatorname{sinc}(2x)$ ($\mathcal{K}_{\text{Matern52}}$ kernel)



Periodic exponential kernel When working with time series, it is useful to be able to include periodicity effects. Some covariance kernels exhibit this property, like the periodic exponential kernel (MacKay, 1998):

$$\mathcal{K}_{\text{PE}}(x, x') = \sigma^2 \exp\left(-\frac{1}{2}\sum_{j=1}^{d}\frac{1}{\ell_j^2}\sin^2\left(\frac{\pi}{\lambda_j}(x_j - x'_j)\right)\right)$$

where each dimension of the input space has a period λ_j. More periodic kernels can be computed from other kernels, such as the Matérn class. They can actually be built from any kernel for which the Gram matrix on a periodic basis can be computed (Durrande, 2016).

Remark 4. *Since kernels can be combined in various ways, we will often be interested in separating time patterns from space patterns. If one observation is defined by the couple (t, x) where $x \in \mathbb{R}^d$ and $t \in \mathbb{R}$ is the time index*[8]*, we can then define a kernel on the whole space-time (Osborne et al., 2012):*

$$\mathcal{K}((x,t),(x',t')) = \mathcal{K}_{Time}(t,t') \cdot \mathcal{K}_{Space}(x,x')$$

where $\mathcal{K}_{Time}$ is the kernel function for time patterns and $\mathcal{K}_{Space}$ is the kernel function for space patterns.

Spectral mixture kernel Wilson and Adams (2013) introduce a new kernel construction method based on Gaussian mixtures in the Fourier space. For that, they use the Bochner's theorem, which states that a real-valued function k defined on $\mathbb{R}^d$ is a covariance kernel of a stationary continuous random process if and only if it can be represented in the following way:

$$k(s) = \int_{\mathbb{R}^d} e^{2\pi i \lambda^\top s}\, \mu(\mathrm{d}\lambda)$$

where μ is a positive finite measure on $\mathbb{R}^d$. This theorem establishes equivalence between stationary covariance kernels[9] and their Fourier transform. It is a generalization of the classic one-dimensional spectral analysis when dealing with kernel functions instead of auto-covariance functions[10]. Indeed, the spectral density function $f_k(\lambda)$ is the Fourier transform of the covariance kernel function:

$$f_k(\lambda) = \int_{\mathbb{R}^d} k(s)\, e^{-2\pi i \lambda^\top s}\, \mathrm{d}s$$

whereas the covariance kernel function is the inverse Fourier transform of the spectral density function $f_k(\lambda)$:

$$k(s) = \int_{\mathbb{R}^d} f_k(\lambda)\, e^{2\pi i \lambda^\top s}\, \mathrm{d}\lambda$$

For instance, the SE kernel has a spectral density which is Gaussian. This leads Wilson and Adams (2013) to consider Gaussian mixtures of spectral densities to extend the SE kernel.

Let us define a mixture of n_m Gaussian densities on $\mathbb{R}^d$ with mean vectors $(\mu_1, \ldots, \mu_{n_m})$ and diagonal covariance matrices $(\Sigma_1, \ldots, \Sigma_{n_m})$. The corresponding density function $g(x)$ is defined by:

$$g(x) = \sum_{m=1}^{n_m} \omega_m \frac{1}{(2\pi)^{d/2}\sqrt{\det \Sigma_m}} \exp\left(-(x-\mu_m)^\top \Sigma_m^{-1}(x-\mu_m)\right) \tag{4}$$

[8] For example, we may consider daily prices of several assets.
[9] They verify $\mathcal{K}(x, x') = k(x - x')$.
[10] This is why it is essential that the covariance kernel is stationary.

where ω_m is the weight of the m^{th} Gaussian distribution. In our case, we are interested in real-valued covariance functions, implying that we replace $g(x)$ by $\frac{1}{2}(g(x)+g(-x))$. Interestingly, the inverse Fourier transform of Equation (4) is analytically tractable and is given by[11]:

$$k_{\text{SM}}(s) = \sum_{m=1}^{n_m} \omega_m \cos\left(2\pi s^\top \mu_m\right) \exp\left(-2\pi^2 s^\top \Sigma_m s\right)$$

Wilson and Adams (2013) show that the spectral mixture (SM) kernel can recover the usual kernels (squared exponential, Matérn, rational quadratic). Another interesting property is that it can learn negative covariances, which is essential when considering mean-reverting processes and contrarian trading strategies.

2.1.4 Hyperparameter selection

The covariance functions introduced before all have hyperparameters, such as length scales $\Sigma = \text{diag}\left(\ell_1^2, \ldots, \ell_d^2\right)$ in the squared exponential kernel, power α in the rational quadratic kernel, etc. All these parameters influence how the GP model can fit the observed data. This is why their choice is critical. They can be fixed ex-ante or we can estimate them.

For a given model, we denote by θ the parameters of the model. The usual way of selecting parameters is to maximize the likelihood function $L(\theta) = p(y \mid \theta)$. The underlying idea is to maximize the probability of the sample data y. In the case of the Gaussian process regression, $\theta = (\theta_{\mathcal{K}}, \sigma_\varepsilon)$ consists of the parameters $\theta_{\mathcal{K}}$ of the kernel function and the standard deviation σ_ε of the noise. Let $z = f(x)$ be the GP. We have:

$$p(y \mid \theta) = \int p(y \mid \theta, z)\, p(z \mid \theta)\, \mathrm{d}z$$

It is common to maximize the log marginal likelihood where we integrate out the latent values of z and solve:

$$\hat{\theta} = \arg\max_{\theta} \boldsymbol{\ell}(\theta)$$

where $\boldsymbol{\ell}(\theta) = \ln p(y \mid \theta)$. In the case of Gaussian noise, we have $Y \sim \mathcal{N}\left(\mathbf{0}_n, \mathcal{K}(\theta_{\mathcal{K}}) + \sigma_\varepsilon^2 I_n\right)$ where $\mathcal{K}(\theta_{\mathcal{K}})$ denotes the kernel matrix that depends on the kernel parameters $\theta_{\mathcal{K}}$. It follows that:

$$\boldsymbol{\ell}(\theta) = -\frac{n}{2}\ln(2\pi) - \frac{1}{2}\ln\left|\mathcal{K}(\theta_{\mathcal{K}}) + \sigma_\varepsilon^2 I_n\right| - \frac{1}{2} Y^\top \left(\mathcal{K}(\theta_{\mathcal{K}}) + \sigma_\varepsilon^2 I_n\right)^{-1} Y$$

This problem is usually solved using gradient-descent or quasi-Newton algorithms since it is possible to compute analytically the gradient of $\mathcal{K}(\theta_{\mathcal{K}})$. However, $\boldsymbol{\ell}(\theta)$ is not always convex, and may suffer local maxima (Duvenaud, 2013).

Let us illustrate the ML estimation of the hyperparameters with the periodic kernel. For that, we use 7 training points. In Figure 4, we report the posterior distribution $f(x^\star \mid x, y)$ when $x^\star$ ranges from -3 to $+3$. We assume that the hyperparameters of the kernel function are $\sigma = 1$, $\lambda_1 = 2$ and $\ell_1 = 1$ whereas the standard deviation of the noise σ_ε is set to 10^{-7}. Then, we estimate the parameters $\theta = (\sigma, \lambda_1, \ell_1, \sigma_\varepsilon)$ by the method of maximum likelihood. We obtain $\hat{\sigma} = 0.7657$, $\hat{\lambda}_1 = 1.6506$, $\hat{\ell}_1 = 0.7664$ and $\sigma_\varepsilon = 2.46 \times 10^{-7}$. The corresponding posterior distribution is given in Figure 5. As expected, we better fit the training set after the ML estimation than before.

Remark 5. *The Bayesian approach puts a prior distribution on the hyperparameters θ and marginalizes the posterior GP distribution over θ. However, this is not analytically*

[11] The correct formula is given in Wilson (2015).

Figure 4: Posterior distribution before marginal likelihood maximization

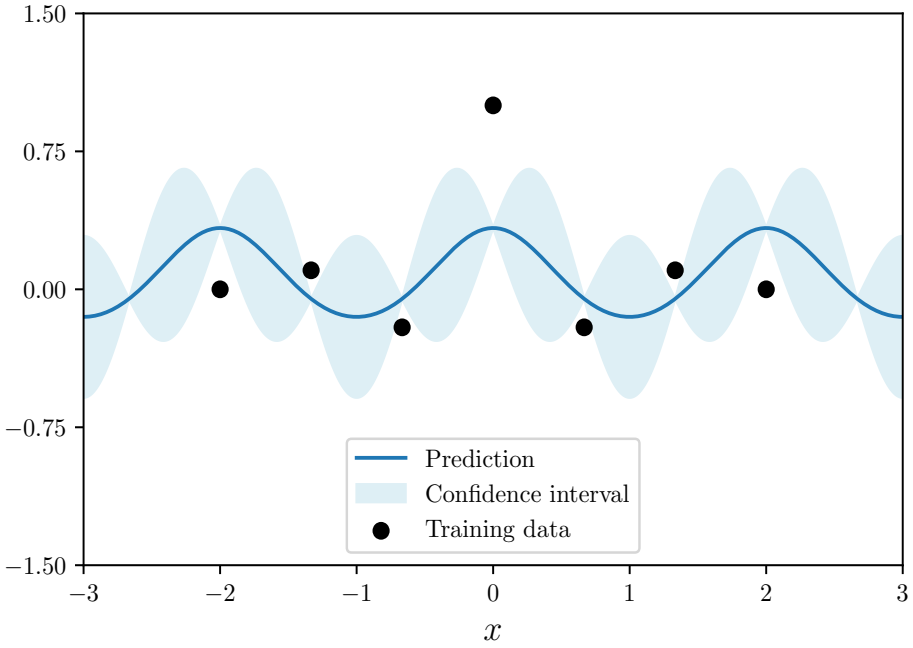
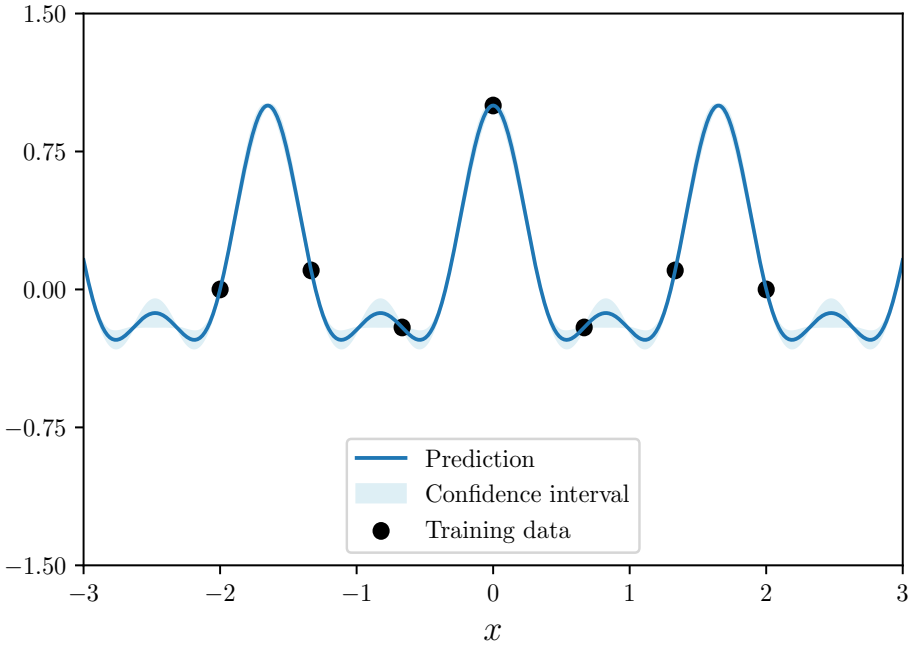


Figure 5: Posterior distribution after marginal likelihood maximization



tractable. We also notice that the posterior distribution of the GP is intractable if the noise is non-Gaussian. Both cases require to use Monte Carlo methods such as the Hamiltonian or Hybrid Monte Carlo (Neal, 2011), which is described in Appendix A.4 on page 36.

2.1.5 Classification

Gaussian processes regression can be extended for classification problems, where the output is a discrete variable corresponding to the class index. For example, we could want to predict the movements of asset prices: 1 for a positive return and 0 otherwise. In what follows, we consider the case of binary classification.

To model two classes with a GP prior over the data, one generally uses a sigmoid function[12] $g(x)$, for example the logistic function $\text{logit}(x) = (1 + e^{-x})^{-1}$. The output y is such that:

$$\Pr\{y = 1\} = g(f(x))$$

where $f(x)$ is the GP over $\mathcal{X}$. The predictive distribution for new inputs $x^\star$ can be marginalized over the latent GP values:

$$p(z^\star \mid y, z) = \int p(z^\star \mid z)\, p(z \mid y)\, \mathrm{d}z$$

where z and $z^\star$ respectively denote the random variables $f(x)$ and $f(x^\star)$. We deduce that:

$$\Pr\{y^\star = 1 \mid y\} = \int g(z^\star)\, p(z^\star \mid z)\, p(z \mid y)\, \mathrm{d}z^\star\, \mathrm{d}z \tag{5}$$

where the posterior distribution $p(z \mid y)$ can be written using Bayes rule:

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

Here, $p(z^\star \mid z)$ is the usual posterior distribution of the GP. However, the posterior distribution $p(z \mid y)$ is not easy to compute, and this is why approximations are used to evaluate the integral (5). Laplace approximation and expectation propagation are the two popular methods (Rasmussen and Williams, 2006). The first one approximates the posterior using a two-order Taylor expansion around its maximum, while the second one approximates the intractable probability distribution by minimizing the Kullback-Leibler divergence.

2.2 Bayesian optimization

Bayesian optimization is a black-box optimization method, meaning that little information is known about the objective function $f(x)$. Typically, Bayesian optimization is useful when the function is expensive to evaluate, its analytical expression is inaccessible or the gradient vector is not stable. This is the case with many complex machine learning problems where one would like to optimize the hyperparameters. For instance, the score of a deep neural network architecture is difficult to compute for a given set of hyperparameters (because training the model itself can take a long time), and it is impossible to compute the gradient vector with respect to each hyperparameter.

[12] This means that $g(x)$ is a monotonically increasing function in $[0, 1]$.

