Imperial College London
Department of Computing

High-dimensional Bayesian optimization with intrinsically low-dimensional response surfaces

Riccardo Moriconi

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing of Imperial College London, December 2020

Statement of originality: I, Riccardo Moriconi, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Bayesian optimization is a powerful technique for the optimization of expensive black-box functions. It is used in a wide range of applications such as in drug and material design and training of machine learning models, e.g. large deep networks. We propose to extend this approach to high-dimensional settings, that is where the number of parameters to be optimized exceeds 10–20. In this thesis, we scale Bayesian optimization by exploiting different types of projections and the intrinsic low-dimensionality assumption of the objective function. We reformulate the problem in a low-dimensional subspace and learn a response surface and maximize an acquisition function in this low-dimensional projection. Contributions include i) a probabilistic model for axis-aligned projections, such as the quantile-Gaussian process and ii) a probabilistic model for learning a feature space by means of manifold Gaussian processes. In the latter contribution, we propose to learn a low-dimensional feature space jointly with (a) the response surface and (b) a reconstruction mapping. Finally, we present empirical results against well-known baselines in high-dimensional Bayesian optimization and provide possible directions for future research in this field.

Acknowledgements

I would like to thank my primary supervisor, Prof. Marc Deisenroth, for his invaluable feedback, guidance and advice over the years of my PhD. I am also very grateful for his support, understanding and his constant attitude towards creating a pleasant environment inside the Statistical Machine Learning Group.

My gratitude goes as well to all my secondary supervisors. Especially I wish to thank Dr. Sesh Kumar, for his valuable support, his helpful advice and his constant presence during my PhD career. I wish to thank Prof. Ruth Misener for her useful help and suggestions.

I would also like to thank all my PhD colleagues and friends, especially Hugh and James, who have provided helpful insights in my research and proposed useful research discussions. I wish to thank Janith who has been a great companion during the 2019 NeurIPS conference.

I am thankful to the center of doctoral training "High Performance Embedded and Distributed Systems (HiPEDS)", which has provided a funding for my PhD. I am very thankful to all members of the HiPEDS 2016 cohort and staff for they have made the years of my PhD more lively.

I wish to thank my family for all the support they provided during my studies. Their experience and presence has been pivotal for my well-being. In particular I wish to thank my brother Stefano who has shared three years in London with me.

Finally, I wish to thank all my friends from Italy who have made each day more funny and happy during the challenging times of the PhD. In particular, I wish to thank Marco, Filippo, Gianluca, Leonardo, Francesco, Vittorio, Pasquale, Matteo and Maicol.

Dedication

A mio fratello Stefano, con affetto, per il suo inestimabile supporto.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation and Objectives

Carefully planning experiments is a crucial problem in research studies. In software optimization [HHLB11], for instance, selecting the most promising parameter configurations improves the performances of integer programming solvers. In drug and material design [GBWD$^+$18], identifying the right molecular structure allows for novel drug discovery. In all these experimental design problems, we are concerned with a set of parameters and an unknown or black box objective function. The parameters specify the degrees of freedom, that is the design choices that one is allowed to modify. The black box objective function, instead, denotes a measure of performance of the outcome of a set of parameters for which no analytic expression is available. Examples of parameter configurations are, in algorithm optimization [HHLB11], the configurable numbers and categorical decision variables implicit in a mixed integer programming solver. Here, the black box objective function could be defined as the mean runtime, which we wish to reduce, of a solver with different sets of instances. In drug design [GBWD$^+$18], for instance, the degrees of freedom coincide with the discrete molecular representations and the objective function quantifies drug-likeness and accessibility score.

The overall goal in our experimental design setting is to reach the global optimum of the objective function in as few trials of parameter configurations as possible. In fact, the process of obtaining an output from a utility function may be time consuming or expensive to evaluate. As an example, a software may take hours to find a specific solution. Testing and making new compounds is both costly and time consuming. [1]

Bayesian optimization (BO) is a powerful technique for the optimization of black box functions that are prohibitively costly to evaluate. BO is a model based optimization method that lies at the intersection of statistical inference and Bayesian decision theory. The idea with the model based approach is to build a surrogate model, by means of statistical inference approaches, of the costly objective function and use it as a proxy for the expensive black box optimization. This proxy model is also known as the response surface. Bayesian decision theory instead identifies which sets of parameters are more useful for the optimization purpose by trading off exploration and exploitation in the probabilistic surrogate model with an acquisition function. The acquisition function is a score that quantifies the notion of utility, that is how useful each parameter configuration is for the purpose of optimization. This acquisition function is a heuristic approach that trades off exploration and exploitation of the response surface and is maximized to select the new location to query the black box objective function at.

