Politecnico di Torino

Master of Science in Mathematical Engineering

Master Degree Thesis

Metamodels for Robust Optimization

Supervisor:
Prof. Paolo Brandimarte

Candidate:
Amedeo Biolatti

March 19^{th} 2018

Come una stampa antica bavarese
vedo al tramonto il cielo subalpino...
Da Palazzo Madama al Valentino
ardono l'Alpi tra le nubi accese...
È questa l'ora antica torinese,
è questa l'ora vera di Torino...

Guido Gustavo Gozzano

Abstract

In many real world applications the resolution of optimization problems involving the evaluation of costly functions is needed and sometimes the evaluation process is contaminated by noise. This case has been widely studied in literature, but the case of robust optimization, requiring the evaluation of many different functions, has been neglected. In this work we propose a variant of standard *Bayesian optimization* methods that in some cases achieve better performance with the same amount of observations.

Contents

Chapter 1

Introduction to Optimization

1.1 Mathematical Optimization

In many fields of mathematics, computer science or real life applications a problem that commonly arises is to find the best element between a set of candidates with respect to some criterion. Usually a function, mapping from the set of candidate solutions to real numbers, is used to indicate the "goodness" of a solution, since real numbers induce a partial ordering on the set of candidate solutions. A common example is the maximization of some measure of utility, such the profit, depending on some *decision variables*, i.e. the things that we can directly influence. The details of each problem can be very different and so the procedures used to find a solution, but here we try to present a brief introduction to the general terminology.

Indicated with $\mathcal{X}$ the set of possible choices, also called *search space*, and given the criterion of choice defined by the function $f : \mathcal{X} \to \mathbb{R}$, generally called *objective function*, we want to find $x_{opt} \in \mathcal{X}$ such that $f(x_{opt})$ has the greatest possible value (or the least one if we want to minimize the objective function) between all the elements of $\mathcal{X}$. Often the search space is defined starting from a simple and wider set (for example $\mathbb{R}^n$ or $\mathbb{N}^n$) and then it is refined adding conditions that have to be satisfied by x_{opt}, called *constraints*. Constraints are generally expressed as an equation or an inequality defined by a functions $\mathcal{X} \to \mathbb{R}$ and are generally distinguished in

- equality constraints, such as $g(x) = 0$
- inequality constraints, such as $h(x) \geq 0$

A simple and quite general optimization problem can be expressed as follows

$$\begin{aligned} &\max_{x \in \mathcal{X}} f(x) \\ &g_i(x) = 0 \ \forall i \in I \\ &h_j(x) \geq 0 \ \forall j \in J \end{aligned} \tag{1.1}$$

where $\{g_i\}_{i\in I}$ is the set of equality constraints and $\{h_j\}_{j\in J}$ is the set of inequality constraints.

The strategies to solve this kind of problems are various and in many cases it is not possible to assure that the optimal point has been found in the given time, but often a good approximation is sufficient; in some cases it is even possible to find an upper bound on the difference between the value of the objective function of current solution and the optimal one, to assure that the current solution differs not too much from the optimal one. However the solver method has to be chosen carefully since the search space in many applications is huge and the wrong approaches could not give any result with the available resources.

1.2 Robust Optimization

Sometimes solutions found by standard methods are not satisfactory in practice, even if the optimal one with respect to the objective function has been found; this affirmation could seem contradictory, but more often than not the objective function is not something to take for granted and its definition is as important (or even more important) than the solver methods itself. This could happen for example if the solution found by a method can not be exactly replicated in the real application and noise is "injected" in the solution itself, as in the case of numerical errors or physical measurement tolerances. Based on this consideration, we would like to find more "robust" criterion of choice. The word robust can have many different meanings and therefore many different kinds of "robust optimization" have been proposed, for example resilience to implementation noise, stochastic constraints and uncertainty in the definition of the optimization function. In this work we will focus on robustness with respect to uncertainties in the objective function.

As previously exposed, the objective function is not always given so we could be uncertain on which objective function to use. At a first level, the uncertainty could arise from the fact that the value we want to maximize could be a random variable. In this case we could use some statistic of the random variable and the most obvious possibility is its expected value. This approach has the advantage to be simple, but it also has some limitations: it doesn't consider the variance of the variable and higher moments, therefore it doesn't account for the risk of bad outcomes and the benefits good ones. A common alternative is the *value at risk*, defined as the α-quantile of the distribution of the considered random variable, where $\alpha \in (0, 0.5)$ in the case of maximization. This can be interpreted as *"Our random variable as* $1-\alpha$ *probability of being smaller than the value at risk"*. Also this approach as known limitations, for example it ignores the values at the tails of the distribution, but it is a useful alternative.

A second level of uncertainty could arise from other kinds of ambiguity in the objec-

tive function, for example it could depend on parameters that have to be estimated with numerical and/or statistical methods and are subjected to not negligible errors. Examples are the correlation matrix of financial assets or the sales forecast on a granular level that are notoriously hard to correctly estimate, but innumerable other cases are possible. The uncertainties could also arise from distributional uncertainties: for example *"Is the demand of blue hats distributed according a Poisson or negative binomial distribution?"* has not an obvious answer if we have too few historical data.

The set indexing the possible objective functions when varying the parameters estimations, the distributional hypothesis and the other sources of uncertainty is called *uncertainty set* and is usually indicated by $\mathcal{U}$. Therefore we will indicate the set of possible objective functions as $\{f_u\}_{u\in\mathcal{U}}$. How can we handle many, possibly indexed by a infinite set, objective functions? This answer is not trivial, since it is not obvious how to define an ordering on the search space.

We could use a Bayesian approach, imposing a prior distribution on the uncertainty set, and then we could use the expected value as objective function to define a classical optimization problem. In some cases this approach could work, but not always. Sometimes the costs of a negative outcome are much more relevant than the benefits deriving from positive ones and we would like to avoid them as much as possible. Moreover it is not obvious how to place the prior and even a flat one could be too strong as an hypotheses. We should try to find a method that doesn't require a probabilistic distribution over the uncertainty set.

Another idea could be to use the concept of *Pareto efficiency*: an element x of the search space $\mathcal{X}$ is said to be Pareto efficient with respect to a set of objective functions $\{f_u\}_{u\in\mathcal{U}}$ if $\forall\hat{x} \in \mathcal{X}\ \exists u \in \mathcal{U}$ such that $f_u(x) \geq f_u(\hat{x})$; this is equivalent to saying that x can not be beaten contemporaneously by another solution in all the criterions defined by the objective functions. This definition doesn't require a probabilistic distribution over the uncertainty set and it should always avoid solutions that are particularly bad with respect to some criterion. A potential drawback of this method is that there is no guarantee that the Pareto efficient solution is unique, in fact, as shown in figure 1.1, in the vast majority of cases it is not, therefore we would need an ulterior criterion in order to choose between the efficient solutions. Moreover if the uncertainty set is large, or even infinite, find this kind of solutions could be really difficult.

Another possibility could be to use a worst case scenario, trying to find a solution that is quite good with respect to each objective function and it is never too bad. This can be written as follows

$$\max_{x\in\mathcal{X}} \min_{u\in\mathcal{U}} \ f_u(x) \tag{1.2}$$

If $f(x, u)$ is a random variable, i.e. if f is a stochastic process (see 2.2 for the definition),

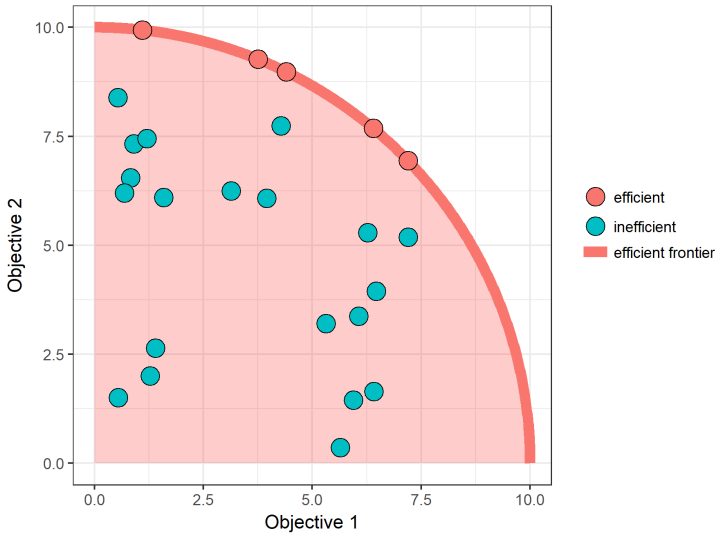


Figure 1.1: An example of some Pareto efficient points and some inefficient ones. The feasible area is shaded in red. The set of all efficient points defines an *efficient frontier* that separates feasible and unfeasible points.

we could use the expected value to define the objective function of our optimization problem as follows:

$$\max_{x \in \mathcal{X}} \min_{u \in \mathcal{U}} \mathbb{E}[f_u(x)] \tag{1.3}$$

Other approaches are also possible rather than using the expected value, such as previously described, but for simplicity we only discuss this case.

It should be noted that this problem is quite different from the maximization of $\mathbb{E}[\min_{u \in \mathcal{U}} f_u(\cdot)]$. In fact minimizing the expected value means that we want to impose a lower bound as a precautionary measure to the value that we could obtain in order to avoid bad situations, while considering expected value of the minimums means that each time the worst scenario between all the possible ones is realized.

1.3 Discretizing the Uncertainty Set

Many methods cannot handle a continuous or mixed discrete-continuous uncertainty set, so it is useful to find a way to discretize it. Fortunately it is not always necessary to keep all the points of the uncertainty set to retain the accuracy of the results, in fact in many cases some values $u \in \mathcal{U}$ cannot lead to a minimum of f_u and therefore doesn't influence the behavior of the objective function.

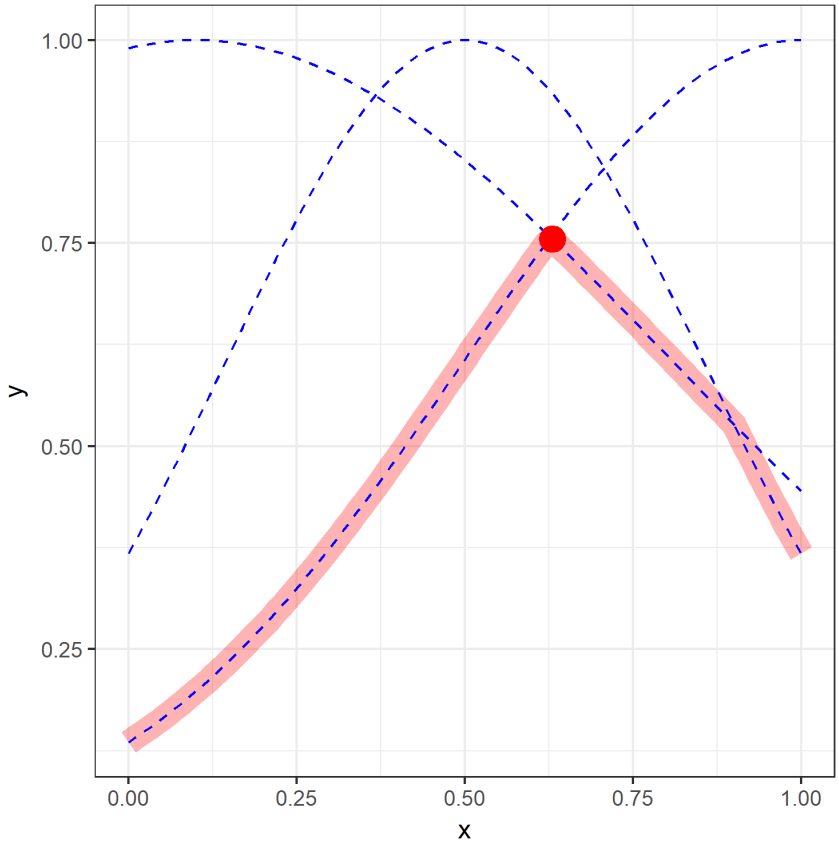


Figure 1.2: An example of a robust optimization problem in the sense of robustness with uncertain objective function, defined by three functions. The dashed lines represent the value of the f_us, the red line represents their minimum, while the red dot is the optimal point.

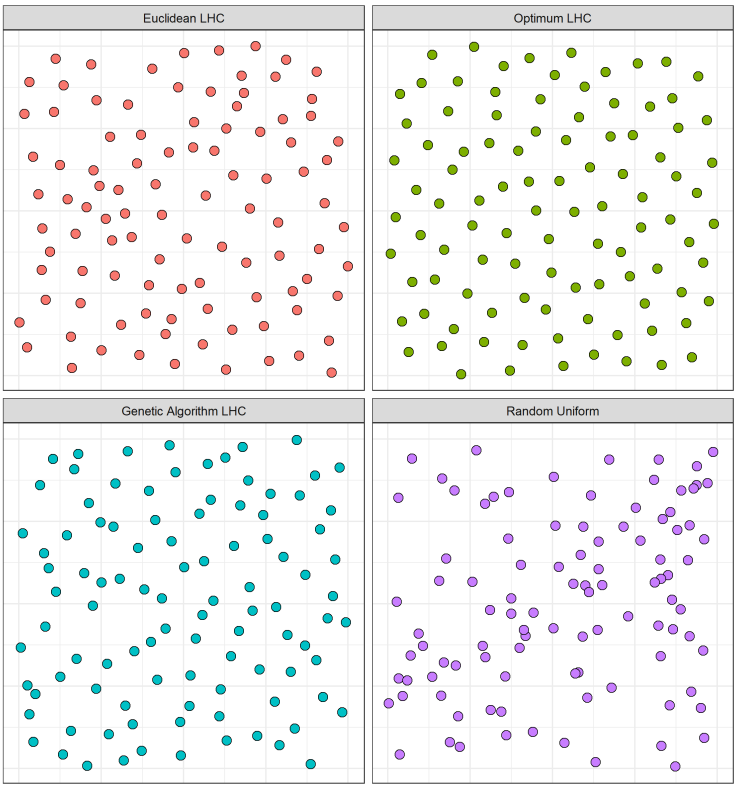


Figure 1.3: An example of the results of three different Latin Hyper-cube Sampling and a random uniform sampling strategies. Each sample consists of 100 points.

Knowledge of the underling function can be useful, in fact it can lead to simplifications of the uncertainty set. For example if the underling function is monotonic it is sufficient to consider points on a subset of the exterior part of the uncertainty set.