2.2.1 General principles

We are interested in finding the maximum of $f(x)$ on some bounded set $\mathcal{X}$. Bayesian optimization consists of two parts: (1) the '*probabilistic surrogate*' and (2) the '*acquisition function*' (or utility function). First of all, we build a prior probabilistic model for the objective function $f(x)$, and then update the probability distribution with samples drawn from $f(x)$ to get a posterior probability distribution. This approximation of the objective function is called a surrogate model. Gaussian processes are a popular surrogate model for Bayesian optimization because the GP posterior is still a multivariate normal distribution[13]. We then use a utility function based on this posterior probability distribution to choose a new point to evaluate the objective function at the next step. This utility function is called acquisition function. Intuitively, we consider the trade-off between exploitation and exploration. Exploitation means sampling where the surrogate model predicts a high objective gain and exploration means sampling where the prediction uncertainty is high. Therefore, the general idea of Bayesian optimization consists of the following steps:

1. Place a GP prior on the objective function $f(x)$.
2. Update the GP posterior probability distribution on $f(x)$ with all available samples.
3. Based on the acquisition function, decide where to make the next measurement.
4. Given this measurement, update the GP posterior probability distribution.
5. Repeat steps 2-4 until an approximated maximum of the objective function $f(x)$ is obtained (or stop after a predefined number of iterations).

2.2.2 Acquisition function

We assume that the function $f(x)$ has a Gaussian process prior and we observe samples of the form $\{(x_i, y_i)\}_{i=1}^n$. We have $y_i = f(x_i) + \varepsilon_i$ where $\varepsilon_i \sim \mathcal{N}\left(0, \sigma_\varepsilon^2\right)$ is the noise process. We denote by x and y the matrices $(x_1, \dots, x_n)$ and $(y_1, \dots, y_n)$. As shown previously, we can compute the posterior probability distribution $f(x^\star \mid x, y)$ for a new observation $x^\star$, and we have:

$$\hat{f}_n(x^\star) \sim \mathcal{N}\left(\hat{m}_n(x^\star), \hat{\mathcal{K}}_n(x^\star, x^\star)\right)$$

where:

$$\hat{m}_n(x^\star) = \mathcal{K}(x^\star, x)\left(\mathcal{K}(x, x) + \sigma_\varepsilon^2 I_n\right)^{-1} y$$

and:

$$\hat{\mathcal{K}}_n(x^\star, x^\star) = \mathcal{K}(x^\star, x^\star) - \mathcal{K}(x^\star, x)\left(\mathcal{K}(x, x) + \sigma_\varepsilon^2 I_n\right)^{-1}\mathcal{K}(x, x^\star)$$

The subscript n indicates that $\hat{f}_n$, $\hat{m}_n$ and $\hat{\mathcal{K}}_n$ depend on the sample of size n, which corresponds to the optimization step n. We note $\mathcal{D}_n$ the augmented data with the GP:

$$\mathcal{D}_n = \left\{\left(x_i, y_i, \hat{f}_i(x_i)\right)\right\}_{i=1}^n$$

Let $\mathcal{U}_n(x^\star)$ be the acquisition (or utility) function based on $\mathcal{D}_n$. The Bayesian optimization consists then in finding the new optimal point $x_{n+1} \in \mathcal{X}$ such that:

$$x_{n+1} = \arg\max \mathcal{U}_n(x^\star)$$

and updating the set of observations and the posterior distribution (see Algorithm 1).

[13] Other models exist such as random forests (Hutter *et al.*, 2011).

Algorithm 1 Bayesian optimization algorithm

The goal is to perform a Bayesian optimization
We initialize the data sample $\mathcal{D}_1$ and the vector θ_1 of hyperparameters
for $n = 1, 2, \ldots$ **do**
Find the optimal value $x_{n+1} \in \mathcal{X}$ of the utility maximization problem:

$$x_{n+1} = \arg\max \mathcal{U}_n \left(x^\star\right)$$

Update the data:

$$\mathcal{D}_{n+1} \leftarrow \mathcal{D}_n \cup \left\{ \left(x_{n+1}, y_{n+1}, \hat{f}_{n+1} \left(x_{n+1}\right) \right) \right\}$$

Update the hyperparameter vector θ_{n+1} of the kernel function
end for
return $\mathcal{D}_n$ and θ_n

Improvement-based acquisition function Let $f_n\left(\varkappa_n^\star\right)$ be the current optimal value among n samples drawn from $f\left(x\right)$:

$$\varkappa_n^\star = \arg\max_{\varkappa \in x} f\left(\varkappa\right)$$

where $\varkappa_n^\star$ is the point that maximizes the GP function over the first n steps. We would like to choose the next point x_{n+1} to be evaluated in order to improve this value. We define the improvement $\Delta_n\left(x^\star\right)$ as follows:

$$\begin{aligned} \Delta_n\left(x^\star\right) &= \left(\hat{f}_n\left(x^\star\right) - f_n\left(\varkappa_n^\star\right) \right)^+ \\ &= \max\left(\hat{f}_n\left(x^\star\right) - f_n\left(\varkappa_n^\star\right), 0 \right) \end{aligned}$$

The most intuitive strategy, as proposed by Kushner (1964), is to choose the point that maximizes the probability of a positive improvement:

$$\begin{aligned} \Pr\left\{\Delta_n\left(x^\star\right) > 0\right\} &= \Pr\left\{ \hat{f}_n\left(x^\star\right) > f_n\left(\varkappa_n^\star\right) \right\} \\ &= \Pr\left\{ \mathcal{N}\left(0,1\right) > \frac{f_n\left(\varkappa_n^\star\right) - \hat{m}_n\left(x^\star\right)}{\sqrt{\hat{\mathcal{K}}_n\left(x^\star, x^\star\right)}} \right\} \\ &= \Phi\left(\frac{\hat{m}_n\left(x^\star\right) - f_n\left(\varkappa_n^\star\right)}{\sqrt{\hat{\mathcal{K}}_n\left(x^\star, x^\star\right)}} \right) \end{aligned}$$

Since the probability of improvement fails to quantify the level of improvement, Močkus (1975) introduces an alternative acquisition function which takes into account the expected value of improvement (EI):

$$\mathrm{EI}_n\left(x^\star\right) = \mathbb{E}\left[\Delta_n\left(x^\star\right)\right]$$

In the GP framework, we obtain a closed form of the expected improvement acquisition function[14]:

$$\begin{aligned}\mathrm{EI}_n(x^\star) &= (\hat{m}_n(x^\star) - f_n(\varkappa_n^\star))\,\Phi\left(\frac{\hat{m}_n(x^\star) - f_n(\varkappa_n^\star)}{\sqrt{\hat{\mathcal{K}}_n(x^\star, x^\star)}}\right) + \\ &\quad \sqrt{\hat{\mathcal{K}}_n(x^\star, x^\star)}\phi\left(\frac{\hat{m}_n(x^\star) - f_n(\varkappa_n^\star)}{\sqrt{\hat{\mathcal{K}}_n(x^\star, x^\star)}}\right)\end{aligned}$$

It follows that $\mathrm{EI}_n(x^\star)$ and its derivatives are easy to evaluate, and we can use optimization algorithms such as quasi-Newton methods to find its maximum. Here, we have defined two utility functions $\mathcal{U}_n(x^\star) = \Pr\{\Delta_n(x^\star) > 0\}$ and $\mathcal{U}_n(x^\star) = \mathrm{EI}_n(x^\star)$ that are good candidates of acquisition functions. Applications of improvement-based acquisition functions are studied in Jones *et al.* (1998), Jones (2001), Brochu *et al.* (2010), and Shahriari *et al.* (2016), whereas the convergence of improvement-based optimization has been shown by Bull (2011). Appendix A.6 on page 38 extends the previous results to minimization problems.

Remark 6. *The previous approach can be generalized by considering a given threshold τ. In this case, we define the improvement by $\Delta_n(x^\star) = \left(\hat{f}_n(x^\star) - \tau\right)^+$. We have:*

$$\Pr\{\Delta_n(x^\star) > 0\} = \Phi\left(\frac{\hat{m}_n(x^\star) - \tau}{\sqrt{\hat{\mathcal{K}}_n(x^\star, x^\star)}}\right)$$

and:

$$\mathrm{EI}_n(x^\star) = (\hat{m}_n(x^\star) - \tau)\,\Phi\left(\frac{\hat{m}_n(x^\star) - \tau}{\sqrt{\hat{\mathcal{K}}_n(x^\star, x^\star)}}\right) + \sqrt{\hat{\mathcal{K}}_n(x^\star, x^\star)}\phi\left(\frac{\hat{m}_n(x^\star) - \tau}{\sqrt{\hat{\mathcal{K}}_n(x^\star, x^\star)}}\right)$$

Most of the time, the threshold τ is set to $f_n(\varkappa_n^\star) + \xi$ where $\xi > 0$.

In Figure 7, we illustrate the improvement-based optimization using the following minimization problem[15]:

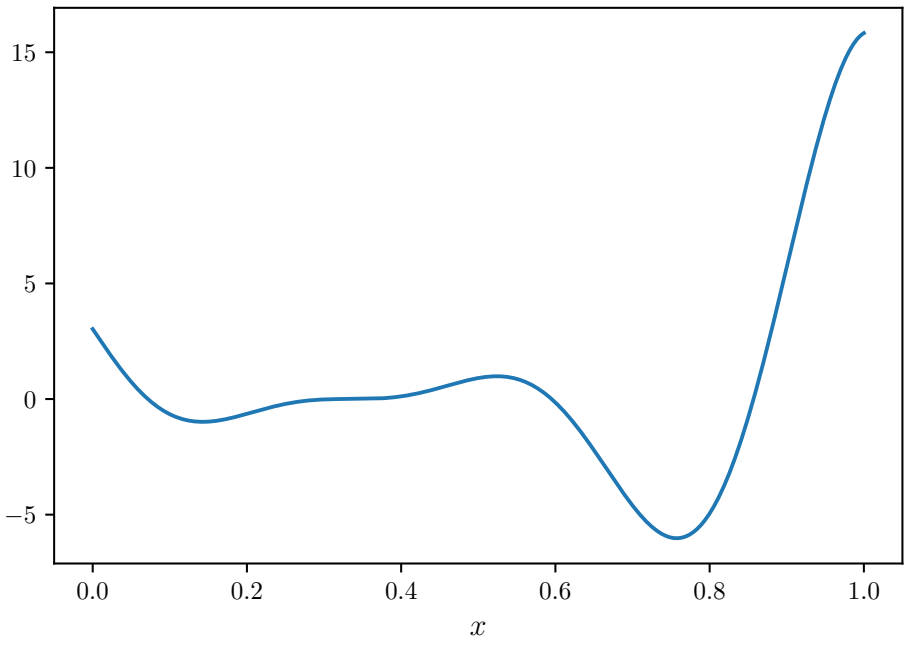
$$\min f(x) = (6x - 2)^2 \sin(12x - 4)$$

The objective function $f(x)$ is reported in Figure 6. In practice, we start with an initial design, usually consisting in measuring several random points of the domain. In the top/left panel in Figure 7, we start the algorithm with three initial points. We report the mean (blue solid line) and the confidence interval (blue shaded area) of the GP distribution. We also show the acquisition function $\mathcal{U}_n(x^\star) = \mathrm{EI}_n(x^\star)$ (red dashed line) and indicate the suggested next location by a vertical black line, which corresponds to the maximum of $\mathcal{U}_n(x^\star)$. The top/right panel corresponds to the second iteration where we have updated the sample. Indeed, the sample now contains the initial three points and the maximum point $x^\star$ obtained at the previous iteration. Then, we continue the process and show the results of the Bayesian optimization for the next five steps. We notice that steps $n = 3$, $n = 4$ and $n = 5$ correspond to an exploration stage (sampling where the variance is high), whereas step $n = 1$, $n = 2$ and $n = 6$ corresponds to an exploitation stage (sampling where the improvement is high). Finally, after six iterations, we have located the minimum since the acquisition function is equal to zero.

[14] See Appendix A.5 on page 37.

[15] This example is taken from Forrester *et al.* (2008).

Figure 6: Objective function of the minimization problem



Entropy-based acquisition function In this paragraph, we introduce two other acquisition functions based on differential entropy of information theory. Given random variables X and Y with continuous probability functions $p(x)$ and $p(y)$ and joint probability function $p(x, y)$, the differential (or Shannon) entropy $H(X)$ is equal to:

$$H(X) = -\int p(x) \ln p(x) \, \mathrm{d}x$$

while the conditional differential entropy $H(X \mid Y)$ of X is defined by:

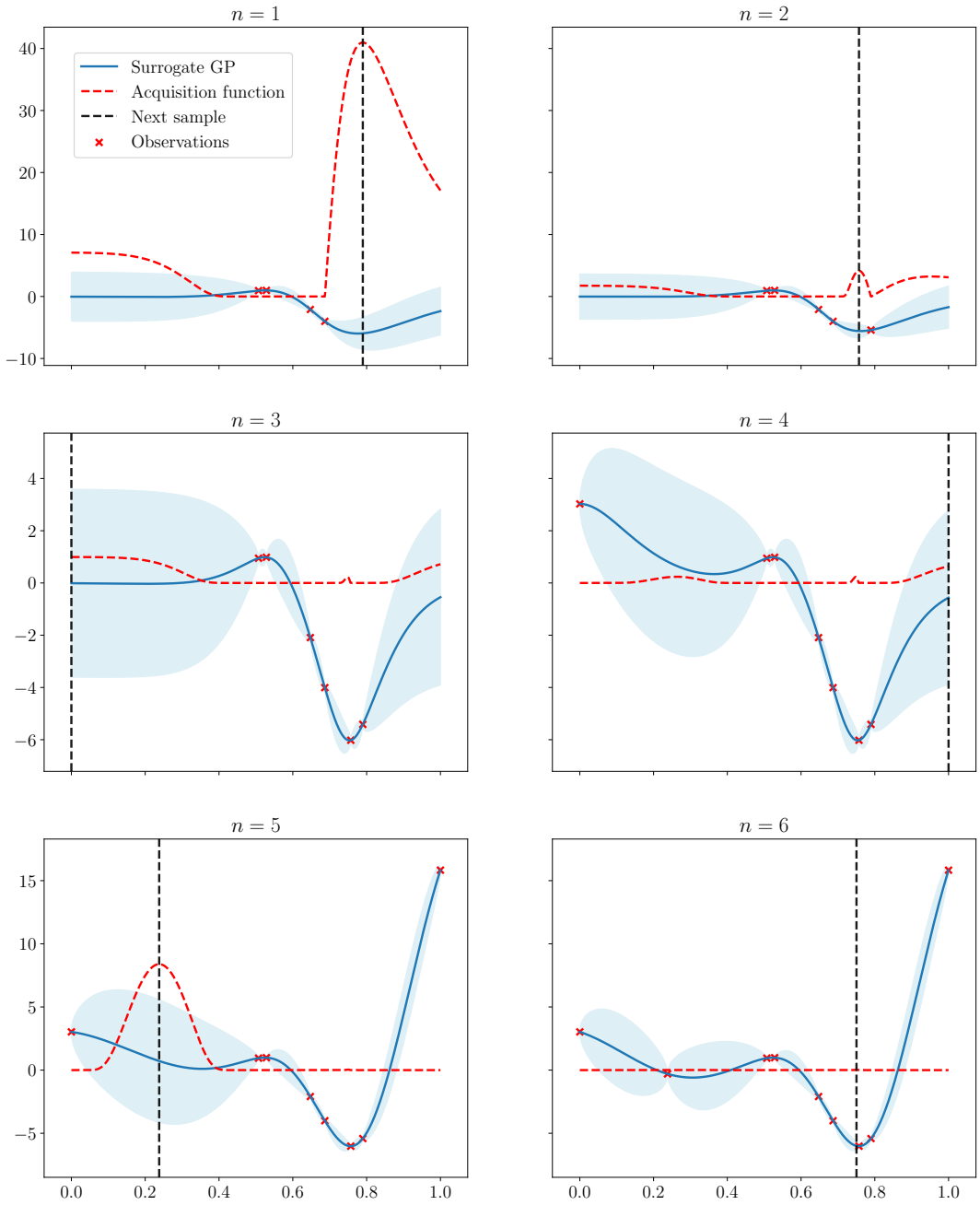
$$H(X \mid Y) = -\int \int p(x, y) \ln p(x \mid y) \, \mathrm{d}x \, \mathrm{d}y$$

Using Bayes theorem, we note that $H(X \mid Y)$ is the result of averaging $H(X \mid Y = y)$ over all possible values y of the random variable Y (Cover and Thomas, 2012).