The Bayesian optimization approach was initially introduced in 1933 by William Thompson [Tho33]. In this first example, a probabilistic approach to a decision problem involving sequential clinical treatment experiments was proposed. The idea was to estimate probabilities of success of a set of treatments and then weight them based on their estimated chances of success. The novelty in this was to not discard the seemingly sub-optimal treatment based on a single trial but rather weight all of them according to their probability of success. Thompson argued that, with this novel approach, the fraction of patients receiving a sub-optimal treatment would tend to zero. An early work on Bayesian optimization was introduced by Kushner in 1964 [Kus64] which was based on a one-dimensional unconstrained optimization problem solved with the use of Wiener processes [RY13]. The idea of the proposed approach was to maximize the probability of the improvement with each new selection of the decision variable. The strategy for selecting the new points featured a term which was trading off exploration

[1]The objective function need not be a cost function to be minimized. The black box objective function may also represent a reward function and therefore require to be maximized instead of minimized.

and exploitation of the search space. The term Bayesian optimization was then introduced later in [MTZ78] by Jonas Močkus who also developed a multidimensional BO based on a linear combination of Wiener fields [MTZ78, Moč94] in the nonparametric setting. These early works, together with *kriging* [Kri51] developed by a master student named Krige in 1951, have been seminal for the development of the design and analysis of computer experiments (DACE) [SWMW89, JSW98, Jon01].

Bayesian optimization seeks the global extrema of objective functions for which an analytic expression is missing and the evaluation process is particularly cumbersome and expensive. It has proven successful for a wide range of experimental design problems. For instance, selecting gait properties of legged robots that lead to robust and fast walking performances [CSPD16]; adjusting controller configurations in robotic platforms and drones which results in reduced feedback-error of closed loop systems [BSK16]; clever tuning of hyper-parameters settings in multi-layer convolutional neural networks speeds up learning and lowers generalization error [SLA12].

Despite the great success of this approach in many applications, the employment of Bayesian optimization has been limited only to problems that feature a moderate number of parameters to tune, namely 10–20 parameters. In fact, as we increase the number of dimensions of the parameter space we incur in a series of challenges. The amount of data required for learning a response surface increases exponentially with the number of dimensions of the search space. This implies large amounts of input parameters and observations of the black box objective function which is in general expensive to evaluate. Learning and making predictions with the response surface model, for some choices of models that are standard in Bayesian optimization, become computationally challenging due to the amount of data involved in the Bayesian training and prediction derivation. The acquisition function is a score that tells us how useful each parameter configuration is for the purpose of optimization. This acquisition function needs to be maximized to select a new parameter configuration and requires an exponential number of evaluations in the number of dimensions to be maximized. In order to elevate Bayesian optimization to state-of-the-art in global optimization we therefore need to address the challenges that arise in high dimensional settings.

The purpose of this thesis is to introduce techniques and methods to tackle the curse of dimensionality that affects Bayesian optimization. In particular, we aim at attacking the high dimensional BO problem with novel strategies based on projections.

Among the benefits that this thesis could bring to the scientific community in Bayesian optimization, one important aspect to consider is that of a democratization of technology. The task of tuning hyper-parameter and parameter settings in academic research and industry is now performed by only few human experts that have prior knowledge about the system to be optimized [SSW+15]. An extension of optimization algorithms to high dimensions would benefit non expert users and allow a wider community to use highly parametric models. An example of these highly parametric models is that of machine learning models such as deep networks. A large number of parameters in optimization would also imply the applicability of Bayesian optimization to a wider class of problems. An automated approach to hyper-parameter tuning would also allow exploiting inter-dependencies and relationships between parameters that are overwhelmingly high to deal with by humans and cannot be captured or exploited by experts.

1.2 Contributions

This thesis collects contributions made in the direction of high-dimensional Bayesian optimization. In particular, we argue that the use of projections is particularly beneficial in the field of high-dimensional Bayesian optimization. Using projections allows reformulating the optimization problem on a low dimensional subspace characterized by a feasible dimensionality for optimization which is less or equal to 20 dimensions. Both the challenges that arise in high-dimensions, such as the exponential number of evaluations of the objective function and exponential number of maximization steps of the acquisition function, become feasible in a projected space. In fact, if we project to a low dimensional space characterized by dimensionality less or equal to 20, the optimization becomes feasible again.

1.2.1 Axis aligned projections

One contribution of this thesis is to present a Bayesian optimization method that divides the original search space into disjoint subsets of dimensions and then performs optimization on each subset of dimensions independently. This optimization strategy is based on axis aligned projections, that is selecting a subset of dimensions from the original search space. In this thesis, we address challenges that arise from the introduction of axis aligned projections in the optimization framework. The result of axis aligned projections is the presence of ambiguities in the data. This means that more than one observation may be associated with the same input variable. For instance, in a two dimensional example, two distinct input variables that share the same first dimension will result in the same point if we suppress the second dimension which makes them different. This suppression of the second dimensions makes the two distinct points in a two-dimensional space become equal in the first single dimension. The corresponding observation variables will remain two distinct values therefore causing an ambiguity or inconsistency that is multiple observation values for the same input point.