It is not always possible to exclude completely some points from the uncertainty set, so we could need to choose a more or less arbitrary subset of points to use as a simplified version of the uncertainty set. A couple of simple methods are the use of a homogeneous grid of points or randomly sampled of points from $\mathcal{U}$, for example according to a uniform distribution. A first improvement on the sampling strategy could be to use sampling methods that avoid to over-sample some regions while under-sampling other ones, such as the Latin Hyper-Cube sampling method. An example of the results given by different sampling methods can be seen in figure 1.3.

It could even be possible to dynamically change the sample points of the uncertainty set but we leave it to future development.

Chapter 2

Optimization via Response Surface

It is not always possible to directly evaluate the objective function and/or its derivatives. In fact in many real world applications the objective function can be evaluated with a combination of: high economic cost, high latency, high computational complexity, non negligible errors. In some cases the empirical evaluation is not even possible, so we have to rely on computer simulations. An example is the design of complex buildings that requires long periods of time to be completed and can collapse if the choices are not sufficiently accurate; the simulations used in this cases can require up to days, or even weeks, but this is nothing compared to the time necessary to complete the building itself and the costs of a potential collapse.

This kind of complexity vastly reduces the number of evaluations of the objective function that we can reasonably require in order to estimate the optimal solution. This problem is even more evident in the case of the maximization of the minimum of the expected value of many noisy functions, indeed to correctly estimate the minimum we have to estimate the expected value of all of them, so in theory many evaluations are required. Smart approaches to the approximation can reduce the total number of observations required, but the problem persists.

We need a method to guide the optimization process in ways that don't require too many function evaluations. But how can we optimize a function without knowing its value? We have to made some reasonable hypothesis on its behavior and then use the available observations to infer characteristics of the function (for example the value in a point not sampled yet) to decide if we need more sample, for which function and in which point of the search space.

A common approach is to substitute the objective function with a surrogate function, often called *response surface*, that is much cheaper to evaluate and exhibits nice analytic properties. Various kind of regressions can be used as *surrogate model*, such any kind of linear or non-linear regression, but non parametric model are often preferred for their flexibility.

2.1 Lipschitzian Model

The first step to build a surrogate model for a function is to explicitly define our assumptions. For example, how smooth is it? How fast can its value change? This kind of information are fundamental to define the strategy we will use to explore the search space.

The simplest (and probably most important) characterization of a function is describing how steep it can be; if the function is differentiable this is equivalent to impose an upper bound to the norm of the gradient, but we can do something very similar without requiring the function to be differentiable. A simple way to do so is to suppose that the function is Lipschitzian with constant L. A function between metric spaces $f : (X, d_1) \to (Y, d_2)$ is said to be Lipschitzian with constant L if

$$d_2(f(x_1), f(x_2)) \leq L d_1(x_1, x_2) \ \forall x_1, x_2 \in X \tag{2.1}$$

If we are working with functions from $\mathbb{R}^n$ to $\mathbb{R}$ we can easily use the norm L_1, L_2 or L_∞ to characterize the spaces as metric, and so the definition of Lipschitzian function is meaningful.

The main advantage of supposing an objective function to be Lipschitzian is that it let us exclude big regions of X surrounding examined points, with larger regions around points associated with lower values. This can be done computing the upper bound (or the lower one if we want to minimize the function) of the values that the objective function can assume and then examining only the points than can potentially lead to an improvement. For example, if we consider $f : ([0, 1]^D, \|\cdot\|_\infty) \to (\mathbb{R}, \|\cdot\|_\infty)$ we can proceed with a grid search that requires at most $\lceil L/\epsilon \rceil^D$ function evaluations. This could be excessive in many cases, in fact even for relatively "good" values as $L = 1$, $\epsilon = 0.1$, $D = 10$ we could need as much as 10^{10} function evaluation, but in many cases this is a good starting point.

Other limitations of this method are

- it does not account for non deterministic values
- it does not give additional information to guide the search
- it could be hard to find a good value for the Lipschtiz constant
- if we underestimate the Lipschtiz constant, we could permanently exclude the optimal point

2.2 Gaussian Processes

The previous Lipschitzian hypothesis is not always sufficiently good to guide the search, also because it lacks flexibility and a probabilistic interpretation. Moreover the hard bound

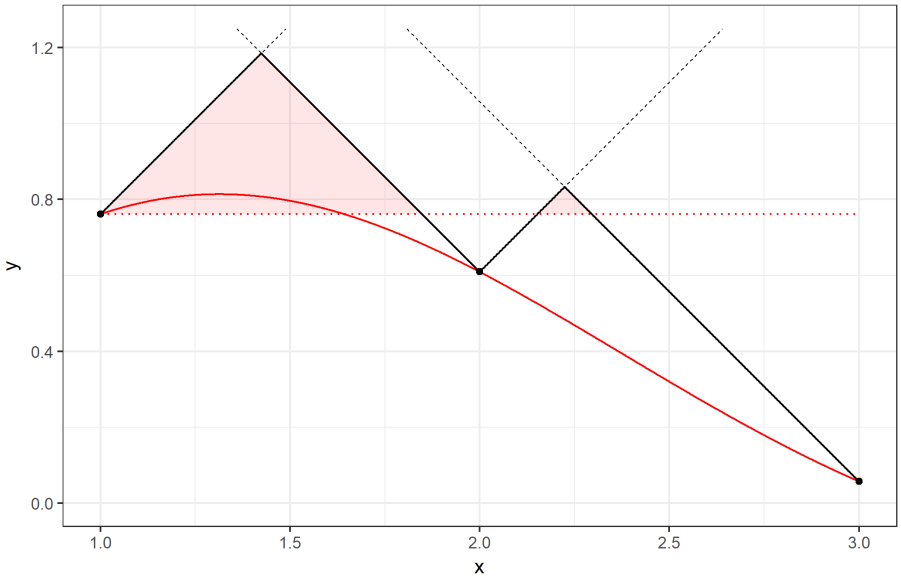


Figure 2.1: Example of Lipschitzian function sampled in three points, and the relative upper bound for $L = 1$. The portion in red is the area where an improvement can happen, therefore only x values relative to it have to be evaluated.

imposed by the choice of the Lipschitzian constant can potentially lead to the exclusion of potentially good solutions.

An idea to solve this issue could be to relax the Lipschitzian hypothesis from a deterministic rule to a probabilistic one, to model not only some bound, but the distribution of the function values in all the space points. To do so we need to define a *stochastic process*

Definition 1 (Stochastic process)**.** *A stochastic process is a collection of random variables defined on the same probability space* $(\Omega, \mathbf{F}, \mathbb{P})$ *indexed by a set* I*, all taking values in a common space* S*, that has to be measurable with respect to a* σ*-algebra* Σ*.*

We will refer to the value at the points x of the stochastic process g as g_x or $g(x)$.

If we want to describe the behavior of a function $f : \mathcal{X} \to \mathbb{R}$ from a probabilistic point of view we can use a stochastic process with values in $\mathbb{R}$ indexed by $\mathcal{X}$. This let us to describe the probabilistic distribution of the value of the function in a given point and also to characterize the correlation between the value in different points; the latter is especially useful to describe the link between the known data and the unknown values not yet evaluated.

It is possible to define an immense variety of stochastic processes but we will focus on subset that exhibits nice properties, *Gaussian processes*.

Definition 2 (Gaussian process)**.** *A Gaussian process* $\mathbf{g}$ *is a stochastic process with values on* $\mathbb{R}$ *indexed by* $\mathcal{X}$ *such that for each finite number of values* $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathcal{X}$*,* $[g_{\mathbf{x}_1}, ..., g_{\mathbf{x}_n}]^T$ *is distributed according to a multivariate normal with mean* $[m(\mathbf{x}_1), ..., m(\mathbf{x})]^T$ *and variance-covariance matrix* K *such that* $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$*, where* $m : \mathcal{X} \to \mathbb{R}$ *is called mean function and* $k : \mathcal{X}^2 \to \mathbb{R}$ *is called covariance function.* k *must be chosen in such a way that the matrix* K *is always symmetric and positive definite.*

Gaussian processes derive many nice properties from the multivariate normal distribution, for example is possible to compute the conditional distribution using the following lemma.

Lemma 1. *Let* $\mathbf{z}$ *be a multivariate normal partitioned in* $\begin{bmatrix}\mathbf{z}_1 \\ \mathbf{z}_2\end{bmatrix}$ *with mean* $\mu = \begin{bmatrix}\mu_1 \\ \mu_2\end{bmatrix}$ *and variance-covariance matrix* $\Sigma = \begin{bmatrix}\Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22}\end{bmatrix}$*, then* $(\mathbf{z}_2|\mathbf{z}_1 = a)$ *(the conditional distribution of* $\mathbf{z}_2$ *given* $\mathbf{z}_1 = a$*), where* $a \in \mathbb{R}$*, is a multivariate normal with mean* $\mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(a - \mu_1)$ *and variance-covariance matrix* $\Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}$*.*

For simplicity in the following part we will work with matrices containing the data, in particular an observation on each row. We will also use a similar notation for random variables representing unobserved quantities.

The lemma 1 let us to infer the value of the Gaussian process given some observations. For examples, supposing $\mathcal{X} \subseteq \mathbb{R}^p$, if we have a set of n observations $\mathcal{D} = (\mathbf{X}, \mathbf{Y})$ of a Gaussian process g, where $\mathbf{X} \in \mathbb{R}^{n \times p}$ and $\mathbf{Y} \in \mathbb{R}^{n \times 1}$ (they are matrices with observations on each row) for each set of points defined by a $n_2 \times p$ matrix $\mathbf{x}$, we can infer the distribution of $g(\mathbf{x})$. In fact, with a slight abuse of notation the mean and the variance-covariance matrix can be expressed as follows

$$\mu_{g|\mathcal{D}}(\mathbf{x}) = m(\mathbf{x}) + k(\mathbf{x}, \mathbf{X})k^{-1}(\mathbf{X}, \mathbf{X})(\mathbf{Y} - m(\mathbf{X}))$$
$$\sigma_{g|\mathcal{D}}(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - k(\mathbf{x}, \mathbf{X})k^{-1}(\mathbf{X}, \mathbf{X})k(\mathbf{X}, \mathbf{x})$$

In theory, if we know m and k this is all we need to know to make inferences from observations. Actually computing the value of $\mu_{g|\mathcal{D}}$ and $\sigma_{g|\mathcal{D}}$ is not trivial for data set that are not really small or that exhibit strongly correlated data (as a point of $\mathcal{X}$ sampled more than once); this difficulty arises since it is necessary to compute the inverse of $k^{-1}(\mathbf{X}, \mathbf{X})$. To solve this issue techniques of linear algebra such as the *Choleski* or *singular value decomposition* (or the truncated SVD [4]) can be used, but we omit the details. However for the resolution on big amounts of data custom methods are necessary, such as presented in [25].

The process of inference can be seen as defining a new Gaussian process with different mean and covariance functions, where $m_{\mathcal{D}}(\cdot) = \mu_{g|\mathcal{D}}(\cdot)$ and $k_{\mathcal{D}}(\cdot) = \sigma_{g|\mathcal{D}}(\cdot)$. This new

process encodes the data in the modified mean and covariance functions and can be seen as a standard Gaussian process.

The inference method defined so far is enough in many cases, but often we cannot directly sample from the objective function; for example the result of a computer simulation could depend on a set of pseudo-random number. In many circumstances we can suppose the randomness of the sampling to be equivalent to additive normal noise. In this case our data set is not directly composed of values assumed by the Gaussian process but their noisy version. We can express the new model as

$$\begin{aligned} \mathbf{y} &\sim \mathcal{N}(g(\mathbf{x}), \sigma^2(\mathbf{x})) \\ g &\sim \mathcal{GP}(m, k) \end{aligned} \tag{2.2}$$

Often the noise is supposed to be independent homoscedastic, i.e. independent from $\mathbf{x}$.

Obviously in many cases the noise can also be non Gaussian or non additive. Common alternatives are the Bernoulli, binomial and Poisson distributions for the discrete cases, or student-t for the continuous ones. In this case the Gaussian process is used as a latent process that model the parameter of the distribution, similarly to a *generalized linear model*. In this situation it is not obvious how to do inference and usually approximated method have to be used. In literature many methods have been proposed to approximate the posterior distribution of the model; the most widely used are Laplacian approximation, KL minimization, expectation maximization and Monte Carlo sampling. Each method gives different results with different characteristics; for example the Laplacian approximation tries to match the maximum of the true posterior with a multivariate normal likelihood.

In the case of a Gaussian process with additive Gaussian noise direct inference can be used, in fact is sufficient to modify the covariance function in order to include the noise. Again this result directly descents from an analytic property of the Gaussian distribution, the additivity. In fact if the noise has zero mean and variance σ^2 the prior distribution of the i^{th} observation y_i is

$$y_i \sim \mathcal{N}(m(x_i), \sigma^2 + k(x_i, x_i)) \tag{2.3}$$

Since the noise is distributed with zero mean the prediction mean for y_i and $g(x_i)$ is the same but the variance is generally different, in fact the variance of the distribution of y_i is greater or equal than the variance of the distribution of f_i. This let us define two kinds of posterior intervals, *predictive intervals* and *credible intervals*.

For $\alpha \in (0, 1)$ the α-credible interval is the interval of $\mathbb{R}$ centered in the prediction value such that the integral of the posterior distribution of the Gaussian process has a a value of

α. Since the posterior distribution is normally distributed this is equivalent to the interval defined by $\mu_{g|\mathcal{D}}(\cdot) \pm w_\alpha \sigma_{g|\mathcal{D}}(\cdot)$ where w_α is an opportune coefficient $\in (0, \infty)$.

For $\alpha \in (0, 1)$ the α-prediction interval is the interval of $\mathbb{R}$ centered in the prediction value such that following the posterior distribution a new sample y would have a probability of α of being inside of. Again, since the posterior distribution is normally distributed this is equivalent to the interval define by $\mu_{g|\mathcal{D}}(\cdot) \pm w_\alpha(\sigma_{g|\mathcal{D}}(\cdot) + \sigma(\cdot))$.

Now we know how to make inference from data given a Gaussian process defined by a mean function, a covairance function and eventually a sample noise variance (supposed constant across all the space $\mathcal{X}$). This is a good result, but in the vast majority of applications, we know none of the precedents. Usually the mean function can be supposed to be identically zero, since the method will adapt to different values (at least locally) even with few observations; this is especially true if the data are normalized. We could do something similar with the covariance function and the sample noise variance but this could lead to bad performance of this method, over or underestimating both the spatial variance of the function and the noise scale. A better approach could be to estimate the covariance function and the noise level from the data.