We consider the location $x_{\max}$ of the global maximum of $f(x)$ as a random variable, which has the posterior distribution $\hat{p}_n(x_{\max}) = p(x_{\max} \mid x, y)$. Then, we can use the differential entropy to quantify the uncertainty of this point. The smaller the differential entropy, the lower the uncertainty. In order to have more certainty about the location of the global minimum, we want to choose the next point to evaluate x_{n+1} that implies the largest decrease in the differential entropy. For this purpose, we define the entropy search (ES) acquisition function as the difference between the current differential entropy of $\hat{p}_n(x_{\max})$ and the expected differential entropy of posterior probability distribution $p\left(x_{\max} \mid x, y, x^\star, \hat{f}_n(x^\star)\right)$ after adding a new sample $\left\{x^\star, \hat{f}_n(x^\star)\right\}$:

$$\mathrm{ES}_n(x^\star) = H(x_{\max}) - H\left(x_{\max} \mid \hat{f}_n(x^\star)\right)$$

Figure 7: Iterations of the Bayesian optimization



It follows that the next point is the solution of the maximization problem:

$$x_{n+1} = \arg\max_{x^\star \in \mathcal{X}} \mathrm{ES}_n\left(x^\star\right)$$

Although Henning and Schuler (2012) propose a method to approximate the above equation, Frazier (2018) indicates some difficulties in practice:

- $\hat{p}_n\left(x_{\max}\right)$ does not have always a closed-form expression;
- we need to compute the differential entropy of a large number of samples of $\left\{x^\star, \hat{f}_n\left(x^\star\right)\right\}$ to evaluate the expectation in the second term $H\left(x_{\max} \mid \hat{f}_n\left(x^\star\right)\right)$.

This is why Hernández-Lobato *et al.* (2014) propose an alternative approach called *predictive entropy search* (PES):

$$\mathrm{PES}_n\left(x^\star\right) = H\left(\hat{f}_n\left(x^\star\right)\right) - H\left(\hat{f}_n\left(x^\star\right) \mid x_{\max}\right)$$

Using the symmetric property of the mutual information, we can demonstrate that $\mathrm{PES}_n\left(x^\star\right)$ and $\mathrm{ES}_n\left(x^\star\right)$ are equivalent acquisition functions[16]. In the case of the PES acquisition function, we can compute a closed-form expression for $H\left(\hat{f}_n\left(x^\star\right)\right)$ and Hernández-Lobato *et al.* (2014) uses the expectation propagation method (Minka, 2001) to find an approximation of $H\left(\hat{f}_n\left(x^\star\right) \mid x_{\max}\right)$. Therefore, we can find the maximum of the PES acquisition function by a simulation approach.

Knowledge gradient-based acquisition function The knowledge gradient (KG) acquisition function is closed to the expected improvement. It was first introduced in Frazier *et al.* (2009) for finite discrete decision spaces before Scott *et al.* (2011) extended it to Gaussian processes. The main difference between KG and EI acquisition functions is that KG accounts for noise and does not restrict the final solution to a previously evaluated point, meaning that it can return any point of the domain and not only one observed point.

Suppose that we have observed the sample $\{(x_i, y_i)\}_{i=1}^n$. As previously, we compute the posterior probability distribution:

$$\hat{f}_n\left(x^\star\right) \sim \mathcal{N}\left(\hat{m}_n\left(x^\star\right), \hat{\mathcal{K}}_n\left(x^\star, x^\star\right)\right)$$

Under risk-neutrality assumption (Berger, 2013), we value a random outcome according to its expected value $\hat{m}_n\left(x^\star\right)$. To maximize the objective function, we can try to maximize $\hat{m}_n\left(x^\star\right)$ and we note $\hat{m}_n\left(\varkappa_n^\star\right) = \max_\varkappa \hat{m}_n\left(\varkappa\right)$. If we consider one supplementary point, we could also

[16] We have:

$$\begin{aligned}
\mathrm{ES}_n\left(x^\star\right) &= I\left(x_{\max}, \hat{f}_n\left(x^\star\right)\right) \\
&= I\left(\hat{f}_n\left(x^\star\right), x_{\max}\right) \\
&= \mathrm{PES}_n\left(x^\star\right)
\end{aligned}$$

where $I\left(X, Y\right)$ is the mutual information of two continuous random variables:

$$\begin{aligned}
I\left(X, Y\right) &= \int\int p\left(x, y\right) \ln \frac{p\left(x, y\right)}{p\left(x\right) p\left(y\right)} \,\mathrm{d}x \,\mathrm{d}y \\
&= H\left(X\right) - H\left(X, Y\right)
\end{aligned}$$

compute the new posterior distribution for f with conditional expected value $\hat{m}_{n+1}(x^\star)$. The idea is then to choose a new point that maximizes the increment in conditioned expectation gained from sampling this point. For example, we can maximize the expected value of the difference, which is called *knowledge gradient*:

$$\mathrm{KG}_n(x^\star) = \mathbb{E}\left[\hat{m}_{n+1}(x^\star) - \hat{m}_n(\varkappa_n^\star)\right]$$

We have:

$$x_{n+1} = \arg\max_{x^\star \in \mathcal{X}} \mathrm{KG}_n(x^\star)$$

The solution can then be obtained via simulation using Algorithm 2 formulated by Frazier (2018).

Algorithm 2 Simulation-based computation of $\mathrm{KG}_n(x^\star)$

$x^\star$ is the input parameter
n_s is the number of simulations
We note $\hat{m}_n(\varkappa_n^\star) = \max_\varkappa \hat{m}_n(\varkappa)$
for $s = 1 : n_s$ **do**
  Generate $y_{(s)}^\star \sim \mathcal{N}\left(\hat{m}_n(x^\star), \hat{\mathcal{K}}_n(x^\star, x^\star)\right)$
  Add the simulated point $(x^\star, y_{(s)}^\star)$ to the current sample (x, y)
  Compute $\hat{m}_{n+1}\left(\varkappa_{(s)}\right) = \max_{\varkappa \in \mathcal{X}} \hat{m}_{n+1}(\varkappa)$ where $\hat{m}_{n+1}(\varkappa)$ is the posterior mean that depends on $(x^\star, y_{(s)}^\star)$
  $\Delta_{(s)} \longleftarrow \hat{m}_{n+1}\left(\varkappa_{(s)}\right) - \hat{m}_n(\varkappa_n^\star)$
end for
return $\mathrm{KG}_n(x^\star) \leftarrow n_s^{-1} \sum_{s=1}^{n_s} \Delta_{(s)}$

Remark 7. *In the noise-free case and if the final solution is limited to the previous sampling, the KG acquisition function reduces to the EI acquisition function:*

$$\begin{aligned} \hat{m}_{n+1}(x^\star) - \hat{m}_n(\varkappa^\star) &= \max\left(f_n(\varkappa_n^\star), \hat{f}_n(x^\star)\right) - f_n(\varkappa_n^\star) \\ &= \left(\hat{f}_n(x^\star) - f_n(\varkappa_n^\star)\right)^+ \end{aligned}$$

3 Financial applications

In this section, we use Gaussian processes to model and forecast the yield curve and Bayesian optimization to build an online trend-following strategy.

3.1 Yield curve modeling

To illustrate the potential of GP methods in finance, we first consider the fitting of the U.S. yield curve. One of the most used models is the Nelson-Siegel parametric model, for which problems have been reported regarding parameter estimation and their large variations over time (Annaert *et al.*, 2013). GP can be thought of as a Bayesian nonparametric alternative. We display in Figures 8 and 9 the fitting of the yield curve corresponding to two different dates and presenting different shapes[17].

[17] The covariance kernel is $\mathcal{K}_{\mathrm{SE}} \cdot \mathcal{K}_{\mathrm{EXP}} + \mathcal{K}_{\mathrm{RQ}}$, where the exponential kernel $\mathcal{K}_{\mathrm{EXP}}$ is equal to:

$$\mathcal{K}_{\mathrm{EXP}}(x, x') = \sigma^2 \exp\left(-\frac{\|x - x'\|_2}{2\ell}\right)$$

Figure 8: GP fitting of the yield curve (June 2007)

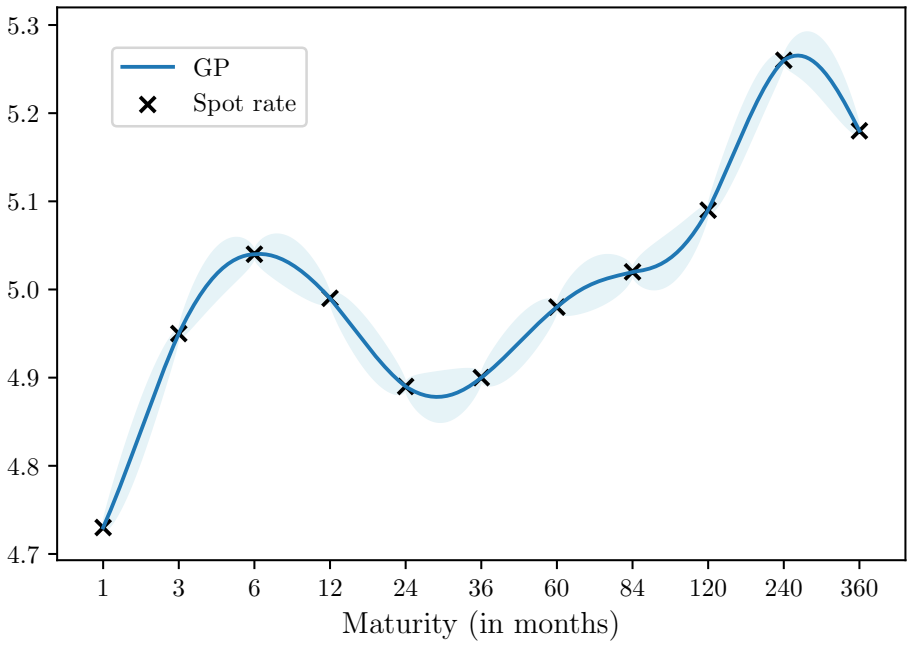
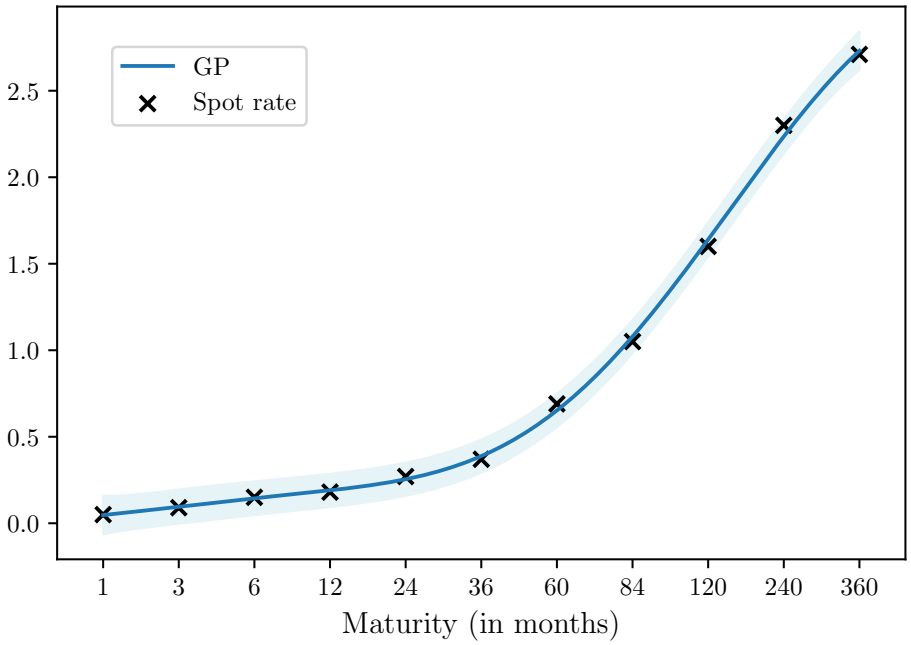


Figure 9: GP fitting of the yield curve (June 2012)



We now try GP-based methods to forecast movements of the U.S. yield curve. It is a classical macroeconomic factor that can be used as a signal in the forecasting of equity and bond returns (Rebonato, 2015; Cochrane *et al.* (2005)) and thus is of practical interest in quantitative asset management. Several approaches exist in time-series prediction with Gaussian processes. In this paper, we mainly focus on the GP-ARX model, which is an application of ARX models in the GP framework. The ARX model assumes a nonlinear relationship between the time-series Y_t and its previous values plus some exogenous factors X_t:

$$Y_t = f\left(Y_{t-1}, Y_{t-2}, \dots, X_{t-1}, X_{t-2}, \dots\right) + \varepsilon_t$$

where $\varepsilon_t \sim \mathcal{N}\left(0, \sigma_\varepsilon^2\right)$ is a white noise process. The main idea of GP-ARX is to use a Gaussian process surrogate for the function $f \sim \mathcal{GP}\left(\mathbf{0}, \mathcal{K}\right)$. Once the kernel $\mathcal{K}$ is chosen, the training set simply consists in observations of X_t and Y_t. Inference of hyperparameters is done with the method of maximum likelihood as explained in the previous section. Chandorkar *et al.* (2017) use this model to forecast weather data and note the importance of the "*persistence prediction*" in model building. Persistence prediction is taking the current value as a forecast for tomorrow: $\hat{Y}_{t+1} = Y_t$. The persistence model assumes then that the process is a random walk. For time-series which exhibit memory effects, this trivial forecast yields very good results if we use usual accuracy measures, such as mean squared error. One way to measure memory in time-series is to compute the Hurst exponent, which measures the long-term autocorrelation. A Hurst exponent $H > \frac{1}{2}$ indicates positive long-term autocorrelation while $H < \frac{1}{2}$ is the opposite[18]. The Hurst exponent is related to the fractal dimension of the time-series, and can be an indicator of the predictability of the time-series (Kroha and Škoula, 2018).