The problem of performing optimization on axis aligned projections of the search space has been addressed in [UBC$^+$16] where different subsets of dimensions are assigned different datasets resulting in a data-costly optimization. A similar optimization strategy was also presented in [KSP15] where the objective function is characterized by a the additivity property which implies a structural assumption about the composition of the subsets of dimensions. In our work, we adopt a probabilistic model that allows exploiting all the data available in each axis aligned projection. In addition, our proposed method does not require the introduction of structural assumptions about the objective function. In particular, we consider a probabilistic model based on Gaussian processes that generalizes only on a specific quantile of the data available. By selecting a small quantile in a minimization setting, a quantile Gaussian process automatically generalizes on the lowest observations therefore overcoming the problem of inconsistencies. The response surface obtained with this model results to be a good indicator of the location of the optimum in a minimization framework. We present in this thesis empirical evidence of the competitive performances of the proposed method across a set of well established baselines in high-dimensional Bayesian optimization.

1.2.2 Projections onto feature space

In this thesis we present the contribution of a Bayesian optimization algorithm in a low dimensional feature space. This method learns a low dimensional manifold of the data and then performs Bayesian optimization in this learned feature space. The idea is to model the composition of two functions, namely a feature mapping that takes care of the dimensionality reduction and a nonparametric mapping that learns the objective function in a low dimensional manifold of the original search space. The feature mapping models the relationship between the high-dimensional inputs and the low dimensional features, the response surface then models the link between the data in this low dimensional space and the data in observation space. The observation space is the space of the objective function evaluations. This allows for learning a response surface under the assumption that the objective function has an intrisically lower dimensionality, that is the high-dimensional objective function effectively depends on a small set of parameters. In the literature, this assumption is common and has been used in [WZH+13], where a linear mapping is assumed to model the relationship between the high dimensional search space and the low dimensional features. In the case of an intrinsically low dimensional objective function, a nonlinear feature mapping could achieve a higher compression rates between the high dimensional search space and the features.

The feature mapping therefore allows for learning the response surface and maximizing the acquisition function in a feature space, which is characterized by a feasible dimensionality, which means a dimensionality less or equal to 20. One additional step that is required for the optimization of the high dimensional objective function is the reconstruction of the input in the original data space. In this thesis, we present a probabilistic model that takes into account the reconstruction part by means of multiple output Gaussian processes. The idea with this approach is to introduce a *decoder* that takes features as input and transforms them back to the high dimensional space. This is an essential requirement for the optimization of the high dimensional objective function and defines an autoencoder like structure together with the feature mapping (*encoder*). The resulting probabilistic model is based on Gaussian processes both for the response surface modeling and the reconstruction of the decoder mapping. The joint training of this model allows learning features that are optimal in a marginal likelihood sense for two distinct tasks: i) modeling the response surface for Bayesian optimization in feature space and ii) reconstructing the original inputs via multi-output mapping.

Finally, we take into account the un-identifiability issues involved with a reconstruction mapping based on multi-output Gaussian processes. We introduce a nonlinear constraint to reduce the ambiguity set of the maximization of the acquisition function in feature space. The ambiguity set is the set in feature space where the maximization of the acquisition function is performed. This nonlinear constraint uses the Lipschitz constant of Gaussian process predictions and ensures the maximization does not move the decision variable too far away from data in feature space. In this thesis, we provide an empirical evaluation of the proposed Bayesian optimization in feature space where we compare to well known baselines in the literature.

1.2.3 Publications

- Riccardo Moriconi, K.S. Sesh Kumar and Marc P. Deisenroth. **High-dimensional Bayesian optimization with projections using quantile Gaussian processes**. *Optimization Letters. Pages 51–64, Volume 14. Springer.* 2020.

- Riccardo Moriconi, Marc P. Deisenroth and K.S. Sesh Kumar. **High-dimensional Bayesian optimization using low-dimensional feature spaces**. *NeurIPS Workshop on Bayesian Deep Learning.* 2019.

- Riccardo Moriconi, Marc P. Deisenroth and K.S. Sesh Kumar. **High-dimensional Bayesian optimization using low-dimensional feature spaces**. *Machine Learning Journal.* 2020.

1.3 Outline of the thesis

Chapter 2 includes a background section where all required knowledge about Bayesian optimization and its components is presented. It presents concepts related to the response surface models, acquisition functions descriptions and Gaussian processes. Gaussian processes are popular probabilistic approaches for modeling the response surface in Bayesian optimization literature. We further introduce challenges that arise from high dimensional settings in more detail. In Chapter 3, we present our first approach to the problem of high dimensional Bayesian optimization with axis aligned projections. We present the notion of a quantile Gaussian process and the approximate inference methods that are involved in its implementation. We conclude the chapter with empirical results, which compare the proposed method to well known baselines in the literature of high dimensional Bayesian optimization. In Chapter 4, we present the second approach to scaling Bayesian optimization to high dimensions based on feature spaces. We first present the probabilistic model and its components in detail. We describe the setting adopted for learning the feature space and reconstructing the features into high dimensional data. We present the results achieved for computational efficiency of multi output Gaussian processes. Finally we describe the acquisition maximization with the nonlinear constraint based on Lipschitz continuity of Gaussian process predictions. We conclude the chapter with an experimental evaluation of our baseline in comparison with standard choices of competing approaches. Chapter 5 describes the conclusions and the future work of this thesis.