2.3 The Choice of the Covariance Function

In the majority of practical application the exact nature of the covariance function is unknown, even if some degree of prior knowledge could be available; generally we don't know the expression but we only expect certain qualitative characteristics, so usually parametric functions are used. The parametric functions used to model the covariance function are generally called *kernel function* (from the Middle English word kernel, meaning core or seed) since they are the central part of a Gaussian process from which the characteristics of the samples arose.

From certain points of view the quality of the results doesn't depend too much on the kernel function, since a Gaussian process always tries to fit the data points. This statement should not let us underrate the importance of selecting the correct kernel function, since the quality of the interpolation, particularly in areas of the space not yet explorated, can depend much on the covariance function, especially in term of variance. Other characteristics that the covariance function can give to the process are periodicity, smoothness (or lack of) and spatial trends, such as linear or quadratic ones.

Obviously the kernel function has to be chosen in such a way that the resulting covariance matrix is symmetric and positive definite, so not any function can be used. In literature a vast number of kernel functions has been proposed; we briefly introduce some of the classical choices:

the *Gaussian kernel* (or *quadratic exponential*)

$$k(x, y) = \sigma^2 \exp(-\rho \|x - y\|_2^2) \tag{2.4}$$

with parameters $\sigma > 0$ and $\rho > 0$;

the *exponential kernel*

$$k(x, y) = \sigma^2 \exp(-\rho \|x - y\|_2) \tag{2.5}$$

with parameters $\sigma > 0$ and $\rho > 0$;

the *periodic exponential kernel*

$$k(x, y) = \sigma^2 \exp(-\rho_1 \sin^2(\rho_2 \|x - y\|_2)) \tag{2.6}$$

with parameters $\sigma > 0$, $\rho_1 > 0$ and $\rho_2 > 0$;

the *dot-product kernel*

$$k(x, y) = \sigma_b^2 + \sigma_v^2 (x - c) \cdot (y - c) \tag{2.7}$$

with parameters $\sigma_b > 0$, $\sigma_v > 0$ and $c \in \mathbb{R}^n$;

the *Matérn kernel*

$$k(x, y) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\|x - y\|_2}{\rho} \right)^{\nu} K_\nu \left(\sqrt{2\nu} \frac{\|x - y\|_2}{\rho} \right) \tag{2.8}$$

with parameters $\sigma > 0$, $\nu > 0$ and $\rho > 0$ and where *Gamma* is the gamma function and K_ν is the modified Bessel function of the second kind.

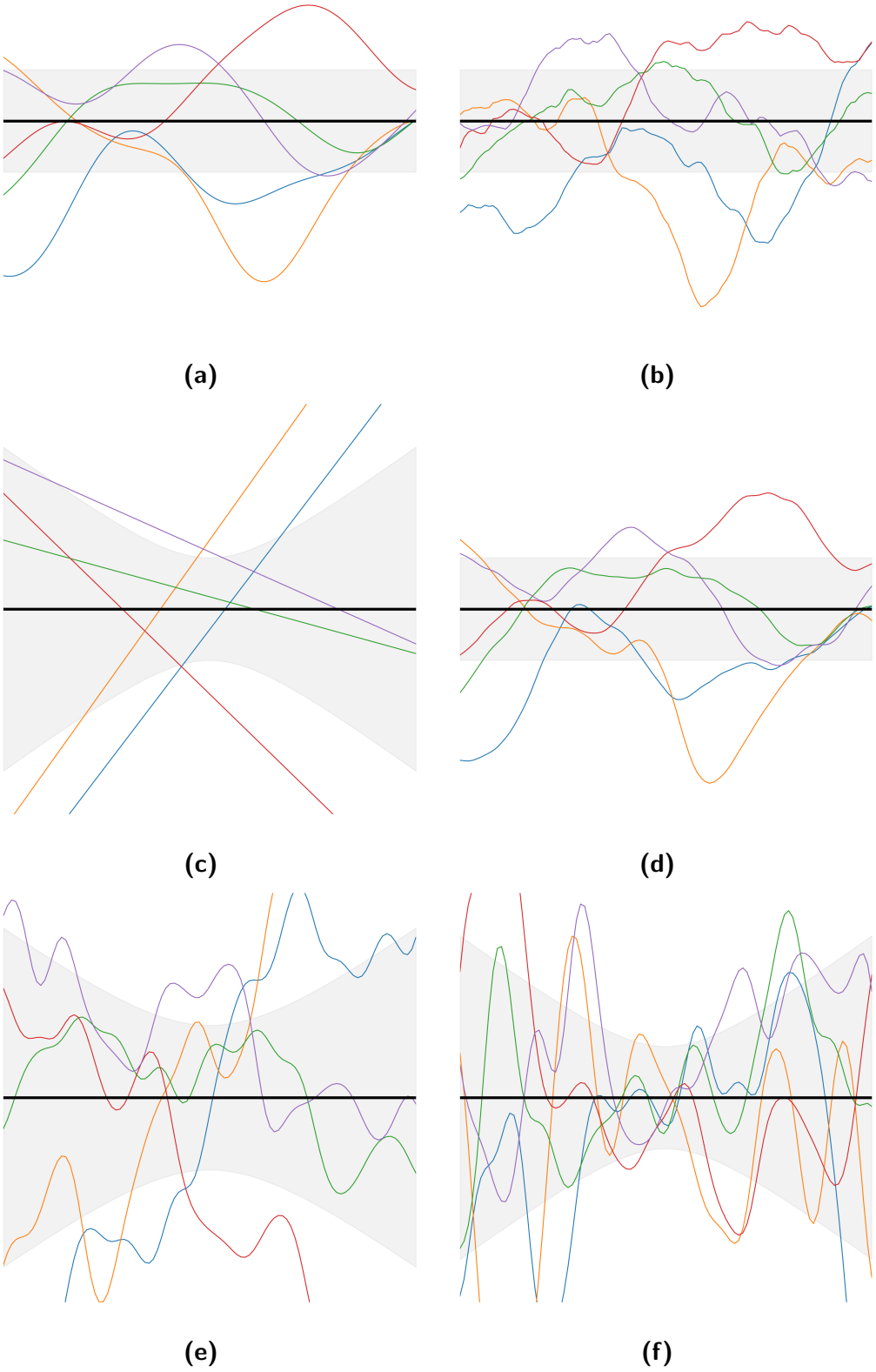


Figure 2.2: Some samples from the prior distribution given by: a Gaussian kernel (2.2a); a Matern kernel with $\nu = 1.5$ (2.2b); a dot product kernel (2.2c); a Matern kernel with $\nu = 2.5$ (2.2d); a sum kernel obtained from a Gaussian kernel and a dot product kernel (2.2e); a product kernel obtained from a Gaussian kernel and a dot product kernel (2.2f).

There are many possible ways to insert prior knowledge in the kernel function. A first way is to impose a prior distribution over the parameter space to encourage some characteristics of the kernel. Another approach is to hand craft special kernel to incorporate the expected structure of the space. For example is possible to combine different kernels to increase the complexity and join their characteristics; possible binary operator between kernels are the addition, the multiplication and the exponentiation. Another possibility is to choose on which dimensions of the input space each kernel has effect. Using these operations we can obtain arbitrarily complex kernel function combining the characteristics of many simpler kernel, furthermore we can let the inference method chose the one that is more adapted to the data, decreasing the magnitude or augmenting the length scale of the less suitable.

Now the only thing that we have to do is to estimate the parameters of the covariance function.

2.4 Hyperparameters Estimation

Once we have chosen a kernel function to model the covariance function we have to find a way to translate the data in the posterior distribution. In other words, we have to find a way to estimate the distribution of the process in a point given the data observed so far.

A fully Bayesian approach to make predictions is to compute the posterior distribution of the process including the uncertainty of the kernel's parameters. This method is generally expensive because it can not be treated analytically and approximated methods such as Monte Carlo must be used. The loss of the analytic properties is given by the fact that generally the posterior distribution is no more normal and therefore closed form inference is not possible; there are exceptions, for example in [11] is proven that in some cases the posterior distribution is distributed along a student-t and therefore exact inference is possible. The main advantage of this kind of methods is that it isn't necessary to chose a single set of parameters and therefore there is not the risk of getting stuck in a local maxima of the kernel coefficients or to underestimate the uncertainty of the prediction due to a non well localized likelihood distribution when the data don't bring much information.

However if we want to make predictions faster or in a simpler way, we could drop the fully Bayesian approach using another common method, the *maximum likelihood estimation* (MLE). We could use a maximum likelihood estimation method to find a suitable set of parameters for the kernel, fix them and then computing the posterior distribution. In this way the value of parameters can be found just once, while the predictions can be made for arbitrarily many points, requiring less computation time. Moreover the posterior in this way is tractable and normally distributed, so the inference is relatively fast. The optimal set of parameters can be found using standard optimization techniques, sometimes derivative free optimization methods such as the *Nelder-Mead algorithm* [18] are used, but

since the derivative of the likelihood with respect to the parameters are generally available is also possible to use optimization algorithms that require the derivative of the objective function such as the *BFGS algorithm*.

In the case of a standard Gaussian process the likelihood with observations $\mathcal{D} = (\mathbf{X}, \mathbf{Y})$ the likelihood can be derived directly from the equation 2.3

$$\mathcal{L}(\theta|\mathbf{X}, \mathbf{Y}) = det(2\pi K_\theta)^{-\frac{1}{2}} \exp\{-\frac{1}{2}(\mathbf{Y} - m(\mathbf{X}))^T K_\theta (\mathbf{Y} - m(\mathbf{X}))\} \tag{2.9}$$

where $K_\theta = k(\mathbf{X}, \mathbf{X}|\theta)$ is the covariance matrix obtained computing the kernel function with parameters θ for each pair of points in $\mathbf{X}$ and $m(\cdot)$ is the mean function.

We can consider the log-likelihood to simplify this expression

$$\log \mathcal{L}(\theta|\mathbf{X}, \mathbf{Y}) = -\frac{n}{2} \log(2\pi \det(K_\theta)) - \frac{1}{2}(\mathbf{Y} - m(\mathbf{X}))^T K_\theta (\mathbf{Y} - m(\mathbf{X})) \tag{2.10}$$

where n is the number of observations. Evaluating $\log \mathcal{L}$ can be relatively costly and numerically difficult due to the determinant and the inverse of K_θ.

In other cases these methods are not feasible or even possible, for example in the case of non normally distributed noise. Other methods that have been used in the literature are for example based on variational inference, expectation maximization and Laplace approximation, but the analysis of this methods exceed the aim of this work.

2.5 Multivariate Gaussian Processes

Is possible to extend the Gaussian Process to functions with multiple outputs. In the simplest case we can fit a model for each dimension but this does not consider correlations between one dimension and another, leaving out some potential information.

The main idea of a *multivariate Gaussian process* is that a multivariate function can be converted into a univariate one if we add an ulterior input, the index of the desired output dimension; for example $f : \mathbb{R} \to \mathbb{R}^n$ can also be seen as a function $\hat{f} : \mathbb{R} \times \{1, ..., n\} \to \mathbb{R}$. Since we have never required the input space to be continuous the new function is suited to be used for a Gaussian process. This can be implemented using an appropriate covariance function, such that it takes into account the index of the desired output, i.e. we want to use a kernel in the form $k(x_1, x_2, i_1, i_2)$ where i_1 and i_2 are the indexes of the output dimensions we are considering. This kind of kernel can be hard to work with, so simplified versions are often used.

A relatively general approach is the *"Linear Model of Coregionalization"*. The main idea of this method is to see each dimension of the target multivariate process as a linear combination of a set of latent independent stochastic processes, i.e.

$$g_i(\mathbf{x}) = \sum_{k=1}^{K} \alpha_k^i u_k(\mathbf{x}) \tag{2.11}$$

where g_i is the i^{th} dimension of the target process and u_k is the k^{th} latent process. Each latent process is supposed to be distributed according to a Gaussian process, and often the same covariance function is shared between different latent function, therefore considering the independence of the processes the covariance function can be expressed as

$$\begin{aligned} K_{ij}(\mathbf{X}_1, \mathbf{X}_2) &= Cov[g_i(\mathbf{X}_1), g_j(\mathbf{X}_2)] = \\ &= Cov[\sum_{k=1}^{K} \alpha_k^i u_k(\mathbf{X}_1), \sum_{h=1}^{K} \alpha_h^j u_h(\mathbf{X}_2)] = \\ &= \sum_{k=1}^{K} \sum_{h=1}^{K} \alpha_k^i \alpha_h^j Cov[u_k(\mathbf{X}_1), u_h(\mathbf{X}_2)] = \\ &= \sum_{k=1}^{K} \alpha_k^i \alpha_k^j Cov[u_k(\mathbf{X}_1), u_k(\mathbf{X}_2)] \end{aligned} \tag{2.12}$$

If the latent processes can be grouped in Q groups with the same covariance function in each group it can be simplified as

$$K_{ij}(\mathbf{X}_1, \mathbf{X}_2) = \sum_{q=1}^{Q} b_{ij}^q k_q(\mathbf{X}_1, \mathbf{X}_2) \tag{2.13}$$

where b_{ij}^q is an opportune coefficient and k_q is the q^{th} kernel function.

Once that for a given set of points the variance-covariance matrix is defined, the inference process works in the same way as in the basic case.

2.6 Heteroschedastic Gaussian Process

Previously we supposed the noise to be independent and identically distributed with constant variance across all the space, but this is a strong hypothesis and in many applications it is simply not true, so if the approximating deriving from supposing the noise homoschedastic is too onerous we could need a model describing how the variance of the noise varies across the space.

In many cases the predictive results with the homoschedastic hypothesis are sufficiently good in term of punctual estimation, but may greatly fail in estimating the distribution of the posterior probability. However it is possible to generalize the model to take in account the heteroschedasticity of the data modeling the variance of the noise with an ulterior Gaussian process. The solution of the new model is more complicate that the solution in the homoschedastic case, but many strategies have been proposed in literature, for example in [19], [20], [21] and [22].

$$\begin{aligned} \mathbf{y} &\sim \mathcal{N}(g(\mathbf{x}), h(z(\mathbf{x}))) \\ g &\sim \mathcal{GP}(m, k) \\ z &\sim \mathcal{GP}(m_z, k_z) \end{aligned} \tag{2.14}$$

where $h : \mathbb{R} \to \mathbb{R}^+$ is a function that map $\mathbb{R}$ to the positive numbers; common examples are $\exp(\cdot)$ and $(\cdot)^2$.

2.7 Gaussian Processes with Non-Gaussian Noise

In many cases the available data are not normally distributed, so the hypothesis of Gaussian noise is not applicable; an example could be a spatial counting problem where the observations of the process are distributed according a Poisson distribution, therefore in $\mathbb{N}$.