Table 1: Hurst exponent of U.S. spot rates

Maturity	1M	3M	6M	1Y	2Y	5Y	10Y	20Y	30Y
Hurst	0.40	0.50	0.61	0.62	0.57	0.51	0.49	0.48	0.50

Our dataset consists in daily zero-coupon yield for the following maturities: 1, 3 and 6 months, 1, 2, 5, 10, 20 and 30 years. Table 1 shows the Hurst exponent for the spot rates of different maturities. In Figures 10 and 11, we report the one-day ahead rolling prediction during 2016, and the 95% confidence interval. We also forecast the 2Y and 10Y spot rates by using for each maturity a lag of one for the GP-ARX model, and the three-month spot rate and its first lag for the exogenous variable. Both spot rates exhibit persistence as it can be seen in Table 2. We notice that the three methods are equivalent in this simplistic example. However, the GP-ARX method is able to estimate the confidence interval, and this metric can be used as a trading signal.

Table 2: Root mean squared error (in %)

Spot Rate	Persistence	ARX	GP-ARX
2Y	3.33	3.33	**3.32**
10Y	**4.40**	4.53	4.45

Remark 8. *The choice of the kernel function and the features are primordial. The ARX model is actually equivalent to the GP-ARX model, when the kernel is linear. In our GP-ARX model, we used a sum of exponential and linear kernels and only used simple time-series*

[18] A Brownian motion has a Hurst exponent of exactly $\frac{1}{2}$ and is memory-less.

Figure 10: Prediction of the 2-year spot rate

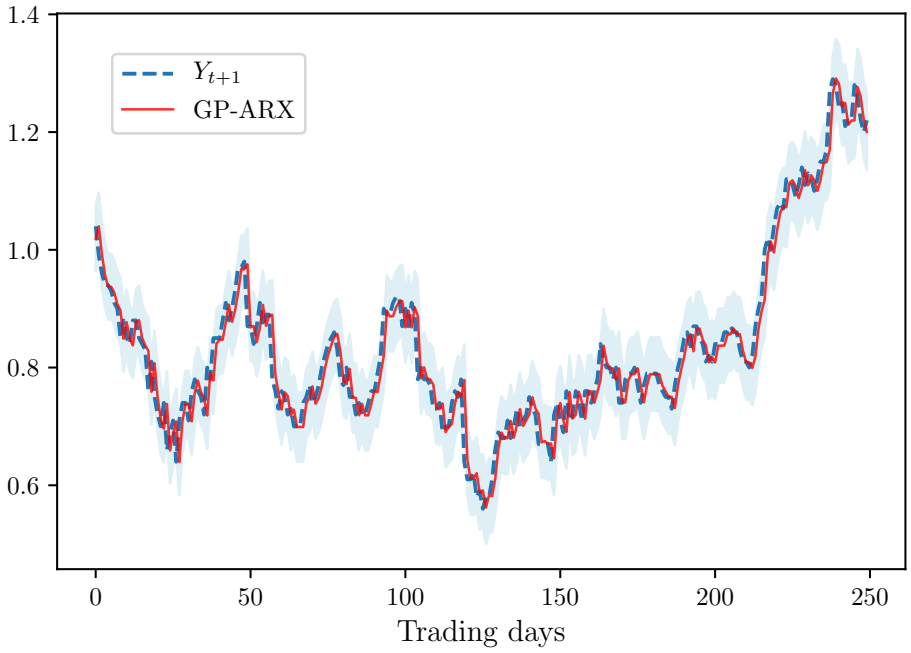
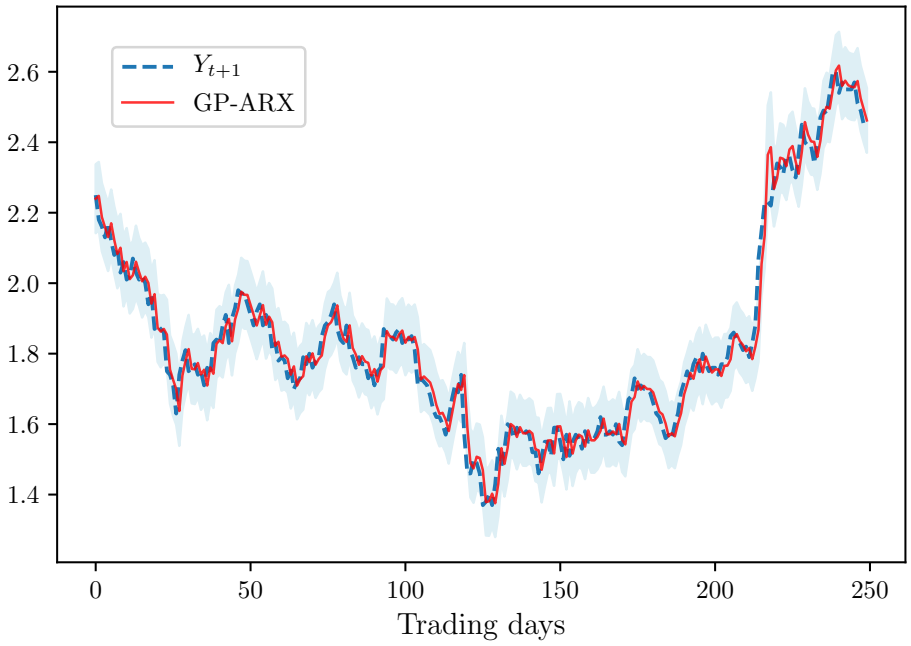


Figure 11: Prediction of the 10-year spot rate



features. Thanks to the ability to capture different length scales and patterns with the choice of the kernel, the GP approach can be used as good interpolators for a wide range of financial applications by finding the suitable kernel combination.

There is a growing interest in using Student-t distribution instead of Gaussian distribution in GP regression (Shah *et al.*, 2014). More specifically, Chen *et al.* (2014) introduce a framework for multivariate Gaussian and Student-t process regression, and use it for stock and equity index predictions. The problem of yield curve forecasting is intrinsically multivariate. We are trying to predict a vector of interest rates for different maturities, that are highly correlated and dependent. Instead of treating each output separately, Student-t multivariate process regression works with a dataset consisting of full yield curve observations as inputs and outputs at the same time, while taking into account output correlations. We recall in Appendix A.7 on page 38 the definition and some useful properties of multivariate and matrix-variate Student-t distributions. Specifically, as for the Gaussian case, the posterior distribution is still a matrix-variate Student-t distribution, which makes computations of inference step and maximum marginal likelihood tractable. This motivates the definition of the Student-t processes (TP). A collection of random vectors is a TP if and only if any finite number of them has a joint multivariate Student-t distribution. We have tested TP-ARX in place of GP-ARX. Unfortunately, we have not found better forecasting values. This result is disappointing since we may think that one of the issues in yield curve modeling is the cross-section correlation of spot rates[19] and the possible fat tails that we observe in fixed-income assets.

3.2 Portfolio optimization

We now consider an application of Bayesian optimization to portfolio optimization in the context of quantitative asset management. We first describe the asset allocation problem, which is a trend-following strategy, then we show how to solve it and finally we use Bayesian optimization with a squared exponential kernel to find optimal hyperparameters of the trend-following strategy.

3.2.1 The trend-following strategy

We consider a universe of n assets for which we observe daily prices and we look for an optimal portfolio, that is an allocation vector $x \in \mathbb{R}^n$ that balances risk and return. If we can predict the vector μ of expected returns and compute the covariance matrix Σ of asset returns, then the regularized Markowitz optimization problem (Roncalli, 2013; Bourgeron *et al.*, 2018) is the following:

$$x^\star(\gamma) = \arg\min_x \frac{1}{2} x^\top \Sigma x - \gamma \mu^\top x + \lambda \|x - x_0\|_2^2$$

where γ is the inverse of the risk-aversion coefficient, λ is the ridge regularization parameter and x_0 is a reference portfolio. We consider a simple version of the trend-following strategy:

- The expected returns are computed using a moving-average estimator. Let $P_{i,t}$ be the daily price of Asset i. We have:

$$\mu_{i,t} = \frac{P_{i,t}}{P_{i,t-\ell(\mu)}} - 1$$

[19] TPs are especially designed to take into account both cross-section and time-series correlations, whereas GPs can only consider the dynamics of one direction (cross-section or time-series), not both (See Equations (7) and (15) in Chen *et al* (2018)).

where $\ell(\mu)$ is the window length of the MA estimator.

- The covariance matrix is estimated using the empirical estimator, the window length of which is denoted by $\ell(\Sigma)$.
- The portfolio is rebalanced at fixed dates t, for example on a monthly or weekly basis.

Let x_t be the optimal portfolio at the rebalancing date t. The allocation problem is given by:

$$\begin{aligned} x_t(\lambda) &= \arg\min_x -\mu_t^\top x + \lambda \|x - x_{t-1}\|_2^2 \\ &\text{s.t.} \quad \sigma_t(x) \leq \bar{\sigma} \end{aligned} \tag{6}$$

where μ_t is the estimated vector of expected returns at time t, $\sigma_t(x) = \sqrt{x^\top \Sigma_t x}$ is the portfolio volatility estimated at time t, $\bar{\sigma}$ is the target volatility of the trend-following strategy. To solve this convex problem, we use the ADMM algorithm given in Appendix A.8 on page 40 (Boyd *et al.*, 2011). Following Bourgeron *et al.* (2018) and Richard and Roncalli (2019), it is natural to write the previous problem as follows:

$$\begin{aligned} x_t &= \arg\min_x -\mu_t^\top x + \lambda \|x - x_{t-1}\|_2^2 + \mathbb{1}_\Omega(z) \\ &\text{s.t.} \quad x - z = 0 \end{aligned}$$

where $\Omega = \left\{ z \in \mathbb{R}^n : \left\| z^\top \Sigma_t z \right\|_2^2 \leq \bar{\sigma}^2 \right\}$. However, we improve the ADMM algorithm by introducing the Cholesky trick:

$$\begin{aligned} x_t &= \arg\min_x -\mu_t^\top x + \lambda \|x - x_{t-1}\|_2^2 + \mathbb{1}_\Omega(z) \\ &\text{s.t.} \quad -L_t x + z = 0 \end{aligned}$$

where $\Omega = \left\{ z \in \mathbb{R}^n : \|z\|_2^2 \leq \bar{\sigma}^2 \right\}$ and L_t is the upper Cholesky decomposition matrix of Σ_t. It follows that $z = L_t x$ and:

$$\begin{aligned} \|z\|_2^2 &= z^\top z \\ &= x^\top L_t^\top L_t x \\ &= x^\top \Sigma_t x \\ &= \sigma_t^2(x) \end{aligned}$$

In fact, our experience shows that the Cholesky trick helps to accelerate the convergence of the ADMM algorithm with respect to the formulation of Bourgeron *et al.* (2018) and Richard and Roncalli (2019). Finally, it follows that the ADMM algorithm becomes:

$$\begin{aligned} x^{(k+1)} &= \arg\min -\mu_t^\top x + \lambda \|x - x_{t-1}\|_2^2 + \frac{\varphi}{2} \left\| -L_t x + z^{(k)} + u^{(k)} \right\|_2^2 \\ z^{(k+1)} &= \arg\min \mathbb{1}_\Omega(z) + \frac{\varphi}{2} \left\| -L_t x^{(k+1)} + z + u^{(k)} \right\|_2^2 \\ u^{(k+1)} &= u^{(k)} - L_t x^{(k+1)} + z^{(k+1)} \end{aligned}$$

We notice that the z-step corresponds to a simple projection on the Euclidean ball of center 0 and radius $\bar{\sigma}$ of the vector $L_t x^{(k+1)} - u^{(k)}$, and the computation of the proximal operator is straightforward. The x-step corresponds to a linear system. If we define $f^{(k)}(x)$ as follows:

$$f^{(k)}(x) = -\mu_t^\top x + \lambda \|x - x_{t-1}\|_2^2 + \frac{\varphi}{2} \left\| -L_t x + z^{(k)} + u^{(k)} \right\|_2^2$$

we deduce that:

$$\begin{aligned}\nabla f^{(k)}(x) &= -\mu_t + \lambda x - \lambda x_{t-1} + \varphi L_t^\top L_t x - \varphi L_t^\top \left(z^{(k)} + u^{(k)}\right) \\ &= -\mu_t + \lambda x - \lambda x_{t-1} + \varphi \Sigma_t x - \varphi L_t^\top \left(z^{(k)} + u^{(k)}\right)\end{aligned}$$

Finally, we obtain the following solution:

$$x^{(k+1)} = (\varphi \Sigma_t + \lambda I_n)^{-1} \left(\mu_t + \lambda x_{t-1} + \varphi L_t^\top \left(z^{(k)} + u^{(k)}\right)\right)$$

3.2.2 Hyperparameter estimation of the trend-following strategy

The trend-following strategy depends on three hyperparameters:

1. the parameter λ that controls the turnover between two rebalancing dates;
2. the window length $\ell(\mu)$ that controls the estimation of trends;
3. the horizon time $\ell(\Sigma)$ that measures the risk of the assets.