Chapter 2

Bayesian optimization

In this chapter, we introduce the background required for the remaining of the thesis. In particular, we describe the Bayesian optimization approach in all its main components. We define the response surface as the surrogate model for learning the black box objective function. We then introduce the acquisition functions employed in this thesis and describe them in detail. We also present the main probabilistic model adopted in this thesis which is the Gaussian process model. Finally, we describe the main challenges involved in all these components when we increase the dimensionality of the search space.

Bayesian optimization is a field of study concerned with seeking the extrema of unknown functions that are expensive to evaluate. It comprises families of optimization problems where gradients or higher order derivatives are generally not accessible, observations of function values are corrupted by noise, and linearity and convexity properties are not granted. In particular, we consider minimization problems of the form

$$\text{find} \quad \mathbf{x}^* \in \operatorname*{arg\,min}_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}) \tag{2.1}$$

where $f\colon \mathcal{X}_f \subset \mathbb{R}^D \to \mathbb{R}$ is the *objective function.* The objective function is a scalar function taking values in a D-dimensional domain $\mathcal{X}_f$, which we can assume without loss of generality to coincide with the *ambiguity set* $\mathcal{X}$ of the optimization problem (2.1), that is $\mathcal{X}_f \equiv \mathcal{X}$. The ambiguity set $\mathcal{X}$ is also referred to as the *search space* of the optimization problem. Although the domain and ambiguity set may involve categorical and conditional variables, we restrict ourselves to the case where these sets involve continuous variables. *Inputs* $\mathbf{x}$ are associated with design choices or parameter configurations for experiments and are typically required to be bounded in all dimensions. In particular, the search space is assumed to be a box constrained set in the D-dimensional space such as the hypercube $\mathcal{X} = [0,1]^D$. *Observations* y of function values may also be corrupted by noise which is usually assumed to be identically distributed and independent (i.i.d.) Gaussian noise, with zero mean and σ_n standard deviation[1]

$$y = f(\mathbf{x}) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \sigma_n^2). \tag{2.2}$$

We consider objective functions f that are prohibitively costly to evaluate and for which no analytic expression is available. We assume we can evaluate the function at arbitrary inputs in the search space $\mathcal{X}$. We restrict ourselves to the typical setting, where neither gradients nor convexity properties of f are available. We further assume we are allowed a small budget of evaluation queries to express our best guess of the optimum's location $\mathbf{x}^*$ in at most T_{end} iterations. The main steps and components of the optimization routine are also described in Algorithm 1. In particular, we start with an initial set of starting inputs $\mathbf{X}_0 := \{\mathbf{x}_i\}_{i=1}^{N_0}$ and observations $\mathbf{y}_0 := \{y_i\}_{i=1}^{N_0}$. Then, at each iteration t, we select a new input point $\mathbf{x}_{t+1}$ at which to measure y_{t+1} i.e. a noisy evaluation of the objective function. At the last iteration T_{end} a Bayesian optimization algorithm returns the best guess of the optimum's location $\mathbf{x}_T^* = \arg\min f(\mathbf{x})$. The general framework models the unknown objective f with a surrogate model referred to as *response surface* and then feeds it into an acquisition function $\alpha(\mathbf{x})$, which expresses utility of evaluations with respect to the current estimate. Maximizing the acquisition function returns input locations $\mathbf{x}_{t+1}$ where to evaluate the true objective, i.e.

$$\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha(\mathbf{x}). \tag{2.3}$$

The true objective is then evaluated at the maximum utility input, $\mathbf{x}_{t+1}$, and the dataset $\mathcal{D}$ is

[1] Here the subscript n denotes the measurement noise

updated with the new sample.

Algorithm 1 Key steps of Bayesian optimization.

```
1: Inputs: $\mathbf{X}_0 = \{\mathbf{x}_1, ..., \mathbf{x}_{N_0}\} \in \mathbb{R}^{N_0 \times D}$, $\mathbf{y}_0 = \{y_1, ..., y_{N_0}\} \in \mathbb{R}^{N_0}$
2: for $t = 0, 1, 2, ..., T_{\text{end}} - 1$ do
3:     Response surface learning
4:         $p(f|\mathbf{X}_t, \mathbf{y}_t)$
           $\{$Training of the probabilistic model for data fit.$\}$
5:     Optimal input selection $\mathbf{x}_{t+1}$
6:         $\mathbf{x}_{t+1} = \underset{\mathbf{x} \in \mathcal{X}}{\text{argmax}}\ \alpha(\mathbf{x})$
           $\{$Acquisition function maximization$\}$
7:     Evaluation
8:         $y_{t+1} = f(\mathbf{x}_{t+1}) + \varepsilon$
           $\{$Evaluation of the high-dimensional objective function with measurement noise$\}$
9:     $\mathbf{X}_t \cup \{\mathbf{x}_{t+1}\}, \quad \mathbf{y}_t \cup \{y_{t+1}\}$
10: end for
11: Return $\mathbf{x}^* = \arg\min \{y_0, ..., y_{T_{end}-1}\}$
    $\{$Computed minimizer of the objective function $f\}$
```