Similarly to what append in a *generalized linear model*, we can suppose that our Gaussian process is a latent factor influencing the distribution of the observations; in the case of Poisson observations we could try to use a Gaussian process to model the behavior of the unobservable parameter of the Poisson distribution $\lambda(\cdot)$ that varies depending on the space point on which the observation is made. As we did in the case of the standard deviation of a heteroschedastic Gaussian process, we need a non-linear link function to map from $\mathbb{R}$ to $\mathbb{R}+$. Again, common choices are $\exp(\cdot)$ and $(\cdot)^2$.

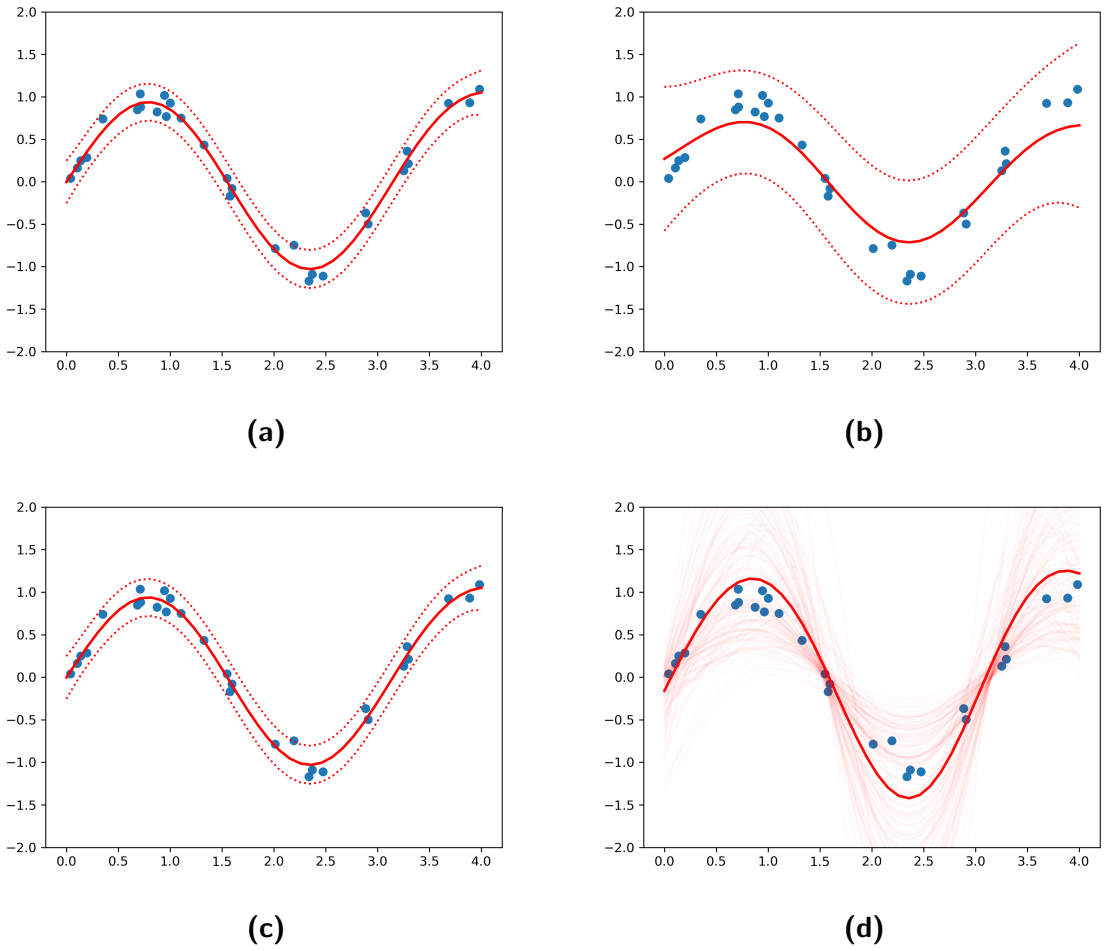
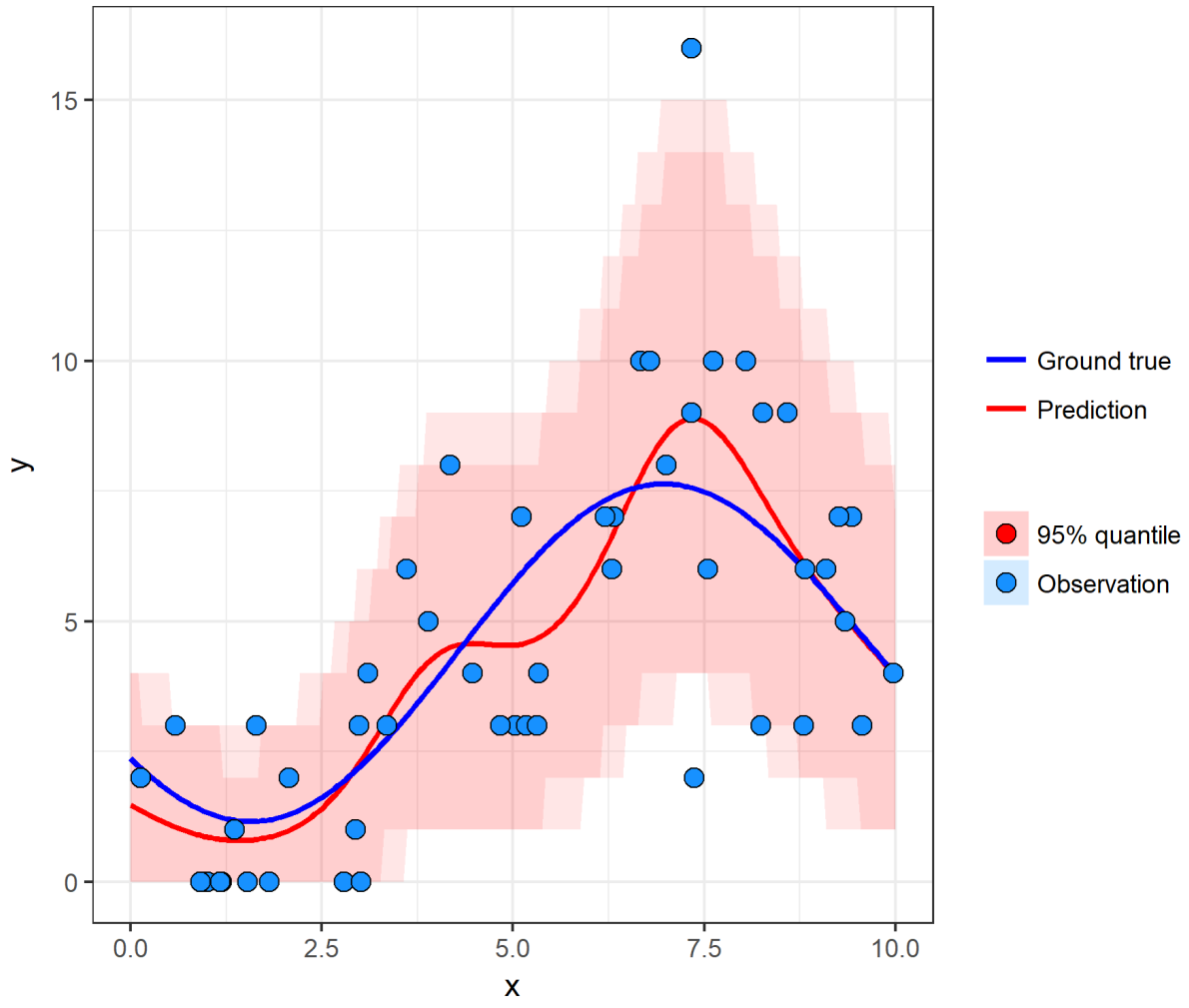


Figure 2.3: Example of the result obtained with maximum likelihood estimation with 2 standard deviation bar (2.3a), expectation propagation with 2 standard deviation bar (2.3b), Laplace approximation with 2 standard deviation bar (2.3c), Monte Carlo Bayesian inference method showing 100 observations (2.3d)



(a)

Figure 2.4: A Gaussian process (the mean, the 0.025 and 0.05 quantiles) fitted with a Laplace method on 50 observations obtained from a Poisson distribution with parameter varying with respect to the space position; in blue the true function (2.4a).

Chapter 3

Application to Worst Case Optimization

In this chapter we present some of the methods that use Gaussian processes to guide the optimization process of a function that is, in some sense, expensive to evaluate. This is the case of numerical simulations, physical experiments etc. In many cases the evaluation process is possible only with a non-negligible noise, so it is necessary to take it in account.

The main objective of the fitted Gaussian process is to provide a strategy to choose the point were we want to draw the next function evaluation; a secondary one is to provide a way to approximate the expected value of the function even when only one or few observations are available for a given point.

Often the branch of optimization that uses this kind of methodologies is called *Bayesian optimization*, since the Gaussian process can be interpreted as a prior distribution of the objective function over an opportune space of functions.

3.1 Efficient Global Optimization

Efficient global optimization (EGO) is the name given to the iterative method of optimization that uses a Gaussian process, or a variant of it, as a surrogate model.

Suppose that we want to maximize a function $f : \mathcal{X} \subseteq \mathbb{R}^n \to \mathbb{R}$; note that we are not requiring $\mathcal{X}$ to be continuous and therefore, for example, $\mathbb{N}^2 \times \mathbb{R}^{n-2}$ fits our requirements if $n \geq 2$. This function is supposed to be expensive (in some arbitrary meaning) to evaluate or even completely impossible to. In fact, sometimes, only a noisy version of it can be evaluated; a classical example is the presence of additive noise with zero mean. We will generally suppose the noise to be additive and independently identically distributed according to a zero mean normal distribution.

Given a set of p observations $\mathcal{D} = (\mathbf{X}, \mathbf{Y})$, where $\mathbf{X} \in \mathbb{R}^{m \times p}$ is the matrix containing the predictor variables and $\mathbf{Y} \in \mathbb{R}^p$ is the response variable vector, we want to find the observation point that correspond to the best value of the objective function. In the matrix of observations, we suppose to have each observation on a different row and we will indicate with X_i or $X_{i\cdot}$ the i^{th} observation of the predictor variable. Analogously Y_i correspond to the i^{th} observation of the response variable.

Now, how can we choose which observation maximize the objective function? If the sampling method is noiseless, we can simply choose the observation that maximize Y_i, otherwise we need a method to estimated the objective function. In fact, if the response variable doesn't identically correspond to the value of the objective function point by point, we have no guarantee that the point maximizing one is also the point maximizing the other. The method have to be carefully chosen and should consider some kind of correlation between different observations, since we could have one or few ones for each point. A possible approach can be to fit a Gaussian process $g|\mathcal{D}$ on the observations, then its predictions can be used as an approximation of the objective function. Once that one of the observations is chosen as *current solution*, we can do two things: choose to take another sample in a query point in the hope of finding a better solution, or stop the optimization process. The latter possibility doesn't need many explanations, but the former is not obvious. What we need is a strategy to choose a sampling point given the current observations; some methodologies even use more than one query point per time. After that another observation is taken, the process can restart.

There are many possible approaches to this phase of the process, since it is not completely clear how should be chosen the next query point. Ideally, we would like the next point to be the optimal one, but we also need to know that this point is optimal, or at least good, otherwise we could take the risk of ignoring it. Moreover focusing only on the value of the query point in the next step could be too *greedy* and could lead to bad performance. Ideally we would like to choose a point that improve both the objective function in correspondence of the current solution and our insight of the characteristics of the function. This duality define a general trade-off between the exploration of the search space and the exploitation of good areas of it.

While some approaches try to maximize some information-theoretic measure of our knowledge of the function ([16]) or resort in heavy simulation of the simulation process itself ([24]), but the vast majority of the methods used in the literature define an auxiliary function called *acquisition function* that quantifies, based on some metric, the "goodness" of a particular point and then chooses the point which maximizes such metric as the candidate point to be examined. Some metrics just search a point that has a good probability of improving the current solution, while other look many steps ahead; the former are usually called *non-myopic*.

Many techniques have been proposed, with even more variations depending on the specific case of use, but we will present three of the most common ones. At first we need to the find the *current solution* point and value as the point associated to the best value of the objective function. This can be generally expressed as

$$\begin{aligned} \mathbf{x}_{cs} &= arg \max_{\mathbf{x}_i \in \mathbf{X}} \mu_{g|\mathcal{D}}(\mathbf{x}_i) \\ y_{cs} &= \mu_{g|\mathcal{D}}(\mathbf{x}_{cs}) \end{aligned} \tag{3.1}$$

where $\mu_{g|\mathcal{D}}(\cdot)$ is predictive mean of the Gaussian process fitted on the observations. It could also be possible to choice a point that has not been sampled (yet), but sometimes is safer to choose only between sampled points to larger avoid estimation errors associated with unexplored ares.

A first criterion for the acquisition function is the *Probability of Improvement*:
the idea is to search the point that, according to the Gaussian process model, has the maximum probability of being associate with a value of the objective function greater than the current one. Therefore the acquisition function is defined as follows

$$\alpha_{pi}(\mathbf{x}) := \mathbb{P}[g|\mathcal{D}(\mathbf{x}) > f_{cs}] \tag{3.2}$$

The idea beyond this method is pretty simple and at first sight it could seem to be the best possible criterion, but it has the tendency to propose points that are near to the current solution, so it is necessary to couple it with techniques that encourage the exploration of the search space.

A second method is the *Upper Confidence Bound*:
in this case the idea is to maximize the α-quantile of the posterior distribution of the Gaussian process, where $\alpha \in (0, 1)$ is a parameter that handles the trade-off between the exploration and the exploitation of the search space. In case of maximization it can assume a value between 0.5 and 1, because otherwise the quantile would be smaller than the mean (in the case of a normal distribution). Greater values favor the exploration over the exploitation.

A third approach is the *Expected Improvement*:
the *acquisition function* is the expected value of the improvement of the Gaussian process between the analyzed point and the current solution. In case of maximization it can be expressed as:

$$\alpha_{ei}(\mathbf{x}) := \mathbb{E}\big[[g|\mathcal{D}(\mathbf{x}) - f_{cs}]^+\big] \tag{3.3}$$

where $[\cdot]^+$ represent the positive part function. In the case of Gaussian processes the posterior is normally distributed, therefore the expected improvement can be expressed as follows

$$\alpha_{ei}(\mathbf{x}) = (\mu_{g|\mathcal{D}}(\mathbf{x}) - y_{cs})\Phi\left(\frac{\mu_{g|\mathcal{D}}(x) - y_{cs}}{\sigma_{g|\mathcal{D}}(\mathbf{x})}\right) + \sigma_{g|\mathcal{D}}(\mathbf{x})\phi\left(\frac{\mu_{g|\mathcal{D}}(\mathbf{x}) - y_{cs}}{\sigma_{g|\mathcal{D}}(\mathbf{x})}\right) \tag{3.4}$$

where $\mu_{g|\mathcal{D}}(\cdot)$ and $\sigma_{g|\mathcal{D}}(\cdot)$ are the mean and the variance respectively of the posterior distribution of the Gaussian process, and Φ and ϕ are CDF and the PDF of a standard normal respectively.