Traditionally, the trend-following strategy is implemented by considering that these hyperparameters are fixed. By construction, their choice has a big impact on the strategy design. For instance, a small value of $\ell(\mu)$ will catch short momentum, whereas a large value of $\ell(\mu)$ will look for more persistent trends. This hyperparameter is then key for distinguishing short-term and long-term CTAs.

In fact, the right asset allocation problem is not given by Equation (6), but is defined as follows:

$$\begin{aligned}x_t(\lambda_t, \ell_t(\mu), \ell_t(\Sigma)) &= \arg\min_x -\mu_t^\top x + \lambda_t \|x - x_{t-1}\|_2^2 \\ &\text{s.t.} \quad \sigma_t(x) \leq \bar{\sigma}\end{aligned} \tag{7}$$

This means that the hyperparameters are not fixed and must be estimated at each rebalancing date. In the previous framework, the estimation consists in finding x_t given that λ, $\ell(\mu)$ and $\ell(\Sigma)$ are constant. In our framework, the estimation consists in finding the optimal portfolio x_t, but also the optimal parameters λ_t, $\ell_t(\mu)$ and $\ell_t(\Sigma)$. This can be done using the Bayesian optimization framework.

By nature, the parameters $\ell(\mu)$ and $\ell(\Sigma)$ are discrete and generally expressed in months, e.g. $\ell(\mu) \in \{3, 6, 12, 24\}$ and $\ell(\Sigma) \in \{3, 6, 12\}$. Discrete, integer or categorical parameters are not easy to manage in Bayesian optimization since Gaussian processes (or random forests), which serve as surrogates for the black-box function, are not adapted. Since no standard approach exists yet[20], we use a simple method which consists in using continuous variables in the Bayesian optimization step while flooring the hyperparameters $\ell(\mu)$ and $\ell(\Sigma)$ to the nearest integer when computing the objective function given by Equation (7).

The choice of the objective function is the main step when implementing Bayesian optimization. In a classical machine learning problem, the objective function can be the cross-validation score of the hyperparameters, in order to reduce the risk of overfitting. Defining an objective for a quantitative strategy is less clear and prone to overfitting. One simple and obvious function is the Sharpe ratio of the strategy. Each rebalancing date, we run a Bayesian optimization to look for hyperparameters which maximizes the historical Sharpe ratio over a given period. A more robust objective function is the minimum of the rolling

[20] Although some new methods are emerging (Garrido-Merchán and Hernández-Lobato, 2017).

Sharpe ratio in order to reduce the overfitting bias. For example, we can compute the rolling six-month Sharpe ratio $\text{SR}_\tau\left(\lambda, \ell\left(\mu\right), \ell\left(\Sigma\right)\right)$ for a backtest with fixed hyperparameters and a period $[\tau - 0.5; \tau]$ and use Bayesian optimization to solve:

$$\left\{\lambda_t, \ell_t\left(\mu\right), \ell_t\left(\Sigma\right)\right\} = \arg\max \left\{ \min_{\tau \in [t-2, t[} \text{SR}_\tau\left(\lambda, \ell\left(\mu\right), \ell\left(\Sigma\right)\right) \right\}$$

However, in order to benefit from the exploration of the parameters space by Bayesian optimization, we use another approach. Since volatility is already controlled via the portfolio optimization constraint and regularization, we prefer to choose the return of the strategy over a two-year historical period for the objective function:

$$\left\{\lambda_t, \ell_t\left(\mu\right), \ell_t\left(\Sigma\right)\right\} = \arg\max \hat{\mu}_t\left(\lambda, \ell\left(\mu\right), \ell\left(\Sigma\right)\right)$$

where $\hat{\mu}_t\left(\lambda, \ell\left(\mu\right), \ell\left(\Sigma\right)\right)$ is the performance of the backtest for the period $[t-2; t]$. We keep track of all samples tested during Bayesian optimization, sort them according to their objective function and select the best three sets of hyperparameters to compute three different optimal weights that are averaged to form the final portfolio. This approach considerably reduces the overfitting bias.

3.2.3 An example

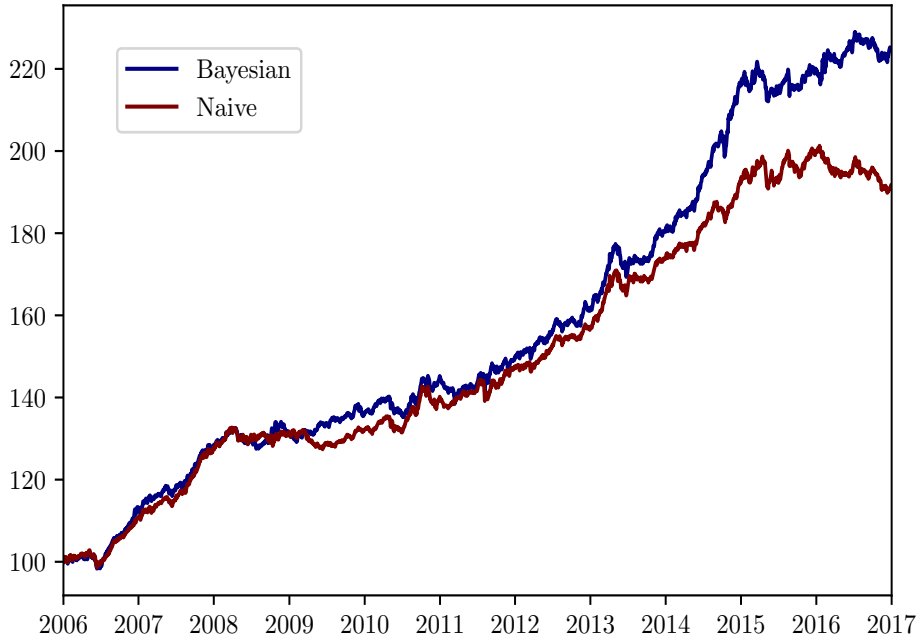
Our dataset consists of daily prices of 13 futures contracts on world-wide equity indices such as the S&P 500 and Eurostoxx indices and 10Y sovereign bonds from 2006 to 2017. The Bayesian optimization strategy described in the previous paragraph is compared to the basic one-year trend-following strategy where parameters remain unchanged through time: λ_t is set to 10% while $\ell_t\left(\mu\right)$ and $\ell_t\left(\Sigma\right)$ are equal to 12 months.

Table 3: Backtest results (2006 – 2016)

Strategy	Sharpe ratio	Return	Volatility	MDD
Naive	1.56	5.9%	3.7%	-5.7%
Bayesian	1.71	7.4%	4.3%	-4.7%

The cumulative performance of the strategies is shown in Figure 12, whereas Table 3 shows the results of the two strategies. We notice that Bayesian optimization is able to improve the annualized Sharpe ratio from 1.56 to 1.71 and reduce drawdowns. However, we do not believe that this result is important, because it is just a backtest. More interesting are the dynamics of the hyperparameters estimated by the Bayesian optimization. On page 41, we report the dynamics of λ_t (Figure 14), $\ell_t\left(\mu\right)$ (Figure 15) and $\ell_t\left(\Sigma\right)$ (Figure 16), whereas the statistics are reported in Table 4. We observe that λ_t moves relatively fast. At the beginning of the 2008 Global Financial Crisis, it is reduced implying a more reactive allocation. At the end of the 2008 crisis, we observe the opposite effect. λ_t is increased and the allocation becomes less reactive. However, this must be compared with the dynamics of $\ell_t\left(\mu\right)$ and $\ell_t\left(\Sigma\right)$. Most of the time, the optimal window $\ell_t\left(\mu\right)$ is high and is equal to 18 months on average. However, during and after the GFC, $\ell_t\left(\mu\right)$ is dramatically reduced. The parallel can be done with the performance of short-term and long-term CTAs. On average, long-term CTAs outperform short-term CTAs, but during some periods, short-term CTAs can do a very good job, and post an incredible performance while long-term CTAs have a strong negative performance. Concerning $\ell_t\left(\Sigma\right)$, the results indicate that a short-term window is better, while the market practice is to consider long-term window (typically a

Figure 12: Cumulative performance of trend-following strategies



one-year empirical covariance matrix). In fact, there is a trade-off between the regularization parameter λ_t and the covariance window $\ell_t(\Sigma)$. Our model chooses a short-term covariance, because it can control the turnover thanks to the ridge parameter.

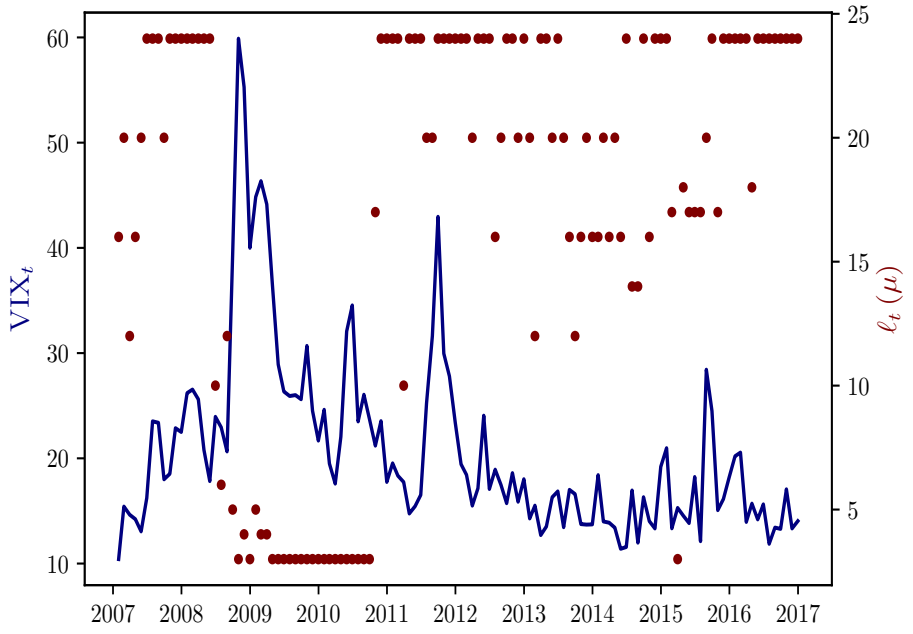
Table 4: Statistics of optimal hyperparameters (2006 – 2016)

θ_t	$\min \theta_t$	$\max \theta_t$	$\bar{\theta}_t$	$\sigma(\theta_t)$	$\rho(\theta_t, \text{VIX}_t)$
λ_t	0.01	2	0.13	0.056	48%
$\ell_t(\mu)$	3.00	24	17.45	8.10	−49%
$\ell_t(\Sigma)$	3.00	12	4.13	1.72	38%

The dynamics of these parameters can be analyzed with respect to market volatility. For instance, if we compute the correlation with the VIX index, we observe that the regularization hyperparameter λ_t and the covariance window $\ell_t(\Sigma)$ show a positive correlation with VIX_t while the correlation is negative between $\ell_t(\mu)$ and VIX_t (see Table 4). This indicates that the strategy focuses on short-term momentum and takes less risk in times of high volatility as it is shown in Figure 13 during 2008. The hyperparameter λ_t, which expresses a turnover penalty, then has a conservative effect during the 2008 Global Financial Crisis.

Remark 9. *Because of local minima (since the objective function might not be well-behaved or even regular when considering categorical variables), periods of instability occur in the estimation of the optimal parameters. This is the case in 2013-2014 (Figure 13). In such cases, we benefit from diversification when mixing the three optimal portfolios unlike when selecting only one solution.*

Figure 13: Comparison of $\ell_t(\mu)$ and VIX_t



4 Conclusion

In this paper, we explore the use of Gaussian processes and Bayesian optimization in finance. Two applications have been considered: the yield curve modeling and the online calibration of trend-following strategies. Our results show that GPs are a powerful tool for fitting the yield curve. Therefore, GPs can be used as a semi-parametric alternative of popular parametric approaches such as the Nelson-Siegel model. However, our results also show that GPs are equivalent to traditional econometric approaches for forecasting interest rates, but they do not do a better job. The case of trend-following strategies is more interesting, because it is a classic problem in finance when we choose ex-ante the value of hyperparameters. Until now, there was no other alternative approach to test several combinations of hyperparameters, and to choose the best combination in trying to avoid the in-sample bias, which is inherent to any backtesting protocol. We show how to implement a Bayesian optimization for estimating the window lengths of the trend vector and the covariance matrix. Results confirm the practice and what we observe in the industry of CTA and dynamic risk parity funds. Generally, it is better to consider a long window for the expected returns and a short window for the risks. However, there is a trade-off between performance, turnover and rebalancing costs. This is why it is necessary to introduce penalty functions in the portfolio optimization. Our results also show that there are some periods where reducing the window of trends may add value. In this context, the Bayesian optimization provides a normative way to build online trend following strategies.

References

[1] Alvarez, M.A., Rosasco, L., and Lawrence, N.D. (2012), Kernels for Vector-valued Functions: A Review, *Foundations and Trends® in Machine learning*, 4(3), pp. 195-266.

[2] Annaert, J., Claes, A.G., De Ceuster, M.J., and Zhang, H. (2013), Estimating the Spot Rate Curve Using the Nelson-Siegel model: A Ridge Regression Approach, *International Review of Economics and Finance*, 27, pp. 482-496.

[3] Berger, J.O. (2013), *Statistical Decision Theory and Bayesian Analysis*, Springer.

[4] Bishop, C.M. (2006), *Pattern Recognition and Machine Learning*, Information Science and Statistics, Springer.

[5] Bochner, S. (1959), *Lectures on Fourier Integrals*, Annals of Mathematics Studies, 42, Princeton University Press.

[6] Bourgeron, T., Lezmi, E., and Roncalli, T. (2018), Robust Asset Allocation for Robo-Advisors, *SSRN*, `www.ssrn.com/abstract=3261635`.

[7] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011), Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers, *Foundations and Trends® in Machine learning*, 3(1), pp. 1-122.

[8] Brochu, E., Cora, V.M., and De Freitas, N. (2010). A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning, *arXiv*, 1012.2599.

[9] Bull, A.D. (2011), Convergence Rates of Efficient Global Optimization Algorithms, *Journal of Machine Learning Research*, 12, pp. 2879-2904.