2.1 Response surface

The response surface is the first component of the Bayesian optimization algorithm. It comprises a surrogate model that is used for learning the expensive black box function. This surrogate model is essentially used as a proxy for the expensive black box optimization process. In fact, instead of optimizing directly the costly black box objective function one can use the inputs and observations to derive an approximation of the objective function and use the response surface to guide the search for the optimum.

A response surface provides a probabilistic description of the unknown objective function, that is, given an input point $\mathbf{x}$, a surrogate model should return the expected value of the true objective function and related uncertainty in the prediction. The Bayesian approach allows for such probabilistic predictions based on data acquired during the optimization process. It mathematically formalizes our prior beliefs about the true objective and then updates it based on inputs and observations. In this framework, posterior predictions $p(f|\mathbf{X}, \mathbf{y})$, after observing data inputs $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ and observations $\mathbf{y} = \{y_1, ..., y_N\}$, where N is the number of data points, are obtained updating a prior $p(f)$ by the likelihood $p(\mathbf{y}|\mathbf{X}, f)$.

$$p(f|\mathbf{X}, \mathbf{y}) \propto p(\mathbf{y}|\mathbf{X}, f)p(f). \tag{2.4}$$

The prior $p(f)$ embodies initial assumptions about the function, and these are of key importance for reducing the sampling complexity of the overall Bayesian optimization algorithm. With Lipschitz continuity alone, for instance, the number of evaluations required to guarantee convergence of the optimization grows exponentially in the number of dimensions D [Moč94]. Covering the search space such that a final guess $\hat{f}$ is guaranteed to be no further than ϵ from the true optimum f^*, i.e. $|\hat{f} - f^*| < \epsilon$ needs at least $[L/(2\epsilon)]^D$ data points, where L is the Lipschitz constant [Bet91]. Prior probabilistic assumptions, instead, embody regularity properties, such as smoothness, and can be integrated with evidence from data. This allows for discarding pathological examples of objectives that would require extremely expensive exhaustive-search strategies, with confidence.

The second component, the likelihood, $p(\mathbf{y}|\mathbf{X}, f)$, updates the prior according to collected data points. Each function satisfying our initial assumptions weights according to how well it fits with the observations. The likelihood updates our state of knowledge about the true objective function after observations as the posterior mass concentrates on plausible functions that conform to data.

There are many examples of response surfaces in the literature, both parametric and nonparametric. In the parametric setting, it is worth mentioning the use of the Beta-Bernoulli bandit model as a probabilistic surrogate. This model has been applied successfully in recommendation and advertisement systems [LCLS10, CL11] and for modeling drug effectiveness [SSW$^+$15].

In the literature, there are also present response surface examples that do no need to specify a vector of parameters. Here we introduce nonparametric models for the response surface, namely Gaussian processes and random forests. Nonparametric models are characterized by an infinite dimensional parameter set that can be thought as a function. The parameters in this set directly correspond to data points. As the amount of data grows, the amount of parameters grows and therefore the learning capacity of the model grows. This renders nonparametric models more flexible than parametric ones. In contrast, parametric models are characterized by a fixed amount of parameters which capture all the information contained in the observed data. In fact, given the parameter vector, parametric model predictions are independent of the observed data. As the amount of data grows, the learning capacity of the parametric model remains the same. This renders parametric models less flexible than nonparametric ones. Gaussian processes are nonparametric probabilistic models that provide a distribution over the unknown function. Gaussian processes have a central role in this thesis as they have been employed as the probabilistic models in all our contributions and will be detailed in the following section. In our Bayesian setting however, the Gaussian process model defines a prior $p(f)$ in the space of functions. This prior models the function values $f(\mathbf{X}) = \{f(\mathbf{x}_1), ..., f(\mathbf{x}_N)\}$ as Gaussian random variables that have a joint distribution with mean specified through a mean function and a covariance matrix specified through a covariance function or kernel. In the case of Gaussian likelihood, that is $y = f(\mathbf{x}) + \varepsilon$ where the noise is Gaussian $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$ as specified in equation (2.2), the posterior $p(\mathbf{f}|\mathbf{X}, \mathbf{y})$ is also Gaussian and can be derived using linear algebra equations from the rules of Gaussian conditioning.