Often the expected improvement is generalized adding a parameter to handle the trade off between exploration and exploitation. One of the most widely known case is the *weighted expected improvement*, where the second term of the equation 3.4 is multiplied by a parameter $w \in (0, \infty)$. Higher values of w move the trade-off toward the exploration, favoring larger potential gains over small but certain ones.

After that the acquisition function is defined, in order to find the next point to be evaluated is sufficient to find the maximum of the acquisition function itself with any standard optimization method. Many acquisition functions are differentiable, but often the computation gradient is relatively complex, so, depending on the details of the case, it could be better to simply numerically approximate the gradient or to use gradient-less optimization techniques. The latter is also the case if the search space is discontinuous.

A common and quite general approach to emphatize the exploration of the search space over the exploitation is to add a small positive number δ to the value of the current solution; this mean that we are interested only in solutions that improve of at least δ the current one, therefore zones that have been widely explored will be more easily discarded. This method has the tendency to concentrate the augmentation function in few zones, while the weighted expected improvement generally diffuses it in larger areas.

This method can not be directly applied to a noisy worst case optimization problem without gross approximations, since the distribution of the minimum of the observations for a fixed point is not normally distributed, even if it is normally distributed for each value of the uncertainty set. We would need to approximate at each step the minimum of the expected values sampling from each function, till reaching a prefixed precision, in this way we can reduce the variance of the estimation of the mean of each function and therefore the distribution of the minimum is approximately normal. While there could be smarter methods than simply sampling each function many times, this requires many function evaluations and could be computationally expensive. If there are many functions to be evaluated or if the noise level is particularly high a method that requires less evaluation could be preferable.

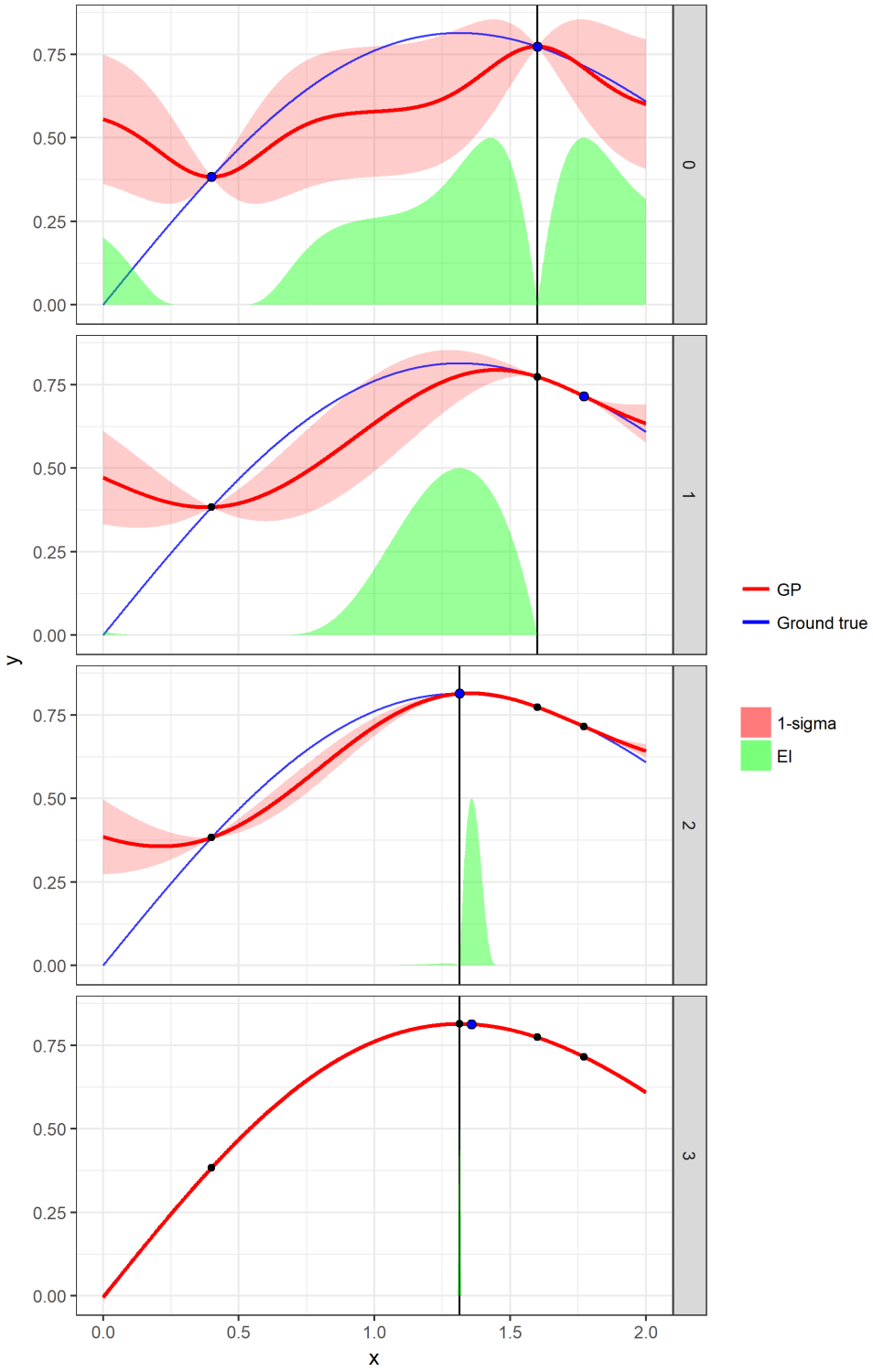
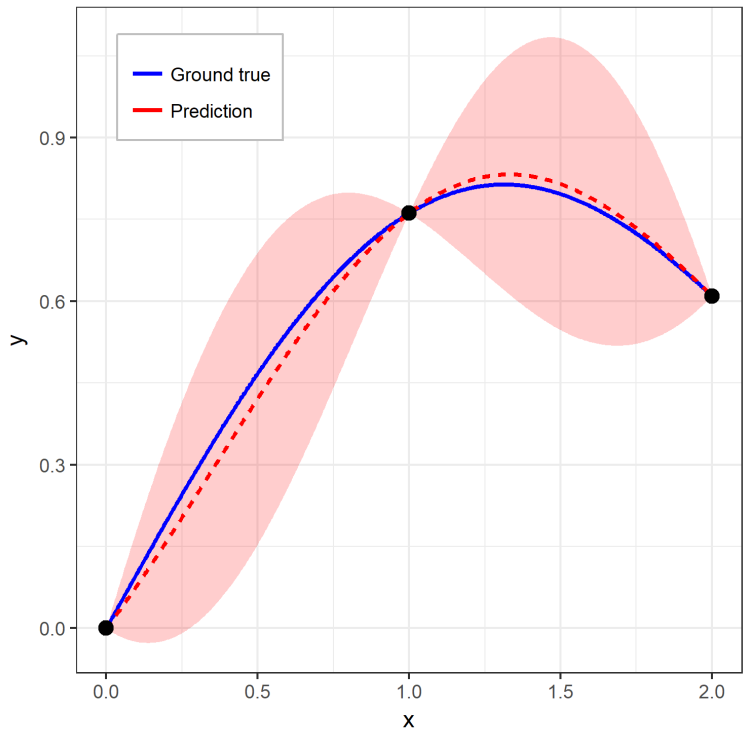
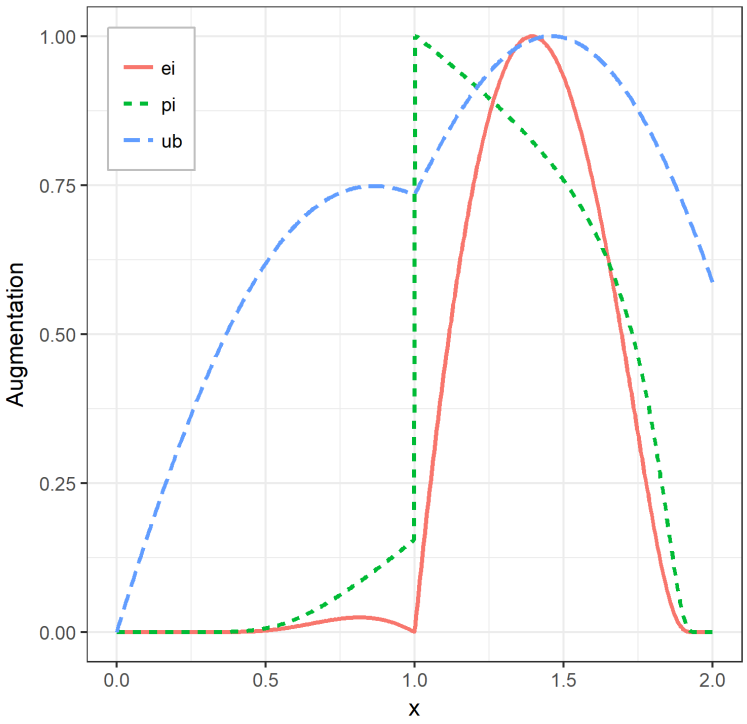


Figure 3.1: An example of the iterative process of optimization using the expected improvement methodology. The vertical line represents the current solution, while the green area is the (normalized) expected improvement augmentation function.

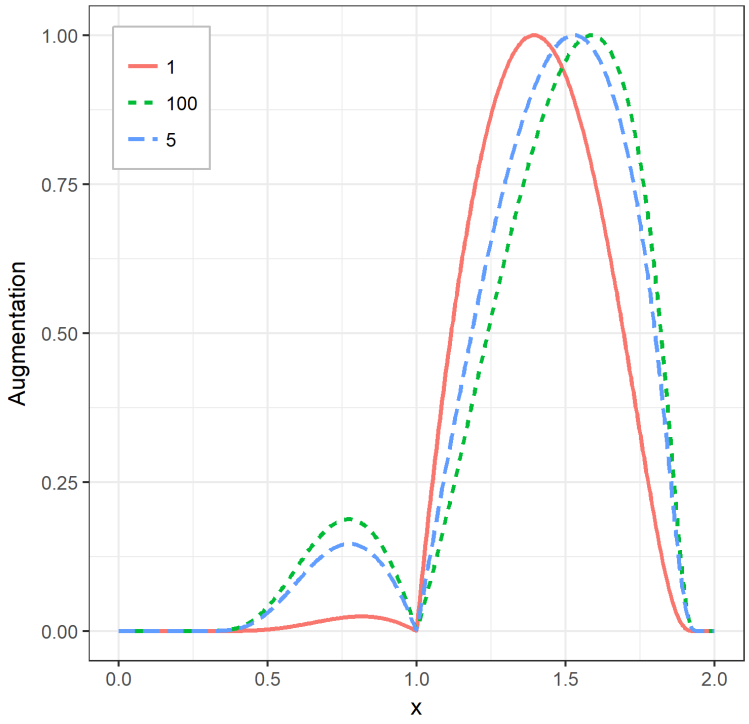


(a)

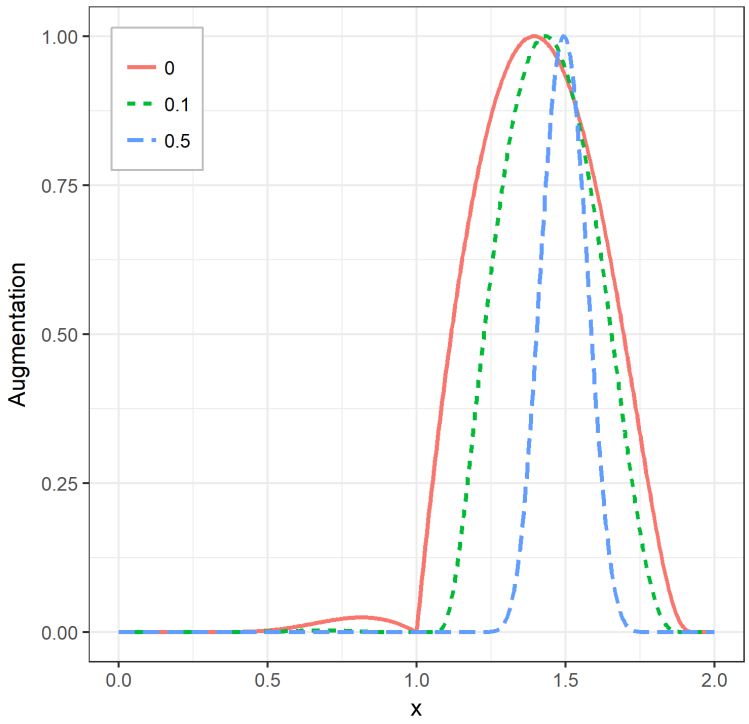


(b)

Figure 3.2: 3.2a a Gaussian process with a one standard deviation band (red) obtained sampling three times the target function (blue). 3.2b the relative augmentation functions, normalized on the interval: upper confidence bound (blue); probability of improvement (green); expected improvement (red).



(a)



(b)

Figure 3.3: The modified expected improvement functions obtain in the case of: weighted expected improvement (3.3a) ; adding delta the current solution value (3.3b). Both the cases refers to the Gaussian process shown in (3.2a).

3.2 Worst Case Efficient Global Optimization

In the framework of the EGO algorithm we need to directly evaluated the objective function, or at least we need to gather noisy observations of it. In the case of worst case optimization this is not easy, because we need to evaluate the minimum of the expectations for each value in the uncertainty set, so various sample are necessary to obtain a correct estimation. Moreover even if the noise on each function is normally distributed, the estimation of the minimum is not normally distributed itself and could be highly skewed, therefore a particular attention is needed. To improve the EGO algorithm we could try to reduce the number of evaluations necessary at each step decoupling the functions for different values of the uncertainty set.