[10] Chandorkar, M., Camporeale, E., and Wing, S. (2017), Probabilistic Forecasting of the Disturbance Storm Time Index: An Autoregressive Gaussian Process Approach, *Space Weather*, 15(8), pp. 1004-1019.

[11] Chen, Z., Wang, B., and Gorban, A.N. (2017), Multivariate Gaussian and Student-t Process Regression for Multi-output Prediction, *arXiv*, 1703.04455.

[12] Cochrane, J.H., and Piazzesi, M. (2005), Bond Risk Premia, *American Economic Review*, 95(1), pp. 138-160.

[13] Cover, T.M., and Thomas, J.A. (1991), *Elements of Information Theory*, John Wiley & Sons.

[14] Cressie, N. (1992), Statistics for Spatial Data, *Terra Nova*, 4(5), pp. 613-617.

[15] DeBrusk, C., and Du, E. (2018), Why Wall Street Needs to Make Investing in Machine Learning a Higher Priority, *Oliver Wyman Report*.

[16] Deisenroth, M.P., and Ng, J.W. (2015), Distributed Gaussian Processes, *arXiv*, 1502.02843.

[17] Deisenroth, M.P, Fox, D., and Rasmussen, C.E. (2015), Gaussian Processes for Data-efficient Learning in Robotics and Control, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2), pp. 408-423.

[18] Diebold, F.X., and Li, C. (2006), Forecasting the Term Structure of Government Bond Yields, *Journal of Econometrics*, 130(2), pp. 337-364.

[19] Duane, S., Kennedy, A.D., Pendleton, B.J., and Roweth, D. (1987), Hybrid Monte Carlo, *Physics letters B*, 195(2), pp. 216-222.

[20] Duvenaud, D., Lloyd, J.R., Grosse, R., Tenenbaum, J.B., and Ghahramani, Z. (2013), Structure Discovery in Nonparametric Regression through Compositional Kernel Search, *Proceedings of the* 30th *International Conference on Machine Learning*, 28(3), pp. 1166-1174.

[21] Durrande, N., Hensman, J., Rattray, M., and Lawrence, N.D. (2016), Detecting Periodicities with Gaussian Processes, *PeerJ Computer Science*, 2:e50.

[22] Financial Stability Board (2017), Artificial Intelligence and Machine Learning in Financial Services: Market Developments and Financial Stability Implications, *FSB Report*, November.

[23] Forrester, A., Sobester, A., and Keane, A. (2008), *Engineering Design via Surrogate Modelling: A Practical Guide*, John Wiley & Sons.

[24] Frazier, P.I., Powell, W., and Dayanik, S. (2009), The Knowledge-gradient Policy for Correlated Normal Beliefs, *INFORMS Journal on Computing*, 21(4), pp. 599-613.

[25] Frazier, P.I. (2018), A Tutorial on Bayesian Optimization, *arXiv*, 1807.02811.

[26] Gabay, D., and Mercier, B. (1976), A Dual Algorithm for the Solution of Nonlinear Variational Problems via Finite Element Approximation, *Computers & Mathematics with Applications*, 2(1), pp. 17-40.

[27] Garrido-Merchán, E.C., and Hernández-Lobato, D. (2017), Dealing with Integer-valued Variables in Bayesian Optimization with Gaussian Processes, *arXiv*, 1706.03673.

[28] Gupta, A.K., and Nagar, D.K. (1999), *Matrix Variate Distributions*, Monographs and Surveys in Pure and Applied Mathematics, 104, Chapman & Hall/CRC Press.

[29] Härle, P., Havas, A., Kremer, A., Rona, D., and Samandari, H. (2015), The Future of Bank Risk Mangement, *McKinsey Working Papers on Risk*, December.

[30] He, D., Guo, M., Zhou, J., and Guo, V. (2018), The Impact of Articial Intelligence (AI) on the Financial Job Market, *Boston Consulting Group*.

[31] Hennig, P., and Schuler, C.J. (2012), Entropy Search for Information-efficient Global Optimization, *Journal of Machine Learning Research*, 13, pp. 1809-1837.

[32] Hernández-Lobato, J.M., Hoffman, M.W., and Ghahramani, Z. (2014), Predictive Entropy Search for Efficient Global Optimization of Black-box Functions, in Ghahramani, Z., Welling, M, Cortes, C., Lawrence, N.D., and Weinberger, K.Q. (Eds), *Advances in Neural Information Processing Systems*, 27, pp. 918-926.

[33] Hutter, F., Hoos, H.H., and Leyton-Brown, K. (2011), Sequential Model-based Optimization for General Algorithm Configuration, in Coello Coello, C.A. (Ed.), *Learning and Intelligent Optimization*, 5th International Conference on Learning and Intelligent Optimization, Springer, pp. 507-523.

[34] Jones, D.R. (2001), A Taxonomy of Global Optimization Methods based on Response Surfaces, *Journal of Global Optimization*, 21(4), pp. 345-383.

[35] Jones, D.R., Schonlau, M., and Welch, W.J. (1998), Efficient Global Optimization of Expensive Black-box Functions, *Journal of Global Optimization*, 13(4), pp. 455-492.

[36] Kroha, P., and Škoula, M. (2018), Hurst Exponent and Trading Signals Derived from Market Time Series, in *ICEIS 2018 – 20th International Conference on Enterprise Information System*, 1, pp. 371-378.

[37] Kushner, H.J. (1964), A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise, *Journal of Basic Engineering*, 86(1), pp. 97-106.

[38] Liu, H., Cai, J., Wang, Y., and Ong, Y.S. (2018), Generalized Robust Bayesian Committee Machine for Large-scale Gaussian Process Regression, *arXiv*, 1806:00720.

[39] MacKay, D.J.C. (1998), Introduction to Gaussian Processes, *NATO ASI Series F Computer and Systems Sciences*, 168, pp. 133-166.

[40] McKinsey (2016), FinTechnicolor: The New Picture in Finance, *McKinsey Report*, 2016

[41] Minka, T.P. (2001), *A Family of Algorithms for Approximate Bayesian Inference*, *Doctoral Dissertation*, Massachusetts Institute of Technology.

[42] Močkus, J. (1975), On Bayesian Methods for Seeking the Extremum, in Marchuk, G.I. (Ed.), *Optimization Techniques IFIP Technical Conference*, Springer, pp. 400-404.

[43] Neal, R.M. (2011), MCMC Using Hamiltonian Dynamics, in Brooks S., Gelman A., Jones G.L., and Meng, X-L. (Eds), *Handbook of Markov Chain Monte Carlo*, Chapman & Hall/CRC Press, pp. 113-162.

[44] OECD (2017), Robo-Advice for Pensions, *OECD Report*, `http://www.oecd.org/going-digital`.

[45] Osborne, M.A., Roberts, S.J., Rogers, A., and Jennings, N.R. (2012) Real-time Information Processing of Environmental Sensor Network Data Using Bayesian Gaussian Processes, *ACM Transactions on Sensor Networks*, 9(1), pp. 1-32.

[46] Quiñonero-Candela, J., and Rasmussen, C.E. (2005), A Unifying View of Sparse Approximate Gaussian Process Regression, *Journal of Machine Learning Research*, 6, pp. 1939-1959.

[47] Quiñonero-Candela, J., Rasmussen, C.E., and Williams, C.K. (2007), Approximation Methods for Gaussian Process Regression, in Bottou, L., Chapelle, O., DeCoste, D., and Weston, J. (Eds), *Large-Scale Kernel Machines*, MIT Press, pp. 202-223.

[48] Rasmussen, C.E., and Nickisch, H. (2010), Gaussian Processes for Machine Learning (GPML) Toolbox, *Journal of Machine Learning Research*, 11, pp. 3011-3015.

[49] Rasmussen, C.E., and Williams, C.K.I, (2006), *Gaussian Processes for Machine Learning*, Adaptive Computation and Machine Learning, MIT Press.

[50] Rebonato, R. (2018), *Bond Pricing and Yield Curve Modeling: A Structural Approach*, Cambridge University Press.

[51] Richard, J-C., and Roncalli, T. (2019), Constrained Risk Budgeting Portfolios: Theory, Algorithms, Applications & Puzzles, *SSRN*, `www.ssrn.com/abstract=3331184`.

[52] Roncalli, T. (2013), *Introduction to Risk Parity and Budgeting*, Chapman & Hall/CRC Financial Mathematics Series.

[53] Sambasivan, R., and Das S. (2017), A Statistical Machine Learning Approach to Yield Curve Forecasting, in *2017 International Conference on Computational Intelligence in Data Science (ICCIDS)*, IEEE, pp. 1-6.

[54] Scott, W., Frazier, P. and Powell, W. (2011), The Correlated Knowledge Gradient for Simulation Optimization of Continuous Parameters using Gaussian Process Regression, *SIAM Journal on Optimization*, 21(3), pp. 996-1026.

[55] Shah, A., Wilson, A., and Ghahramani, Z. (2014), Student-t Processes as Alternatives to Gaussian Processes, in Kaski, S., and Corander, J. (Eds), *Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, 33, pp. 877-885.

[56] Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., and De Freitas, N. (2016), Taking the Human out of the Loop: A Review of Bayesian Optimization, *Proceedings of the IEEE*, 104(1), pp. 148-175.

[57] Sheffield Machine Learning Group, *GPy: Gaussian Process Framework in Python*, `http://github.com/SheffieldML/GPy`, 2012.

[58] Sheffield Machine Learning Group, *GPyOpt: Bayesian Optimization Framework in Python*, `http://github.com/SheffieldML/GPy`, 2016.

[59] Snoek, J., Larochelle, H., and Adams, R.P. (2012), Practical Bayesian Optimization of Machine Learning Algorithms, in Pereira, F., Burges, C.J.C., Bottou, L. and Weinberger, K.Q. (Eds), *Advances in Neural Information Processing Systems*, 25, pp. 2951-2959.

[60] Titsias, M.K., Rattray, M., and Lawrence, N. D. (2011), Markov Chain Monte Carlo Algorithms for Gaussian Processes, in Barber, D., Cemgil, A.T., and Chiappa, S. (Eds), *Bayesian Time Series Models*, Cambridge University Press, pp. 295-316.

[61] Tracey, B.D., and Wolpert, D. (2018), Upgrading from Gaussian Processes to Student's-T Processes, *arXiv*, 1801.06147.

[62] Tresp, V. (2000), A Bayesian Committee Machine, *Neural Computation*, 12(11), pp. 2719-2741.

[63] Wilson, A.G., and Adams, R.P. (2013), Gaussian Process Kernels for Pattern Discovery and Extrapolation, *Proceedings of the* 30$^{\text{th}}$ *International Conference on Machine Learning*, 28(3), pp. 1067-1075.

[64] Wilson, A.G. (2015), Gaussian Correction to Spectral Mixture (SM) Kernel Derivation for Multidimensional Inputs, `http://www.cs.cmu.edu/~andrewgw/typo.pdf`.

A Mathematical results

A.1 Notations

We use the following notations:

- I_n is the identity matrix of $\mathbb{R}^n$.
- $\mathbf{1}$ is a vector of ones.
- $\mathbb{1}_\Omega(x)$ is the convex indicator function of Ω: $\mathbb{1}_\Omega(x) = 0$ for $x \in \Omega$ and $\mathbb{1}_\Omega(x) = +\infty$ for $x \notin \Omega$.
- x^+ is the positive part $\max(0, x)$ of x.
- $\Phi(x)$ is the standard normal cumulative distribution function:

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}t^2} \mathrm{d}t$$

  whereas $\phi(x)$ is its probability density function:

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

- $\Gamma(s)$ is Euler's gamma function:

$$\Gamma(s) = \int_0^\infty x^{s-1} e^{-x} \mathrm{d}x$$

- Γ_n is the multivariate gamma function:

$$\Gamma_n(s) = \pi^{\frac{n(n-1)}{2}} \prod_{i=1}^{n} \Gamma\left(s + \frac{(1-i)}{2}\right)$$

- $f(x) \sim \mathcal{GP}(m(x), \mathcal{K}(x, x))$ denotes a Gaussian process.
- $\hat{f}(x^\star) = f(x^\star \mid x, y)$ is the random vector of outputs conditional to the sample (x, y).
- $\hat{m}(x^\star) = m(x^\star \mid x, y)$ is the conditional expectation of $x^\star$ with respect to the sample (x, y).
- $\hat{\mathcal{K}}(x^\star, x^\star) = \mathcal{K}(x^\star, x^\star \mid x, y)$ is the conditional covariance matrix of $x^\star$ with respect to the sample (x, y).
- $x = (x_1, \ldots, x_n)$ is a matrix of dimension $n \times d$.
- $x^\star = (x_1^\star, \ldots, x_n^\star)$ is a matrix of dimension $n^\star \times d$.