The other nonparametric model described in this thesis is the random forest [Bre01]. Random forests have been proposed as an alternative to Gaussian processes as a regression model for the response surface in Bayesian optimization. The main advantage in their employment is the scalability in the number of data points and their good performances in the presence of categorical data. In particular, this regression model has been employed successfully for the Bayesian optimization of the parameters of integer programming solvers [HHLB11]. Random forests are collections of regression trees which are like decision trees: flowchart-like structures in which each of the internal nodes generally represents a test or an attribute. Regression trees are decision trees where the variables involved are taken from the real numbers. Each leaf is assigned a set of data points and the average of all the regression trees produces an estimate of the unknown black box function. The estimate of the variance can be produced by computing the empirical variance of the predictions of the regression trees [HHLB11]. The result of this model is a jagged response surface. This characteristic can be useful when modeling black box objectives that are non smooth. Moreover, this construction allows the random forest to make fast predictions as computing predictions from regression trees is also fast and has been applied successfully with categorical variables [BBM04, BHBG07]. One of the downsides of this approach for modeling the response surface is that the extrapolation where data points are not available is poor. An additional drawback for using the random forest is that the uncertainty is underestimated because the empirical variance of identical regression trees results overconfident. Another downside is that performing optimization on this jagged response surface may be intrinsically hard and do not allow optimization with gradient based methods.

2.2 Gaussian processes

A central model for the response surface in this thesis is the Gaussian process (GP) [RW06] model. Gaussian processes represent a consistent and popular choice of prior distribution in the Bayesian optimization framework. They specify probability distributions over functions characterized by intuitive and tunable regularity properties. They have been extensively employed in machine learning [RW06] as an empirical inference method for, among other things, nonparametric nonlinear regression.

Definition 2.1 *[RW06] A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.*

A Gaussian process is fully specified by a *mean function*, $m(\cdot)$, and a positive definite *covariance function* (or *kernel*), $k(\cdot, \cdot)$. A Gaussian process specifies a probability distribution over a function space identified by the kernel, called *Reproducing Kernel Hilbert Space* (RKHS), which is characterized by properties, such as smoothness or periodicity, that depend on the functional form of the covariance function and the *hyperparameters* that parametrize the kernel. Here, we provide a definition of the RKHS

Definition 2.2 *[RW06] Let $\mathcal{H}$ be a Hilbert space of real functions f defined on an index set $\mathcal{X}$. Then $\mathcal{H}$ is called a reproducing kernel Hilbert space endowed with an inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ (and norm $\|f\|_{\mathcal{H}} = \sqrt{\langle f, f\rangle_{\mathcal{H}}}$) if there exists a function $k\colon \mathcal{X}\times\mathcal{X}\to\mathbb{R}$ with the following properties:*

- *for every $\mathbf{x}$, $k(\mathbf{x},\mathbf{x}')$ as a function of $\mathbf{x}'$ belongs to $\mathcal{H}$, and*
- *k has the reproducing property $\langle f(\cdot), k(\cdot,\mathbf{x})\rangle_{\mathcal{H}} = f(\mathbf{x})$.*

A key intuition behind the reproducing kernel Hilbert space is that the squared norm $\|f\|^2_{\mathcal{H}}$ in this space can be considered as a generalization to functions of the quadratic form $\mathbf{f}^T\mathbf{K}^{-1}\mathbf{f}$, where $\mathbf{f}$ is a finite vector containing function values and $\mathbf{K}$ is a matrix of covariances between f_i, f_j (elements of the vector $\mathbf{f}$). The matrix of covariances $\mathbf{K}$ is obtained from the evaluation of the kernel k at the input points $\mathbf{x}_i, \mathbf{x}_j$ of each function value in $\mathbf{f} = [f(\mathbf{x}_i)]_{i=1}^N$. The positive definiteness of the covariance function k ensures that the quadratic form $\mathbf{f}^T\mathbf{K}^{-1}\mathbf{f}$ is positive for all vectors $\mathbf{f}$ different from the null vector $\mathbf{0}$ (vector of all zeros), i.e.

$$\mathbf{f}^T\mathbf{K}^{-1}\mathbf{f} > 0 \quad \forall \mathbf{f} \neq \mathbf{0}. \tag{2.5}$$

The condition in equation (2.5) ensures that the norm $\|f\|^2_{\mathcal{H}}$ is also positive definite. In fact, we can recover a finite version of $\|f\|^2_{\mathcal{H}}$ if we express $\mathbf{K}$ and $\mathbf{f}$, in the quadratic form, in terms of the eigenvectors of $\mathbf{K}$. Therefore, the positive definiteness of the kernel ensures the positive definiteness of the norm $\|\cdot\|_{\mathcal{H}}$ defined in the reproducing kernel Hilbert space. For a more in depth treatment please refer to [RW06].