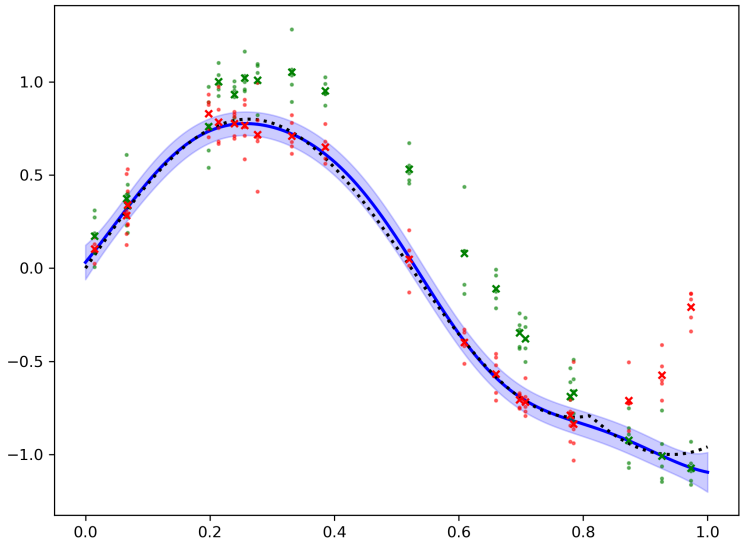
For example we could fit a multivariate Gaussian model or n independent Gaussian processes (or even many multivariate Gaussian processes of reduced size, grouping the elements of the uncertainty set in blocks) and then analyze the behavior of the minimum between the models. The inference phase is identical to the standard case (simply repeated many times), but the posterior distribution of the minimum can not be trivially evaluated. The main problem of making inference in this case is that the distribution of the minimum (or maximum) of n dependent or independent Gaussian random variables is not generally known; there are analytic expressions for the PDF and CDF, but the mean and the variance can only be estimated. So we could extend the augmentation functions only by sampling on both the current solution and the query point. This could be sub-optimal, since it would be more costly and in many cases the derivative of the augmentation function would be not available.

We suppose to have a set of functions $\{f_u\}_{u\in\mathcal{U}}$, where $\mathcal{U}$ is a set indexing the function set; clearly in our case it is the uncertainty set. We want to maximize $\min_{u\in\mathcal{U}} f_u(\cdot)$. We also suppose to start with a set of observations for each function $\mathcal{D} = \{(\mathbf{X}_u, \mathbf{Y}_u)\}_{u\in\mathcal{U}}$, with m_u observation for the u^{th} function. Moreover we denote with $g|\mathcal{D}$ our model (a multivariate Gaussian process or many standard Gaussian processes) and with $g|\mathcal{D}_u$ its u^{th} dimension. Note that in both the cases $g|\mathcal{D}_u$ is a stochastic process indexed by $\mathcal{X}$.

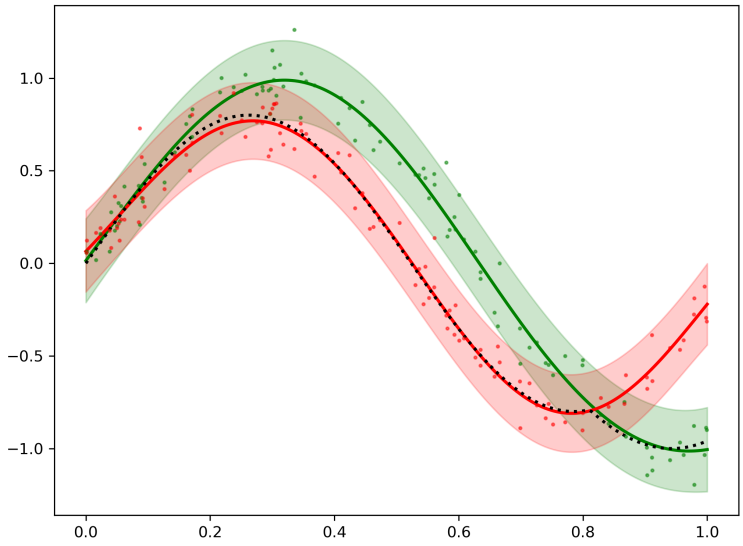
At first we need to define the current solution. We can simply use the minimum of the prediction value of the Gaussian process(es) as follows

$$\begin{aligned} \mathbf{x}_{cs} &= arg\max_{\mathbf{x}\in\mathcal{X}} \min_{u\in\mathcal{U}} \mu_u(\mathbf{x}) \\ y_{cs} &= \min_{u\in\mathcal{U}} \mu_u(\mathbf{x}_{cs}) \end{aligned} \tag{3.5}$$

where μ_u is the u^{th} dimension of the prediction of our model. In the case of a multivariate Gaussian process, it is the u^{th} dimension of the prediction; in the case of many standard Gaussian processes, it is the prediction of the u^{th} Gaussian process. Note that we don't necessarily have a sample for each function in this point, but maybe there is not



(a)



(b)

Figure 3.4: A comparison of the results obtained by the standard EGO algorithm (3.5a) and by the Worst-Case-EGO algorithm (3.4b) with the same number of observations. In this case the standard EGO algorithm uses 5 evaluations per point to estimate the minimum, while the Worst-Case-EGO algorithm uses independent processes. All the sample points are randomly chosen.

such a point.

Similarly to what we did in the standard EGO algorithm, we could try to find the point that maximizes the expected improvement. Is important to underline that in this case the expected improvement cannot be analytically computed, even supposing y_{cs} to be constant and the independence of $g|\mathcal{D}_u$, so approximated methods, like Monte Carlo, have to be used.

Another possibility is to extend the "probability of improvement" criterion to the multivariate case. If we suppose each dimension of our model to be independent, we can write the probability of improvement as:

$$\mathbb{P}[\min_{u\in\mathcal{U}} g|\mathcal{D}_u > y_{cs} + \delta] = \prod_{u\in\mathcal{U}} [1 - \Phi(y_{cs} + \delta|\mu_u(x), \sigma_u(x))] \tag{3.6}$$

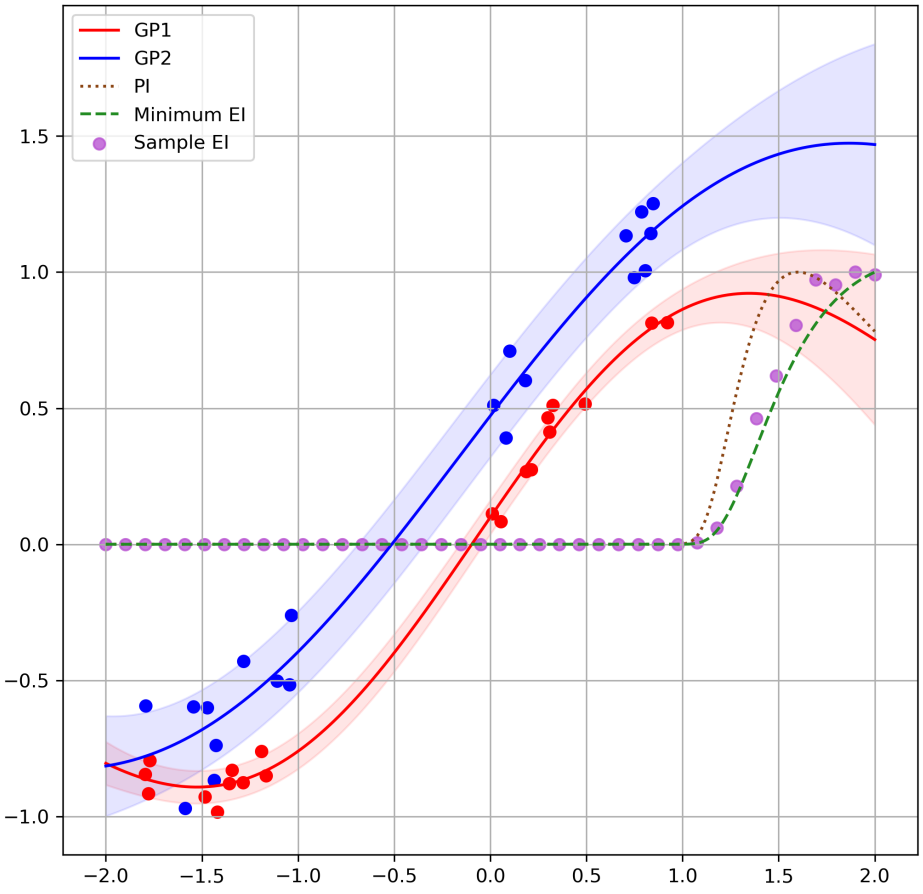
where μ_u and σ_u are the mean and the standard deviation respectively of the posterior distribution of the u^{th} dimension of our model.

Another idea could be to use an augmentation function for each output dimension, then summarize all the obtained values in a single number. Since we are considering the minimum, we are searching a point such that the minimum of the functions is better than the current solution, so we could search the point that maximize the minimum of the augmentation functions. This idea can be summarized as follows

$$\max_{x\in\mathcal{X}} \min_{u\in\mathcal{U}} \alpha_u(x) \tag{3.7}$$

where α_i is the augmentation function applied on the i^{th} dimension. The main advantage of this method is that it can extend almost all the methods used in the standard case.

In figure 3.5 it is possible to see a direct comparison between different augmentation functions, in particular: probability of improvement; minimum expected improvement; expected improvement approximated with a Monte Carlo technique. The qualitative behavior is similar for all of them, but the sample expected improvement seem to be somewhere in the between of the other two.



(a)

Figure 3.5: A comparison between different augmentation functions for multivariate Gaussian processes. There are two independent Gaussian processes, in red and blue respectively, with a band of one standard deviation of the posterior distribution and the relative observations.
Three different augmentation functions (normalized) are also plotted. The *Sample EI* is approximated with a Monte Carlo technique with 5000 samples for each estimation.

Chapter 4

Numerical Experiments

In the previous chapters we have shown how is possible to use Gaussian processes to guide the exploration of the state space in a robust optimization problem. Now we will present some numerical results to test the efficacy of those methods.

4.1 A Simple Benchmark

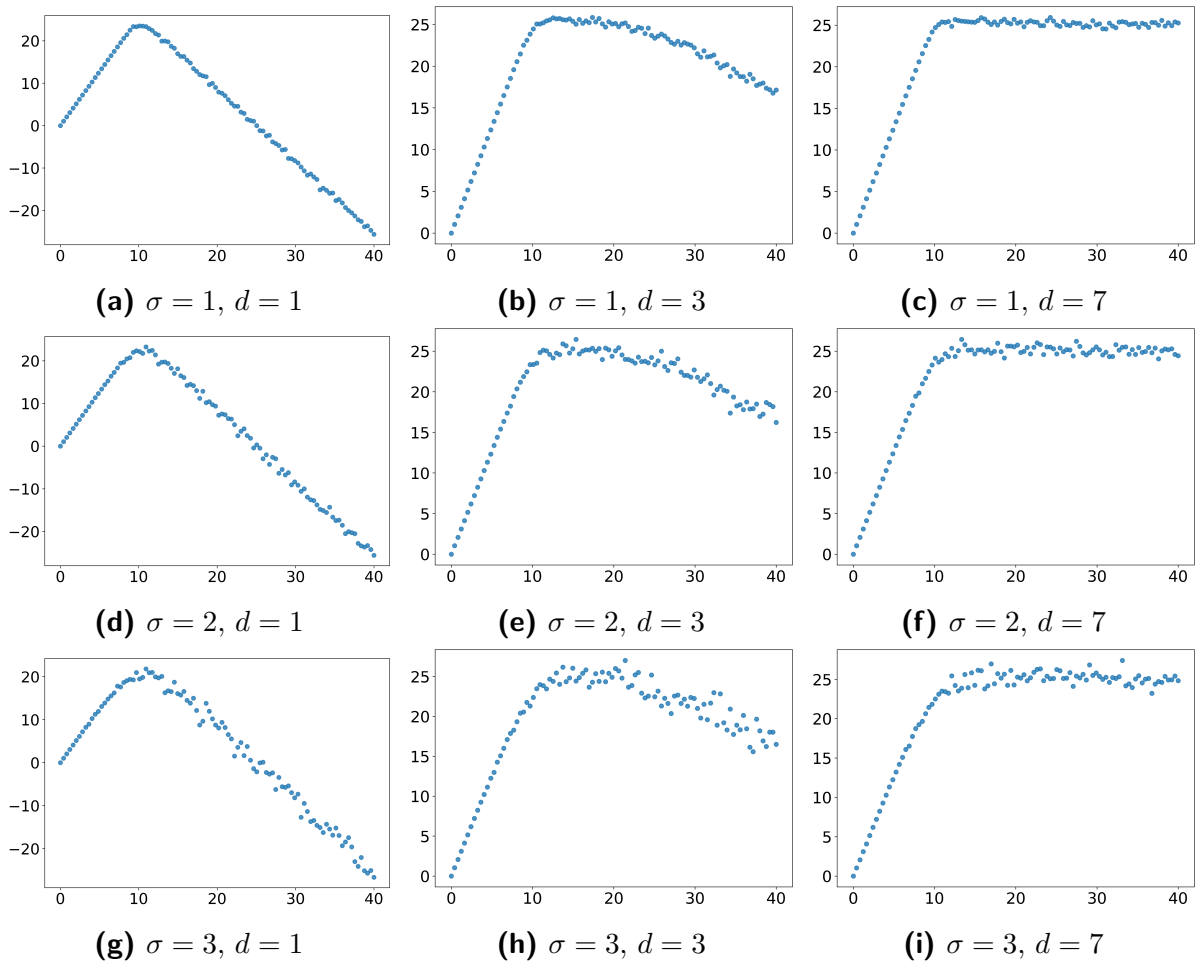
Suppose that we want to sell a perishable product (for example blueberries) with an unknown stochastic demand. We can buy the product at the start of each day to match a prefixed inventory level (optionally depending on the day/week/moth...) at a fixed price, then we sell as many as we can depending on the demand at a fixed price. The demand in different days is supposed to be independent and generally not identically distributed. We want to find the optimal values of the inventory levels that maximize the net revenue. This replenishment strategy is usually called TSL (*target stock level*).

There are many possible scenarios, depending on how many days the goods maintain their proper form, the distribution of the demand (possibly varying seasonally or weekly), how long is the period considered and so on.

We can simulate the behavior of this model with a simple program, in our case implemented in Python 3.

We could approach this problem in a robust fashion for at least a couple of reason: we could have non negligible uncertainty in the estimation of the demand, that is difficult to properly estimate; we can not afford the cost of a potential failure and we are willing to sacrifice some of the potential gains in exchange of the reduction of the risk.

Clearly the decision variables for the optimization problem are the target inventory levels.