A.2 Conditional Gaussian distribution

Let us consider a Gaussian random vector defined as follows:

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, \begin{pmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{pmatrix}\right)$$

Then, the marginal distributions of X and Y are given by $X \sim \mathcal{N}(\mu_x, \Sigma_{xx})$ and $Y \sim \mathcal{N}(\mu_y, \Sigma_{yy})$, and we have $\text{cov}(X, Y) = \Sigma_{xy}$. The conditional distribution of Y given $X = x$ is a multivariate normal distribution:

$$Y \mid X = x \sim \mathcal{N}\left(\mu_{y|x}, \Sigma_{yy|x}\right)$$

where:

$$\mu_{y|x} = \mathbb{E}[Y \mid X = x] = \mu_y + \Sigma_{yx}\Sigma_{xx}^{-1}(x - \mu_x)$$

and:

$$\Sigma_{yy|x} = \sigma^2[Y \mid X = x] = \Sigma_{yy} - \Sigma_{yx}\Sigma_{xx}^{-1}\Sigma_{xy}$$

A.3 Derivation of the SoR approximation

The SoR approximation is based on the Woodbury matrix identity:

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B\left(C^{-1} + DA^{-1}B\right)^{-1}DA^{-1} \tag{8}$$

and four approximations:

$$\begin{aligned}
\mathcal{K}(x, x) &\approx \mathcal{K}(x, x_m)\mathcal{K}(x_m, x_m)^{-1}\mathcal{K}(x_m, x) && (9)\\
\mathcal{K}(x^\star, x) &\approx \mathcal{K}(x^\star, x_m)\mathcal{K}(x_m, x_m)^{-1}\mathcal{K}(x_m, x) && (10)\\
\mathcal{K}(x, x^\star) &\approx \mathcal{K}(x, x_m)\mathcal{K}(x_m, x_m)^{-1}\mathcal{K}(x_m, x^\star) && (11)\\
\mathcal{K}(x^\star, x^\star) &\approx \mathcal{K}(x^\star, x_m)\mathcal{K}(x_m, x_m)^{-1}\mathcal{K}(x_m, x^\star) && (12)
\end{aligned}$$

A.3.1 Preliminary results

Using Equations (8) and (9), we deduce that:

$$\begin{aligned}
(*) &= \left(\mathcal{K}(x, x) + \sigma_\varepsilon^2 I_n\right)^{-1} \\
&\approx \left(\sigma_\varepsilon^2 I_n + \mathcal{K}(x, x_m)\mathcal{K}(x_m, x_m)^{-1}\mathcal{K}(x_m, x)\right)^{-1} \\
&= \frac{1}{\sigma_\varepsilon^2} I_n - \frac{1}{\sigma_\varepsilon^2} I_n \mathcal{K}(x, x_m) \cdot \\
&\quad \left(\mathcal{K}(x_m, x_m) + \mathcal{K}(x_m, x)\frac{1}{\sigma_\varepsilon^2} I_n \mathcal{K}(x, x_m)\right)^{-1} \mathcal{K}(x_m, x)\frac{1}{\sigma_\varepsilon^2} I_n \\
&= \frac{1}{\sigma_\varepsilon^2} I_n - \frac{1}{\sigma_\varepsilon^2}\mathcal{K}(x, x_m)\left(\sigma_\varepsilon^2 \mathcal{K}(x_m, x_m) + \mathcal{K}(x_m, x)\mathcal{K}(x, x_m)\right)^{-1}\mathcal{K}(x_m, x)
\end{aligned}$$

We obtain the following relationship:

$$\left(\mathcal{K}(x, x) + \sigma_\varepsilon^2 I_n\right)^{-1} \approx \frac{1}{\sigma_\varepsilon^2} I_n - \frac{1}{\sigma_\varepsilon^2}\mathcal{K}(x, x_m)\tilde{\mathcal{K}}(x_m, x_m)^{-1}\mathcal{K}(x_m, x) \tag{13}$$

where:

$$\tilde{\mathcal{K}}(x_m, x_m) = \mathcal{K}(x_m, x)\mathcal{K}(x, x_m) + \sigma_\varepsilon^2 \mathcal{K}(x_m, x_m)$$

Moreover, we have $\tilde{\mathcal{K}}(x_m, x_m)\,\tilde{\mathcal{K}}(x_m, x_m)^{-1} = I_m$ or:

$$I_m = \mathcal{K}(x_m, x)\,\mathcal{K}(x, x_m)\,\tilde{\mathcal{K}}(x_m, x_m)^{-1} + \sigma_\varepsilon^2 \mathcal{K}(x_m, x_m)\,\tilde{\mathcal{K}}(x_m, x_m)^{-1} \tag{14}$$

In a similar way, we have $\tilde{\mathcal{K}}(x_m, x_m)^{-1}\,\tilde{\mathcal{K}}(x_m, x_m) = I_m$ or:

$$I_m = \tilde{\mathcal{K}}(x_m, x_m)^{-1}\,\mathcal{K}(x_m, x)\,\mathcal{K}(x, x_m) + \sigma_\varepsilon^2 \tilde{\mathcal{K}}(x_m, x_m)^{-1}\,\mathcal{K}(x_m, x_m) \tag{15}$$

A.3.2 Approximation of the conditional expectation

Using Equations (10) and (13), we have:

$$\begin{aligned}
\hat{m}(x^\star) &= \mathcal{K}(x^\star, x)\left(\mathcal{K}(x, x) + \sigma_\varepsilon^2 I_n\right)^{-1} y \\
&\approx \mathcal{K}(x^\star, x_m)\,\mathcal{K}(x_m, x_m)^{-1}\,\mathcal{K}(x_m, x)\left(\mathcal{K}(x, x) + \sigma_\varepsilon^2 I_n\right)^{-1} y \\
&\approx \mathcal{K}(x^\star, x_m)\,\mathcal{K}(x_m, x_m)^{-1}\,\mathcal{K}(x_m, x)\cdot \\
&\quad \left(\frac{1}{\sigma_\varepsilon^2} I_n - \frac{1}{\sigma_\varepsilon^2}\mathcal{K}(x, x_m)\,\tilde{\mathcal{K}}(x_m, x_m)^{-1}\,\mathcal{K}(x_m, x)\right) y \\
&= \frac{1}{\sigma_\varepsilon^2}\mathcal{K}(x^\star, x_m)\,\mathcal{K}(x_m, x_m)^{-1}\cdot \\
&\quad \left(I_m - \mathcal{K}(x_m, x)\,\mathcal{K}(x, x_m)\,\tilde{\mathcal{K}}(x_m, x_m)^{-1}\right)\mathcal{K}(x_m, x)\, y
\end{aligned}$$

Using Equation (14), we deduce that:

$$\begin{aligned}
(*) &= I_m - \mathcal{K}(x_m, x)\,\mathcal{K}(x, x_m)\,\tilde{\mathcal{K}}(x_m, x_m)^{-1} \\
&= \mathcal{K}(x_m, x)\,\mathcal{K}(x, x_m)\,\tilde{\mathcal{K}}(x_m, x_m)^{-1} + \sigma_\varepsilon^2 \mathcal{K}(x_m, x_m)\,\tilde{\mathcal{K}}(x_m, x_m)^{-1} - \\
&\quad \mathcal{K}(x_m, x)\,\mathcal{K}(x, x_m)\,\tilde{\mathcal{K}}(x_m, x_m)^{-1} \\
&= \sigma_\varepsilon^2 \mathcal{K}(x_m, x_m)\,\tilde{\mathcal{K}}(x_m, x_m)^{-1}
\end{aligned}$$

Finally, we obtain the approximation of $\hat{m}(x^\star)$:

$$\begin{aligned}
\hat{m}(x^\star) &\approx \frac{1}{\sigma_\varepsilon^2}\mathcal{K}(x^\star, x_m)\,\mathcal{K}(x_m, x_m)^{-1}\cdot \\
&\quad \sigma_\varepsilon^2 \mathcal{K}(x_m, x_m)\,\tilde{\mathcal{K}}(x_m, x_m)^{-1}\,\mathcal{K}(x_m, x)\, y \\
&= \mathcal{K}(x^\star, x_m)\,\tilde{\mathcal{K}}(x_m, x_m)^{-1}\,\mathcal{K}(x_m, x)\, y
\end{aligned}$$

A.3.3 Approximation of the conditional covariance

In the previous paragraph, we have shown that:

$$\mathcal{K}(x^\star, x)\left(\mathcal{K}(x, x) + \sigma_\varepsilon^2 I_n\right)^{-1} \approx \mathcal{K}(x^\star, x_m)\,\tilde{\mathcal{K}}(x_m, x_m)^{-1}\,\mathcal{K}(x_m, x)$$

It follows that:

$$\begin{aligned}
\hat{\mathcal{K}}(x^\star, x^\star) &= \mathcal{K}(x^\star, x^\star) - \mathcal{K}(x^\star, x)\left(\mathcal{K}(x, x) + \sigma_\varepsilon^2 I_n\right)^{-1}\mathcal{K}(x, x^\star) \\
&\approx \mathcal{K}(x^\star, x^\star) - \mathcal{K}(x^\star, x_m)\,\tilde{\mathcal{K}}(x_m, x_m)^{-1}\,\mathcal{K}(x_m, x)\,\mathcal{K}(x, x^\star)
\end{aligned}$$

We now replace $\mathcal{K}(x, x^\star)$ and $\mathcal{K}(x^\star, x^\star)$ by Equations (11) and (12):

$$\begin{aligned}
\hat{\mathcal{K}}(x^\star, x^\star) &\approx \mathcal{K}(x^\star, x_m)\,\mathcal{K}(x_m, x_m)^{-1}\,\mathcal{K}(x_m, x^\star) - \\
&\quad \mathcal{K}(x^\star, x_m)\,\tilde{\mathcal{K}}(x_m, x_m)^{-1}\,\mathcal{K}(x_m, x)\,\mathcal{K}(x, x_m)\,\mathcal{K}(x_m, x_m)^{-1}\,\mathcal{K}(x_m, x^\star) \\
&= \mathcal{K}(x^\star, x_m)\left(I_m - \tilde{\mathcal{K}}(x_m, x_m)^{-1}\,\mathcal{K}(x_m, x)\,\mathcal{K}(x, x_m)\right)\cdot \\
&\quad \mathcal{K}(x_m, x_m)^{-1}\,\mathcal{K}(x_m, x^\star)
\end{aligned}$$

Using Equation (14), we notice that:

$$\begin{aligned}(*) &= I_m - \tilde{\mathcal{K}}(x_m, x_m)^{-1}\mathcal{K}(x_m, x)\mathcal{K}(x, x_m) \\ &= \tilde{\mathcal{K}}(x_m, x_m)^{-1}\mathcal{K}(x_m, x)\mathcal{K}(x, x_m) + \sigma_\varepsilon^2\tilde{\mathcal{K}}(x_m, x_m)^{-1}\mathcal{K}(x_m, x_m) - \\ &\quad \tilde{\mathcal{K}}(x_m, x_m)^{-1}\mathcal{K}(x_m, x)\mathcal{K}(x, x_m) \\ &= \sigma_\varepsilon^2\tilde{\mathcal{K}}(x_m, x_m)^{-1}\mathcal{K}(x_m, x_m)\end{aligned}$$

Finally, we obtain:

$$\begin{aligned}\hat{\mathcal{K}}(x^\star, x^\star) &\approx \mathcal{K}(x^\star, x_m)\sigma_\varepsilon^2\tilde{\mathcal{K}}(x_m, x_m)^{-1}\mathcal{K}(x_m, x_m)\mathcal{K}(x_m, x_m)^{-1}\mathcal{K}(x_m, x^\star) \\ &= \sigma_\varepsilon^2\mathcal{K}(x^\star, x_m)\tilde{\mathcal{K}}(x_m, x_m)^{-1}\mathcal{K}(x_m, x^\star)\end{aligned}$$

A.4 Hybrid Monte Carlo method

The hybrid Monte Carlo (HMC) algorithm is a special case of Markov Chain Monte Carlo (MCMC) methods. The objective is to perform sampling from a probability distribution for which the density and the gradients are known. This approach was first introduced by Duane *et al.* (1987) and is also known as Hamiltonian Monte Carlo (Neal, 2011), since it generates Markov chain states through Hamiltonian evolution in phase space. More precisely, we note $\pi(q)$ a probability distribution over $\mathbb{R}^n$. In the formalism of Hamiltonian mechanics, the state of a system is described through a position variable q, a momentum (velocity) variable p:

$$p = m\frac{\mathrm{d}q}{\mathrm{d}t}$$

and a function of the two variables, which is called the Hamiltonian:

$$\mathcal{H}(q, p) = V(q) + K(p)$$

where V and K are respectively potential and kinetic energies. K is usually given by $K(p) = \frac{1}{2}\sum_{i=1}^n p_i^2$. The Hamiltonian describes entirely the dynamical evolution of the physical system with Hamilton's equations:

$$\begin{aligned}\frac{\mathrm{d}q}{\mathrm{d}t} &= \frac{\partial\mathcal{H}}{\partial p} \\ \frac{\mathrm{d}p}{\mathrm{d}t} &= -\frac{\partial\mathcal{H}}{\partial q}\end{aligned}$$

Therefore, the evolution of the system is described by the *phase space* (q, p). When the time increases, the Hamiltonian remains constant and the volume is preserved in phase space. Neal (2011) defines the probability distribution over phase space as:

$$\mathbb{P}(q, p) = \frac{1}{C}\exp\left(-\mathcal{H}(q, p)\right)$$

where C is a normalization constant and chooses $V(q) = -\log\pi(q)$. It follows that:

$$\begin{cases} q \sim \pi \\ p \sim \mathcal{N}(0, I) \end{cases}$$

The idea behind HMC is then simple. We build a mountain that is high for small values of π (*i.e.* places where we would not want to sample often), and deep for large values of π, and kick a ball in a random direction. It will roll on the surface, and is attracted to low values

of potential and we stop it after a fixed time period. Then, we repeat the process, and the successive positions taken by the ball form the sample of π.

To simulate Hamiltonian evolution, the leap-frog method is usually used. It is based on Euler's method and finite differences, but is able to enforce Hamiltonian and volume conservation (Neal, 2011):

$$\begin{aligned} p\left(t+\frac{\varepsilon}{2}\right) &= p(t) - \frac{\varepsilon}{2}\frac{\partial V}{\partial q}(q(t)) \\ q(t+\varepsilon) &= q(t) + \varepsilon p\left(t+\frac{\varepsilon}{2}\right) \\ p(t+\varepsilon) &= p\left(t+\frac{\varepsilon}{2}\right) - \frac{\varepsilon}{2}\frac{\partial V}{\partial q}(q(t+\varepsilon)) \end{aligned}$$

where the parameters are $\varepsilon > 0$ and the number of iterations before stopping the Hamiltonian dynamics. Once the dynamics is stopped, a new state (q', p') is proposed. To compensate for numerical errors in the leap-frog integration, a Metropolis step is carried to accept the proposed state with probability:

$$\min\left(1, \frac{\exp(-\mathcal{H}(q', p'))}{\exp(-\mathcal{H}(q, p))}\right)$$

otherwise, the state remains unchanged.

Remark 10. *In the case of hyperparameter posterior sampling, the distribution $\pi(\theta)$ is given by:*

$$\pi(\theta) = p(\theta \mid y) = \frac{p(y \mid \theta)\, p(\theta)}{\int p(y \mid \theta', z)\, p(\theta')\, \mathrm{d}\theta'}$$

where $p(\theta)$ is the prior distribution on hyperparameters. Note that the normalization constant is not needed to sample from the posterior with HMC.