The resulting general assumption in this thesis is that the unknown objective function f is distributed according to a Gaussian process

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x},\mathbf{x}')\right). \tag{2.6}$$

This assumption allows for computing probabilistic predictions of objective function values for arbitrary inputs simply via linear algebra expressions. In particular, it implies that the function values $f(\mathbf{x}_1), ..., f(\mathbf{x}_N)$ taken at any arbitrary set of input locations $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ are distributed according to a Gaussian distribution with mean $[m(\mathbf{x}_1), ..., m(\mathbf{x}_N)]^T$ and covariance $[k(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^N$. In the case of Gaussian likelihood, also the noisy observations $\mathbf{y} = \{y_1, ..., y_N\}$ of the function values at the inputs are Gaussian distributed with mean $[f(\mathbf{x}_1), ..., f(\mathbf{x}_N)]^T$ and variance $\mathbf{I}_N\sigma_n^2$ (the measurement noise variance of the observations) as specified in equation (2.2). Here the term $\mathbf{I}_N$ denotes an $N \times N$ identity matrix. The probabilistic prediction of objective value at test point $\mathbf{x}_\star$ can then be obtained by conditioning the joint probability of $p(f(\mathbf{x}_\star), y_1, ..., y_N)$. In fact, the Gaussian likelihood implies that the joint vector $[f(\mathbf{x}_\star), y_1, ..., y_N]^T$ has a joint Gaussian distribution. The GP prediction at test point $\mathbf{x}_\star$ is again Gaussian distributed, and its mean and variance can be analytically computed with exact expressions from conditioning rules

$$p(f(\mathbf{x}_\star)|\mathbf{X}, \mathbf{y}) \sim \mathcal{N}(\mu(\mathbf{x}_\star), \sigma^2(\mathbf{x}_\star)) \tag{2.7}$$

$$\mu(\mathbf{x}_\star) = m(\mathbf{x}_\star) + k(\mathbf{x}_\star, \mathbf{X})[k(\mathbf{X}, \mathbf{X}) + \sigma_n^2\mathbf{I}]^{-1}(\mathbf{y} - m(\mathbf{X})) \tag{2.8}$$

$$\sigma^2(\mathbf{x}_\star) = k(\mathbf{x}_\star, \mathbf{x}_\star) - k(\mathbf{x}_\star, \mathbf{X})[k(\mathbf{X}, \mathbf{X}) + \sigma_n^2\mathbf{I}]^{-1}k(\mathbf{X}, \mathbf{x}_\star). \tag{2.9}$$

Here the $1 \times N$ vector $k(\mathbf{x}_\star, \mathbf{X}) := [k(\mathbf{x}_\star, \mathbf{x}_i)]_{i=1}^N$ contains the evaluation of the covariance function between test point $\mathbf{x}_\star$ and the training set $\mathbf{X}$. The square $N \times N$ matrix $k(\mathbf{X}, \mathbf{X}) := [k(\mathbf{x}_i, \mathbf{x}_j)]$ for $i, j = 1, ..., N$ is obtained by computing the kernel for each possible pair of training inputs. The column vector $m(\mathbf{X}) := [m(\mathbf{x}_i)]_{i=1}^N$ is the collection of evaluations of the mean function at the training input set while the term $m(\mathbf{x}_\star)$ is also the evaluation of the mean function but for the single test set. The remaining term $k(\mathbf{X}, \mathbf{x}_\star) := k(\mathbf{x}_\star, \mathbf{X})^T$ is just a transposition of the row vector defined at the beginning and the $k(\mathbf{x}_\star, \mathbf{x}_\star)$ is the kernel evaluated at all possible pairs of the test set (single point). These equations allow for deriving probabilistic statements about function values laying within the confidence intervals around the mean estimate. In particular, equations (2.8)–(2.9) describe the posterior mean and posterior variance of the objective function values at the location $\mathbf{x}_\star$. These posterior predictions are used by the acquisition functions to evaluate the utility of each input location and will be detailed in the next section.

The kernel models the covariance of an arbitrary pair of function values $f(\mathbf{x}), f(\mathbf{x}')$ as a function of the input locations $\mathbf{x}$ and $\mathbf{x}'$ at which the function is evaluated. Stationary covariance functions $k(\mathbf{x}, \mathbf{x}')$ depend on some notion of distance between the inputs. Intuitively, it follows that points that are close in input space $\mathcal{X}$ have function values that have high covariance and therefore are highly correlated. This also defines the property of smoothness in the function modeled by the kernel that is for small variations in input space correspond highly correlated function values. Here we report common choices of kernel functions in the literature of Bayesian optimization, namely the *squared exponential* and *Matérn* kernels [RW06, Fra18]

$$k_{\mathrm{SE}}(r) = \sigma_f^2 \exp\left(-\frac{r^2}{2l^2}\right) \tag{2.10}$$

$$k_{\mathrm{Matérn}}(r) = \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}r}{l}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}r}{l}\right). \tag{2.11}$$