(a) $\sigma = 1$, $d = 1$ **(b)** $\sigma = 1$, $d = 3$ **(c)** $\sigma = 1$, $d = 7$

(d) $\sigma = 2$, $d = 1$ **(e)** $\sigma = 2$, $d = 3$ **(f)** $\sigma = 2$, $d = 7$

(g) $\sigma = 3$, $d = 1$ **(h)** $\sigma = 3$, $d = 3$ **(i)** $\sigma = 3$, $d = 7$

Figure 4.1: Some sample from the presented simulation for various combination of parameters. In this case the demand is normally distributed with parameters $\mu = 10$ and different values of σ; the life of the product is d days.

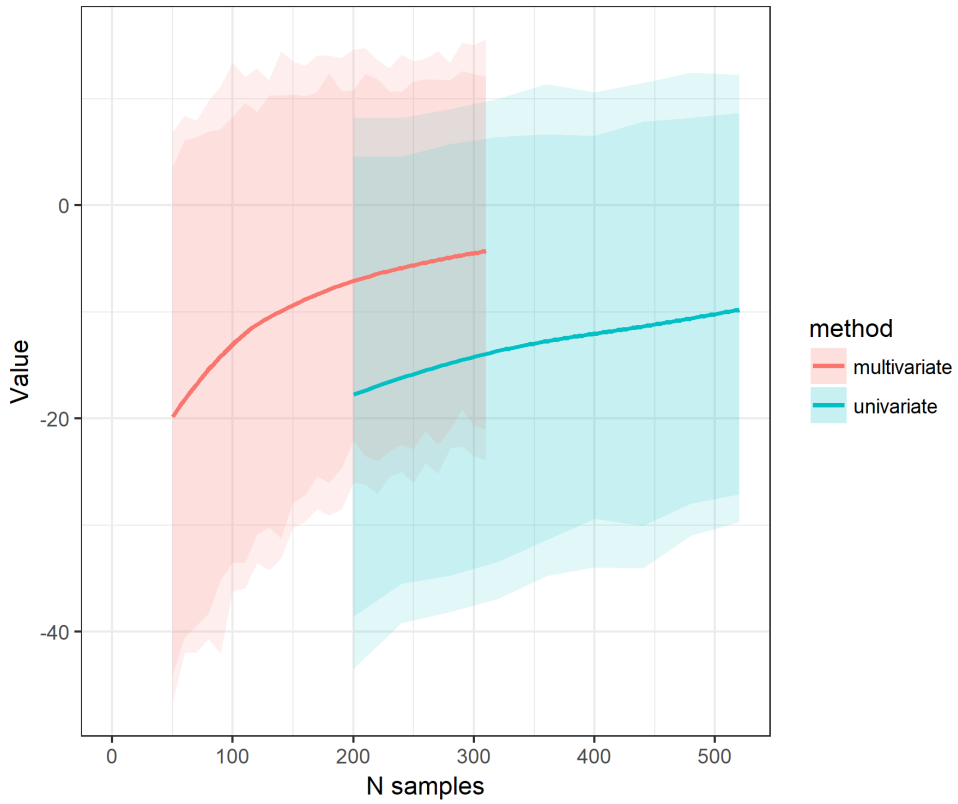


Figure 4.2: The results of the optimization process of the simulation exposed in 4.1. In this case we used an uncertainty set with size 10.

4.2 More Benchmarks and Comparisons

In this section we want to present some numerical comparison of the two methods proposed so far. The total number of possible variations of the methods in terms of characteristics of the problem, choice of the kernel, optimization methods, inference methods, strategies to choice the points to sample etc.., is gargantuan, so we will focus on some simple cases with the objective of illustrating the general behavior of the methods.

The first benchmark is obtained using a parametric version of the Rastrigin function, defined as follows

$$f_{Rastrigin}(x) = c + 10n + \sum_{i=1}^{N}[(a + bx_i)^2 - 10\cos(2\pi(a + bx_i))] \tag{4.1}$$

where n is the dimension of the domain and a, b, c are the parameters of the function.

Randomly sampling the parameters from a normal distribution we can easily build an arbitrary uncertainty set and so we can use the obtained problem to test our methods.

Gaussian noise is added at each sample. Since we always discussed the maximization problem, in the tests we multiplied the function for -1.

A second benchmark function that we used is the Styblinski-Tang function, defined as follows

$$f_{Styblinski-Tang}(x) = c + \sum_{i=1}^{n}[(a + bx_i)^4 - 16(a + bx_i)^2 + 5(a + bx_i)] \tag{4.2}$$

where n is the dimension of the domain and a, b, c are the parameters of the function, sampled as previously discussed.

The results in figure 4.5 show that, for the same number of observations, the multivariate method achieves a better mean results with lower variance, especially for very small samples. It has to be specified that the computational cost of the multivariate method is significantly greater, so it is opportune to use it only if the cost o a sample is sufficiently big to justify the investment.

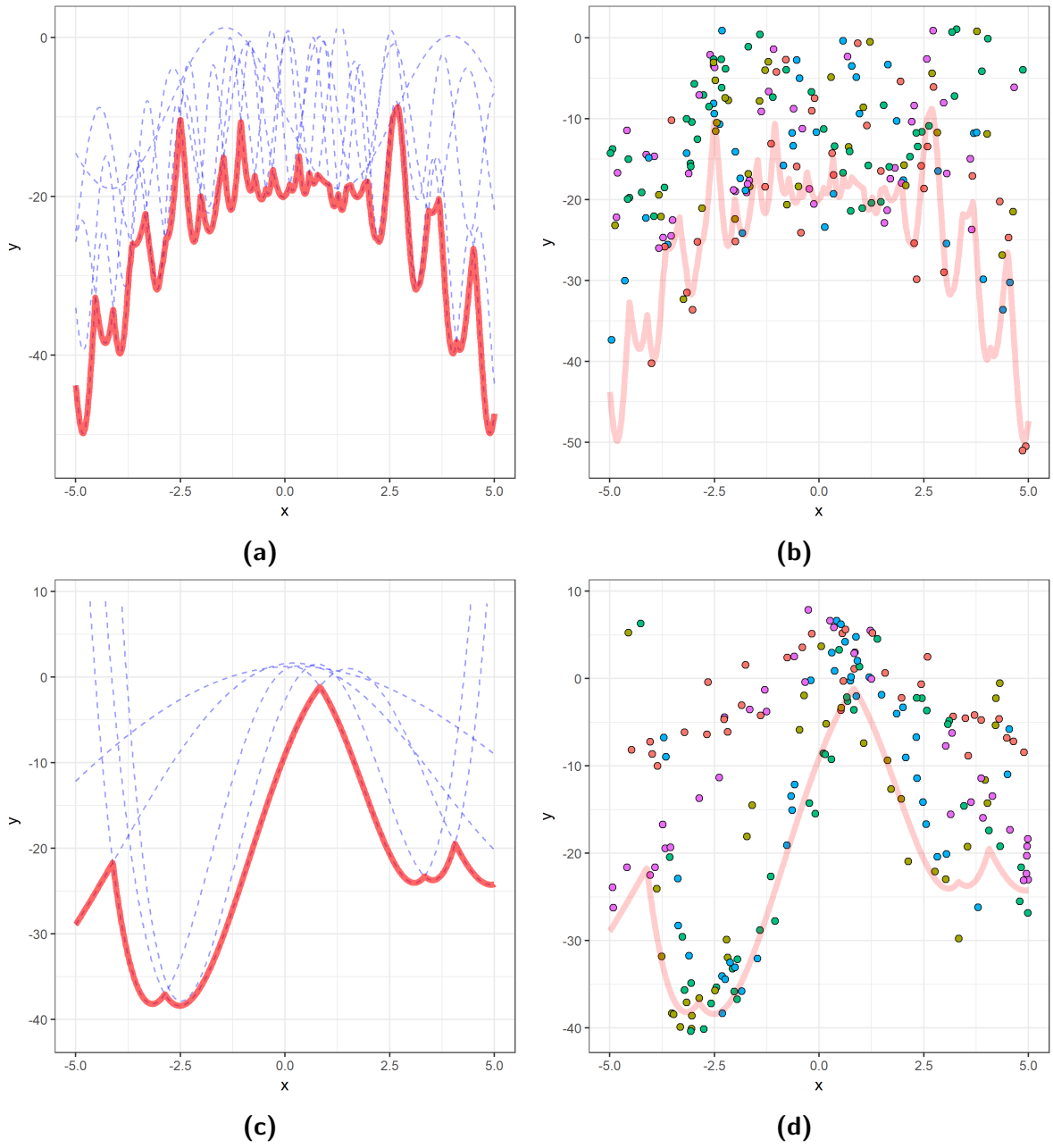
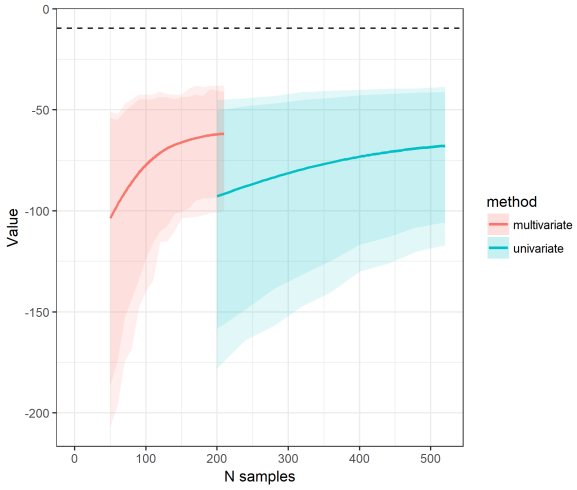
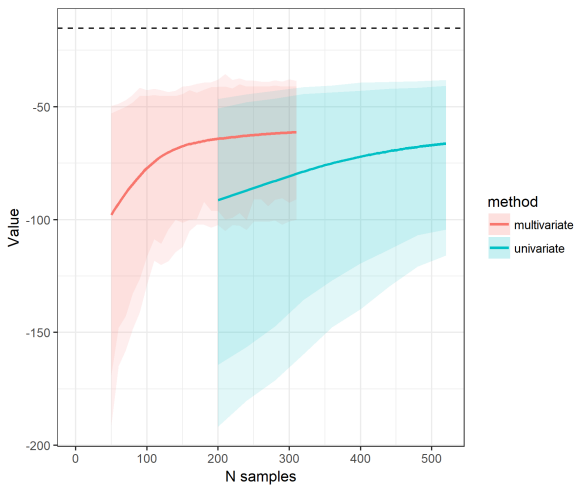


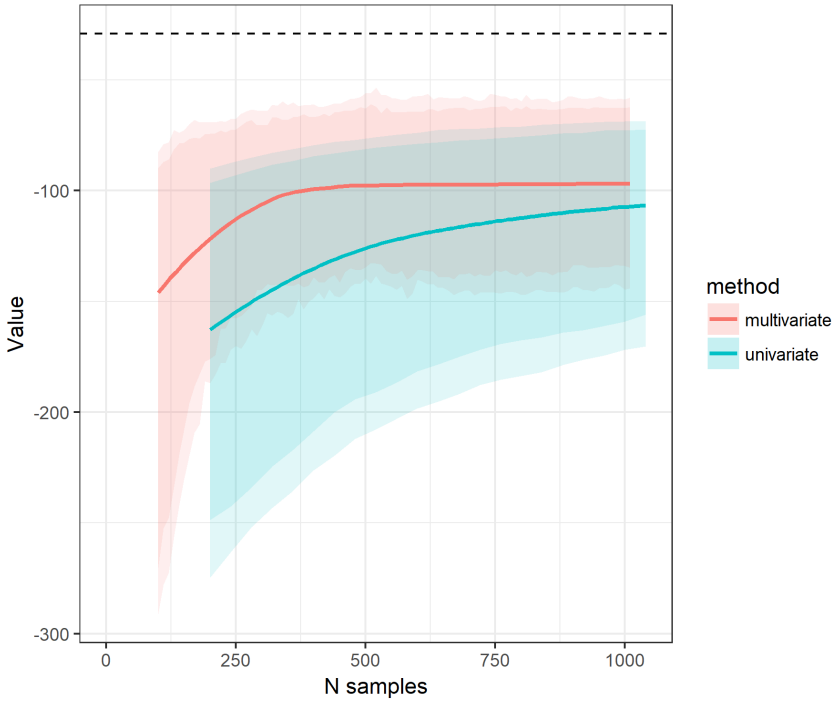
Figure 4.3: An example of a worst case optimization benchmark problems build with: 4.3a 1-D Rastrigin function and a discrete uncertainty set with 5 elements; 4.3c 1-D Styblinski-Tang function and a discrete uncertainty set with 5 elements. Figures 4.3b and 4.3d show samples for a total of 200 points with additive Gaussian noise with standard deviation 3; each color represents one of the functions.