A.5 Computation of $\mathbb{E}\left[(X-c)^+\right]$

We assume that $X \sim \mathcal{N}(\mu, \sigma^2)$ and we would like to calculate $\mathbb{E}\left[(X-c)^+\right]$. We have

$$\begin{aligned} \mathbb{E}\left[(X-c)^+\right] &= \int_{-\infty}^{+\infty} (x-c)\, \mathbb{1}\{x-c \geq 0\}\, \phi\left(x; \mu, \sigma^2\right)\, \mathrm{d}x \\ &= \int_{c}^{+\infty} (x-c)\, \phi\left(x; \mu, \sigma^2\right)\, \mathrm{d}x \\ &= \int_{c}^{+\infty} x\phi\left(x; \mu, \sigma^2\right)\, \mathrm{d}x - c\int_{c}^{+\infty} \phi\left(x; \mu, \sigma^2\right)\, \mathrm{d}x \\ &= \int_{c}^{+\infty} \frac{x}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \mathrm{d}x - c\left(1 - \Phi\left(\frac{c-\mu}{\sigma}\right)\right) \end{aligned}$$

By considering the change of variable $y = \sigma^{-1}(x - \mu)$, we obtain:

$$\begin{aligned}
(*) &= \int_c^{+\infty} \frac{x}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \mathrm{d}x \\
&= \int_{\sigma^{-1}(c-\mu)}^{+\infty} \frac{\mu + \sigma y}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}y^2\right) \mathrm{d}y \\
&= \mu\left(1 - \Phi\left(\frac{c-\mu}{\sigma}\right)\right) + \frac{\sigma}{\sqrt{2\pi}}\left[-e^{-y^2}\right]_{\sigma^{-1}(c-\mu)}^{+\infty} \\
&= \mu\left(1 - \Phi\left(\frac{c-\mu}{\sigma}\right)\right) + \sigma\phi\left(\frac{c-\mu}{\sigma}\right)
\end{aligned}$$

Finally, we deduce that:

$$\begin{aligned}
\mathbb{E}\left[(X-c)^+\right] &= (\mu - c)\left(1 - \Phi\left(\frac{c-\mu}{\sigma}\right)\right) + \sigma\phi\left(\frac{c-\mu}{\sigma}\right) \\
&= (\mu - c)\,\Phi\left(\frac{\mu-c}{\sigma}\right) + \sigma\phi\left(\frac{\mu-c}{\sigma}\right)
\end{aligned}$$

Remark 11. *If we are interested in* $\mathbb{E}\left[(c-X)^+\right]$*, we use the identity:*

$$X - c = (X-c)^+ - (c-X)^+$$

and we find:

$$\mathbb{E}\left[(c-X)^+\right] = (c-\mu)\,\Phi\left(\frac{c-\mu}{\sigma}\right) + \sigma\phi\left(\frac{c-\mu}{\sigma}\right)$$

A.6 Improvement-based minimization problem

If we are interested in finding the minimum, we define the improvement as $\Delta_n(x^\star) = \left(\tau - \hat{f}_n(x^\star)\right)^+$. It follows that:

$$\Pr\{\Delta_n(x^\star) > 0\} = \Phi\left(\frac{\tau - \hat{m}_n(x^\star)}{\sqrt{\hat{\mathcal{K}}_n(x^\star, x^\star)}}\right)$$

and:

$$\mathrm{EI}_n(x^\star) = (\tau - \hat{m}_n(x^\star))\,\Phi\left(\frac{\tau - \hat{m}_n(x^\star)}{\sqrt{\hat{\mathcal{K}}_n(x^\star, x^\star)}}\right) + \sqrt{\hat{\mathcal{K}}_n(x^\star, x^\star)}\phi\left(\frac{\tau - \hat{m}_n(x^\star)}{\sqrt{\hat{\mathcal{K}}_n(x^\star, x^\star)}}\right)$$

Therefore, we have $x_{n+1} = \arg\max \mathcal{U}_n(x^\star)$ where $\mathcal{U}_n(x^\star) = \Pr\{\Delta_n(x^\star) > 0\}$ or $\mathcal{U}_n(x^\star) = \mathrm{EI}_n(x^\star)$. The standard problem is obtained by setting $\tau = f_n(\varkappa_n^\star)$ where $\varkappa_n^\star = \arg\min_{\varkappa \in x} f(\varkappa)$.

A.7 Matrix-variate Student-t distribution

We first recall that a random vector $X \in \mathbb{R}^n$ has a multivariate Student-t distribution with mean μ and scale matrix Σ if its probability density function is equal to:

$$f(x) = \frac{\Gamma(\upsilon/2)}{(\nu\pi)^{n/2}\,\Gamma((\upsilon - n)/2)}\,|\Sigma|^{-1/2}\left(1 + \frac{1}{\nu}y^\top\Sigma^{-1}y\right)^{-\upsilon/2}$$

where $y = x - \mu$, $\upsilon = \nu + n$ and ν is the degrees of freedom. Let X be an $n \times p$ random matrix. It has a matrix-variate Student-t distribution with mean matrix $M \in \mathbb{R}^{n \times p}$ and covariance matrices $\Sigma \in \mathbb{R}^{n \times n}$ and $\Omega \in \mathbb{R}^{p \times p}$ if its probability density function is equal to (Gupta and Nagar, 1999):

$$\begin{aligned} f(X) &= \frac{\Gamma_n(\upsilon/2)}{(\nu\pi)^{np/2}\,\Gamma_n((\upsilon-p)/2)}\,|\Sigma|^{-p/2}\,|\Omega|^{-n/2}\,\cdot \\ &\quad \left|I_n + \Sigma^{-1}(X-M)\,\Omega^{-1}(X-M)\right|^{-\upsilon/2} \end{aligned}$$

where:

$$\upsilon = \nu + n + p - 1$$

We note $X \sim \mathcal{MT}_{n,p}(M, \Sigma, \Omega; \nu)$. This matrix distribution has several properties similar to Gaussian random matrices, which make it suitable for multivariate regression. For example, we have:

$$\mathbb{E}[X] = M$$

and:

$$\operatorname{cov}\left(\operatorname{vec} X^\top\right) = \frac{1}{\nu - 2}\Sigma \otimes \Omega \qquad \text{if } \upsilon > 2$$

Gupta and Nagar (1999) also show that:

$$X^\top \sim \mathcal{MT}_{p,n}\left(M^\top, \Sigma, \Omega; \nu\right)$$

If we assume that:

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim \mathcal{MT}_{n,p}\left(\begin{pmatrix} M_x \\ M_y \end{pmatrix}, \begin{pmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{pmatrix}, \Omega; \nu\right)$$

we have:

$$X \sim \mathcal{MT}_{n_x,p}(M_x, \Sigma_{xx}, \Omega; \nu)$$

and:

$$Y \sim \mathcal{MT}_{n_y,p}(M_y, \Sigma_{yy}, \Omega; \nu)$$

Moreover, the conditional distribution is still a matrix-variate Student-t distribution:

$$Y \mid X = x \sim \mathcal{MT}_{n_y,p}\left(M_{y|x}, \Sigma_{yy|x}, \Omega_{yy|x}; \nu + n_x\right)$$

where:

$$\begin{cases} M_{y|x} = M_y + \Sigma_{yx}\Sigma_{xx}^{-1}(x - M_x) \\ \Sigma_{yy|x} = \Sigma_{yy} - \Sigma_{yx}\Sigma_{xx}^{-1}\Sigma_{xy} \\ \Omega_{yy|x} = \Omega + (x - M_x)^\top \Sigma_{xx}^{-1}(x - M_x) \end{cases}$$

If we consider a column-based partition:

$$\begin{pmatrix} X & Y \end{pmatrix} \sim \mathcal{MT}_{n,p}\left(\begin{pmatrix} M_x & M_y \end{pmatrix}, \Sigma, \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix}; \nu\right)$$

we have:

$$X \sim \mathcal{MT}_{n,p_x}(M_x, \Sigma, \Omega_{xx}; \nu)$$

and:

$$Y \sim \mathcal{MT}_{n,p_y}(M_y, \Sigma, \Omega_{yy}; \nu)$$

For the conditional distribution, Gupta and Nagar (1999) show that:

$$Y \mid X = x \sim \mathcal{MT}_{n,p_y}\left(M_{y|x}, \Sigma_{yy|x}, \Omega_{yy|x}; \nu + p_x\right)$$

where:

$$\begin{cases} M_{y|x} = M_y + (x - M_x)\,\Omega_{xx}^{-1}\Omega_{xy} \\ \Sigma_{yy|x} = \Sigma + (x - M_x)\,\Omega_{xx}^{-1}(x - M_x)^\top \\ \Omega_{yy|x} = \Omega_{yy} - \Omega_{yx}\Omega_{xx}^{-1}\Omega_{xy} \end{cases}$$

A.8 ADMM algorithm

The alternating direction method of multipliers (ADMM) is an algorithm introduced by Gabay and Mercier (1976) to solve problems which can be expressed as[21]:

$$\begin{aligned} \{x^\star, z^\star\} &= \arg\min f(x) + g(z) \\ &\text{s.t.} \quad Ax + Bz - c = 0 \end{aligned} \tag{16}$$

where $A \in \mathbb{R}^{p\times n}$, $B \in \mathbb{R}^{p\times m}$, $c \in \mathbb{R}^p$, and the functions $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ and $g : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ are proper closed convex functions. Boyd *et al.* (2011) show that the ADMM algorithm consists of three steps:

1. The x-update is:

$$x^{(k)} = \arg\min \left\{ f(x) + \frac{\varphi}{2} \left\| Ax + Bz^{(k-1)} - c + u^{(k-1)} \right\|_2^2 \right\} \tag{17}$$

2. The z-update is:

$$z^{(k)} = \arg\min \left\{ g(z) + \frac{\varphi}{2} \left\| Ax^{(k)} + Bz - c + u^{(k-1)} \right\|_2^2 \right\} \tag{18}$$

3. The u-update is:

$$u^{(k)} = u^{(k-1)} + \left(Ax^{(k)} + Bz^{(k)} - c \right) \tag{19}$$

In this approach, $u^{(k)}$ is the dual variable of the primal residual $r = Ax + Bz - c$ and φ is the ℓ_2 penalty variable. In the paper, we use the notations $f^{(k)}(x)$ and $g^{(k)}(z)$ when referring to the objective functions that are defined in the x- and z-steps.

B Software

Throughout this article, we used of the following open-source software libraries:

- GPML Matlab Toolbox (Rasmussen and Nickisch, 2010)
- GPy: A Gaussian process framework in Python

  `http://github.com/SheffieldML/GPy`

- GPyOpt: A Bayesian Optimization Framework in Python

  `http://github.com/SheffieldML/GPyOpt`

C Additional figures

[21] We follow the standard presentation of Boyd *et al.* (2011) on ADMM.

Figure 14: Estimated ridge penalization λ_t (in %)

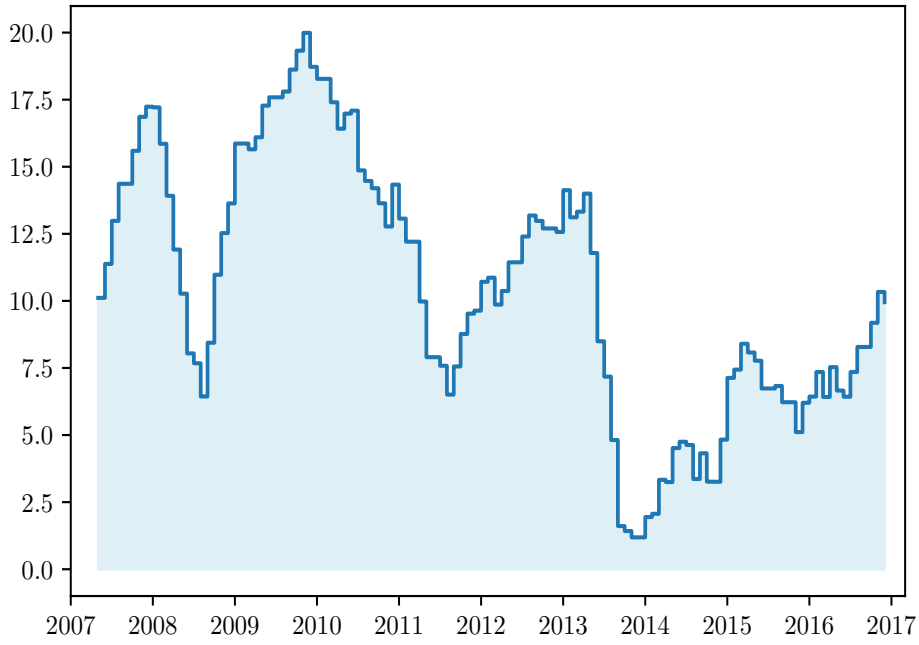


Figure 15: Estimated return window length $\ell_t(\mu)$ (in months)

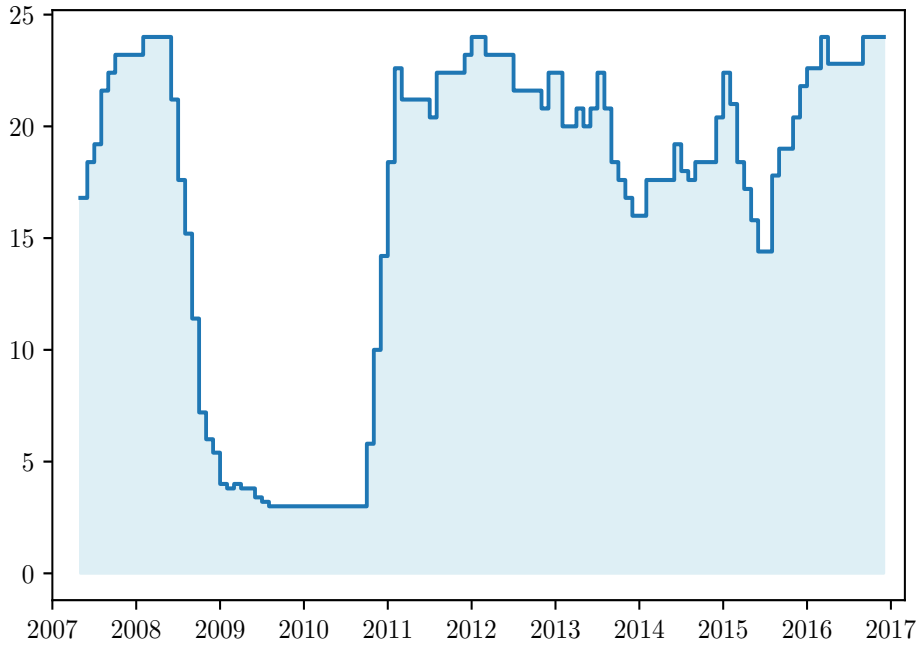


Figure 16: Estimated covariance window length $\ell_t(\Sigma)$ (in months)