(a) $dim(\mathcal{X}) = 2$, $|\mathcal{U}| = 5$, $\sigma = 3$

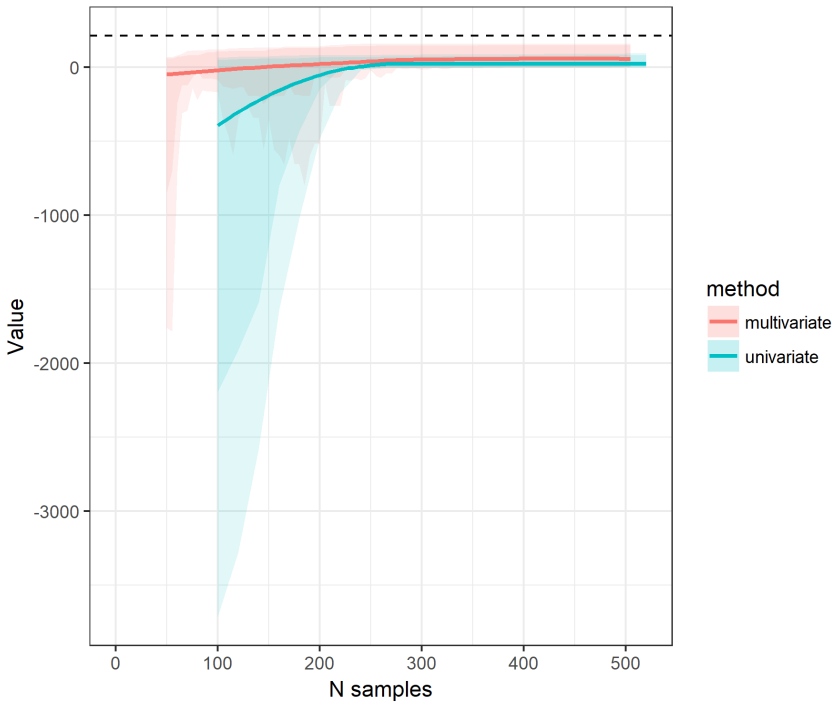


(b) $dim(\mathcal{X}) = 3$, $|\mathcal{U}| = 10$, $\sigma = 5$

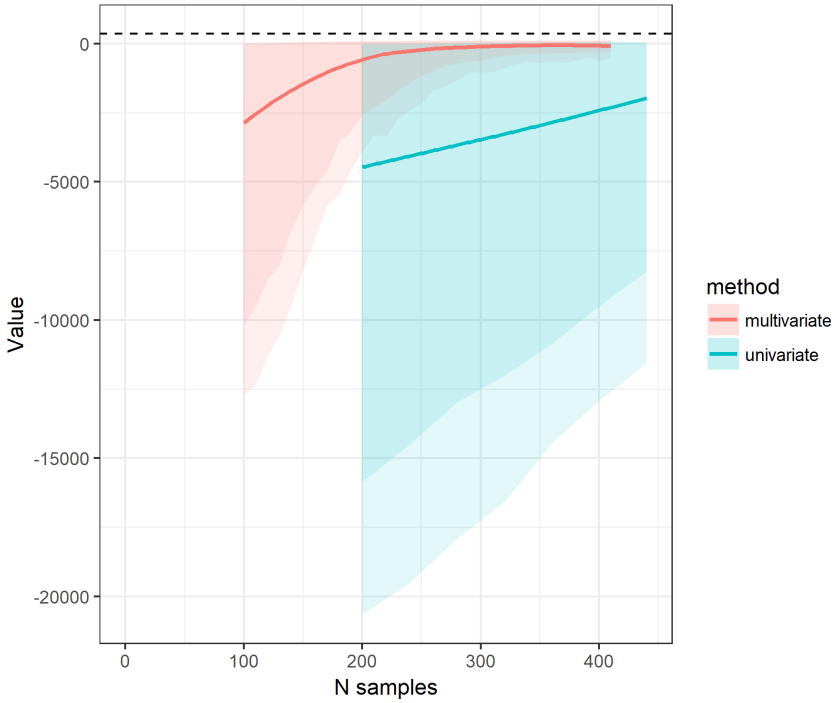


(c) $dim(\mathcal{X}) = 10$, $|\mathcal{U}| = 10$, $\sigma = 5$

Figure 4.4: The results for the Rastrigin function benchmark obtained using the standard univariate Gaussian process model and the multivariate Gaussian process using *expected improvement* and *minimum expected improvement* respectively. The lines represents the mean trend while the shaded area represents the quantiles of the distribution of the results (0.025 and 0.05). The horizontal dashed line represent the true global maximum.

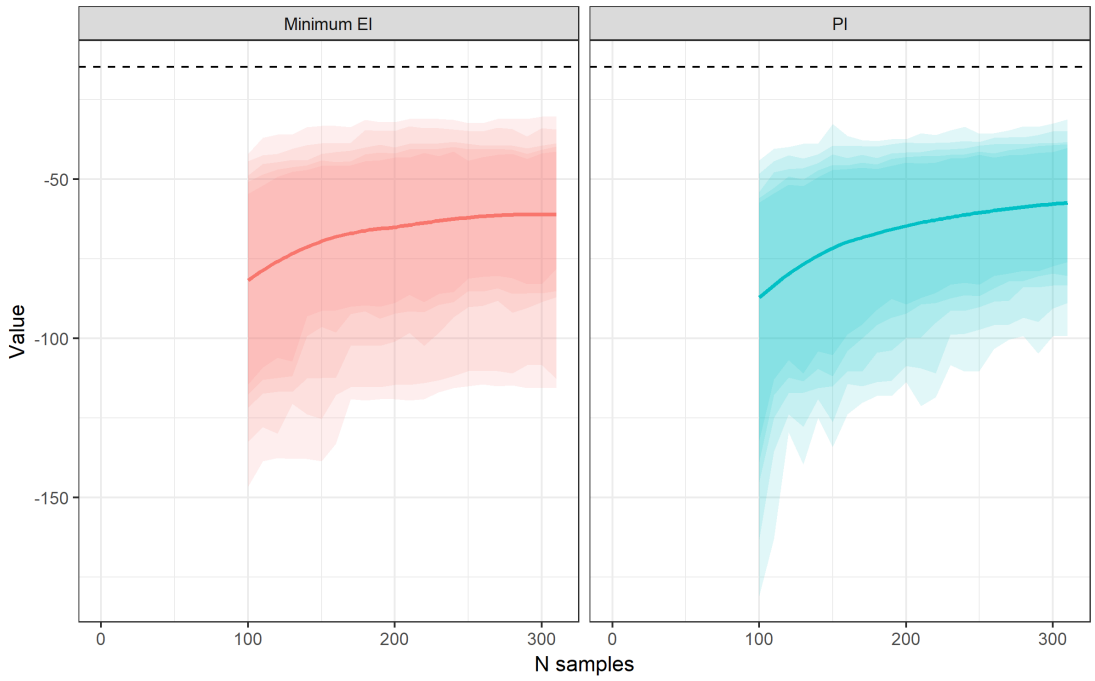


(a) $dim(\mathcal{X}) = 3$, $|\mathcal{U}| = 5$, $\sigma = 5$

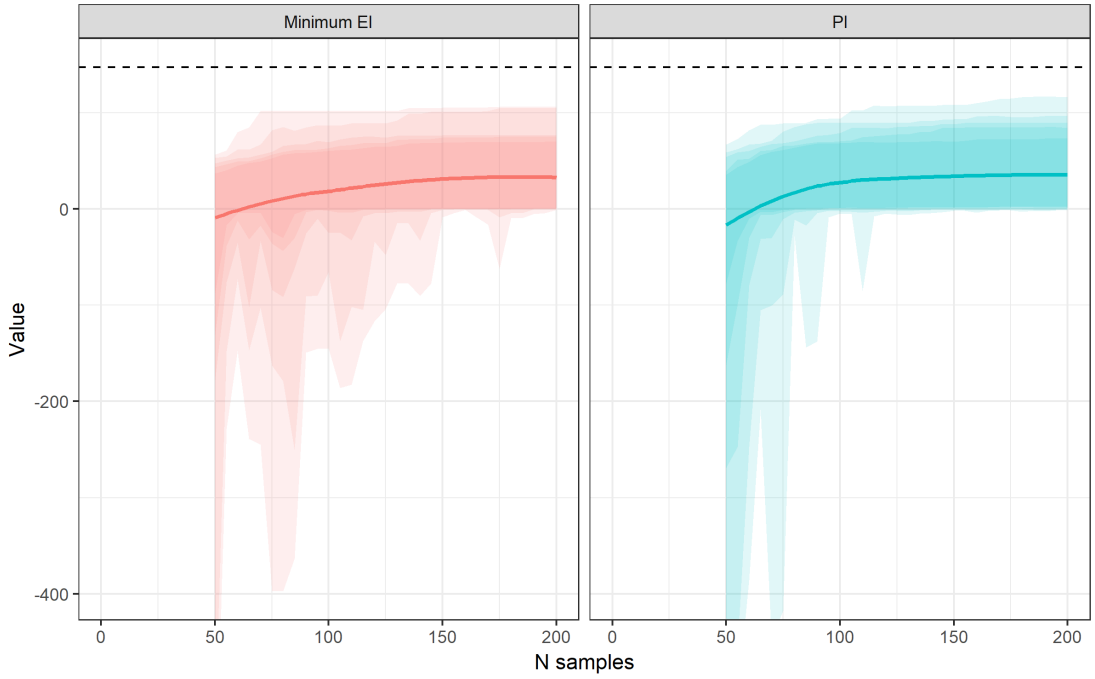


(b) $dim(\mathcal{X}) = 10$, $|\mathcal{U}| = 10$, $\sigma = 5$

Figure 4.5: The results for the Styblinski-Tang function benchmark obtained using the standard univariate Gaussian process model and the multivariate Gaussian process using *expected improvement* and *minimum expected improvement* respectively. The lines represents the mean trend while the shaded area represents the quantiles of the distribution of the results (0.025 and 0.05). The horizontal dashed line represent the true global maximum.



(a)



(b)

Figure 4.6: The results for the Rastrigin function benchmark (4.6a) and the Styblinski-Tang one (4.6b) with different augmentation functions for the multivariate model (minimum expected improvement and probability of improvement respectively).

Chapter 5

Conclusions

In the first chapter we presented a family of robust optimization problems, which aim is to avoid the risk of bad outcomes deriving from uncertainties in the objective function coming from parameters estimation, distributional uncertainties, constituent hypothesis or, more generally, from our lack of full understanding of the real process that we want to optimize. Usually, to represent our nescience, an *uncertainty set* is used to index an ensemble of possible objective functions. Many methods are possible, but we focused on a max-min approach.

This kind of problems can vary much on the details and issues case by case, but at least four different levels of difficulties are quite common: the global optimization of a (possibly) highly non-linear, non-convex function; the uncertainties of the objective function given by our lack of understanding of the mechanisms of the real problem; the uncertainties of the evaluation of the objective function, deriving from the stochastic nature of the numerical simulations or the empirical experiments used in the optimization process; the high cost in term of time and/or money of the sampling method.

Given the described difficulties, direct methods could fail to find good solutions for this kind of problems with reasonable amounts of resources, so we preferred methods that, instead of directly optimize the objective function, use a much cheaper surrogate model for handle both the high cost and the stochastic nature of the sampling process. In particular, we used Gaussian processes to model the objective function and to guide the choice of both the current solution and the next query point. This branch of optimization is usually called *Bayesian optimization*.

We proposed a novel class of variants of methods present in literature of Bayesian optimization capable of handling multivariate functions. The idea behind these methods is quite general and can be combined with almost all the standard techniques.

We showed that these methods often find better solutions given the same amount of observations and can therefore drastically increase the efficiency of Bayesian optimization

strategies in the case of stochastic max-min problems where the sampling function is non-trivial.

Some limitations of our method are the greater computational burden given by the necessity of fitting more models or a bigger one, that can be justified only if the sampling cost is sufficiently high, and, in some cases, an higher sensitivity to the initial sampling scheme and the choice the hyper-parameters of the model.

A more detailed analysis of the results given by different kernels, multivariate models and different augmentation functions is left for future works. Future development could also focus on an extension of the max-min scenario to more complex objective function structures, for example those defined by tree-like hierarchies.

Appendix A

Implementation notes

All the numerical experiments present it this work have been built in Python 3.6 with the extensive usage of various packages, in particular:

- *GPy: A Gaussian process framework in python*,
  version 1.9.2 (http://github.com/SheffieldML/GPy)
- *SciPy: Open source scientific tools for Python*,
  version 1.0.0 (http://www.scipy.org)

GPy is a nice package built appositely for defining and fitting an enormous variety of different Gaussian (and non-Gaussian) processes with many different techniques. Its flexibility permit to explore many different approaches to the same problem changing only a couple of lines of code.

SciPy offers a vast collection of standard local and global optimization methods, that have been used in various parts of the code.

The vast majority of the plot present in this work have been produced with the use of the R's package *ggplot2* (http://ggplot2.org).

Bibliography

[1] C. E. Rasmussen, C. K. I. Williams,
Gaussian Processes for Machine Learning,
the MIT Press, 2006

[2] H. Wackernagel,
Multivariate Geostatistics,
Springer, 1995

[3] Z. Chen, B. Wang, A. N. Gorban,
Multivariate Gaussian and Student−t Process Regression for Multi-output Prediction,
2017

[4] B.W. Rust,
Truncating the Singular Value Decomposition for Ill-Posed Problems,
1998

[5] B. Ankenman, B.L. Nelson, J. Staum,
Stochastic Kriging for Simulation Metamodeling,
2010

[6] D. R. Jones,
A Taxonomy of Global Optimization Methods Based on Response Surfaces,
2001

[7] D.R. Jones, M. Schonalu, and W. J.Welch,
Efficient Global Optimization of Expensive Black-Box Functions,
1998

[8] M. C. Fu,
Handbook of Simulation Optimization,
Springer, 2014

[9] J.P.C. Kleijnen, E. Mehdad,
Multivariate versus univariate Kriging metamodels for multi-response simulation models,
European Journal of Operational Research, 2014

[10] G. Dellino a, J.P.C. Kleijnen, C.Meloni
Robust optimization in simulation: Taguchi and Response Surface Methodology,
Int. J. Production Economics, 2010

[11] Romain Benassi, Julien Bect, Emmanuel Vazquez,
Robust Gaussian process-based global optimization using a fully Bayesian expected improvement criterion,
5th International Conference on Learning and Intelligent Optimization (LION 5), Jan 2011, Rome, Italy. Springer, 6683, pp.176-190, 2011, Lecture Notes in Computer Science

[12] Eric Brochu, Vlad M. Cora and Nando de Freitas,
A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning,
December 2010

[13] Eduardo C. Garrido-Merchán, Daniel Hernández-Lobato,
Dealing with Integer-valued Variables in Bayesian Optimization with Gaussian Processes,
June 2017

[14] Mauricio A. Alvarez, Lorenzo Rosasco, and Neil D. Lawrence,
Kernels for Vector-Valued Functions: a Review,
June 2011

[15] Joshua D. Svenson · Thomas J. Santner,
Multiobjective Optimization of Expensive Black-Box Functions via Expected Maximin Improvement

[16] Kevin Swersky, Jasper Snoek, Ryan P. Adams,
Multi-Task Bayesian Optimization,
December 2013

[17] Jasper Snoek, Hugo Larochelle, Ryan P. Adams,
Practical Bayesian Optimization of Machine Learning Algorithms,
December 2012

[18] J.A. Nelder, R. Mead,
A simplex method for function minimization

[19] K. Kersting, C.Plagemann, P. Pfaf, W. Burgard,
Most Likely Heteroscedastic Gaussian Process Regression,
2007

[20] Q. V. Le, A. J. Smola, S. Canu,
Heteroscedastic Gaussian Process Regression,
2012

[21] M. Lázaro-Gredilla, M. K. Titsias,
Variational Heteroscedastic Gaussian Process Regression,
2011

[22] C. Wang, R. M. Neal,
Gaussian Process Regression with Heteroscedastic or Non-Gaussian Residuals,
2012

[23] E. Contal, V. Perchet, N. Vayatis,
Gaussian Process Optimization with Mutual Information
2013

[24] J. González, M. Osborne, N. D. Lawrence,
GLASSES: Relieving The Myopia Of Bayesian Optimisation,
2015

[25] J. Hensman, N. Fusi, N. D. Lawrence,
Gaussian Processes for Big Data, 2013