Here, ν, l and σ_f^2 are positive parameters, where the σ_f^2 denotes the signal variance of the process. The Γ and K_ν are the Gamma and the modified Bessel functions, respectively. The kernels in equations (2.10)–(2.11) are stationary kernels, that is they only depend on a distance measure r between two points $\mathbf{x}$ and $\mathbf{x}'$. Distances are rescaled by length-scale hyperparameters which express wiggly behavior of the function. The length-scales informally denote the distance in input space that is required for two function values to become uncorrelated. Intuitively, functions that rapidly vary are characterized by short length-scales. It is also possible to define a family of kernels with different length-scales hyperparameters for each dimension. This allows for representing different function variations with respect to input space directions. A kernel that is characterized by a different length-scale parameter per input dimension is referred to as automatic relevance determination (ARD) kernel [RW06]. In this thesis, we will focus on the Matérn kernel with automatic relevance determination. The parameter ν is directly related to the smoothness of the functions modeled by this kernel. A function f is k-times differentiable if and only if $k < \nu$ [RW06]. This becomes relatively simple for values such as $\nu = p + 1/2$ where p is a positive integer number. In our work we consider the value $\nu = 5/2$ which is twice differentiable. The resulting *Matérn*$_{5/2}$ kernel becomes

$$k_{\nu=5/2}(r_l) = \sigma_f^2 \left(1 + \sqrt{5} r_l + \frac{5 r_l^2}{3}\right) \exp\left(-\sqrt{5} r_l\right). \tag{2.12}$$

Here the term r_l is obtained rescaling the distance between arbitrary inputs $\mathbf{x}$ and $\mathbf{x}'$ by the length-scales per each dimension, that is $r_l^2 := (\mathbf{x} - \mathbf{x}')^T \boldsymbol{\Lambda} (\mathbf{x} - \mathbf{x}')$, where the $\boldsymbol{\Lambda}$ matrix is a diagonal with the reciprocal of the squared length-scales on the main diagonal $1/l_i^2$, for $i = 1, ..., D$. Other common choices of the parameter ν are $\nu = 3/2$ and $\nu = 1/2$. Note that for $\nu \to \infty$ we recover the squared exponential kernel described in equation (2.10).

An essential step for the Gaussian process regression is the efficient training via marginal likelihood maximization. The training via marginal likelihood maximization does not simply favour the models that fit the training data the best but also that have low complexity and therefore can explain simply and well the data observed. This effect is called Occam's razor because encourages simplicity in explanations [RW06]. The Gaussian process, although being a nonparametric model, is characterized by a set of hyper-parameters that define the properties of the function space identified by the kernel. In particular, for the Matérn$_{5/2}$ kernel we selected there are two sets of parameters that affect predictions: the signal variance (or kernel variance) σ_f^2 and the characteristic length-scales $l_1, ..., l_D$. We will collect these hyper-parameters into the variable $\boldsymbol{\theta}$, together with the noise variance σ_n^2, which can be learned from data, that is $\boldsymbol{\theta} := \{\sigma_f^2, l_1, ..., l_D, \sigma_n^2\}$. In the rest of the thesis, we will assume without loss of generality that the mean function is the zero function, i.e. $m(\mathbf{x}) = 0$ for all $\mathbf{x} \in \mathcal{X}$ and therefore we will not have any mean function parameters to be learned from data. The marginal likelihood $p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$ is obtained marginalizing over the latent function space, that is averaging the likelihood $p(\mathbf{y}|\mathbf{X}, f, \boldsymbol{\theta})$ with respect to the GP prior $p(f|\boldsymbol{\theta})$ (here we have made explicit the dependency on the parameter set $\boldsymbol{\theta}$). The log marginal likelihood is given by

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2} \mathbf{y}^T \mathbf{K}_y^{-1} \mathbf{y} - \frac{1}{2} \log |\mathbf{K}_y| - \frac{N}{2} \log(2\pi). \tag{2.13}$$

Here, the term $\mathbf{K}_y$ is defined as $\mathbf{K}_y := (k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I})$, which is the covariance matrix of the noisy observations $y_1, ..., y_N$. Each term in this expression of the log marginal likelihood is easily interpretable. The quadratic form on the left-most term on the right-hand side of equation (2.13) represents the data fit term, that is how well the model hyper-parameters explain the data. The second term $\frac{1}{2}\log|\mathbf{K}_y|$ is the log determinant and is related to the model complexity. This is a penalty term that avoids the model to overfit to data and therefore impedes the lengthscales to reach too small values. In fact, the determinant $|\mathbf{K}_y|$ represents the geometric volume of the Gaussian distribution of the data, given the inputs and the parameters. If this distribution has low variance the volume tends to zero and the corresponding logarithm $\log|\mathbf{K}_y|$ tends to minus infinity. Therefore, preventing the log-determinant to reach minus infinity requires increasing the variance and reducing the data fit. The last term on the right hand side is a constant that depends on the number of data points and shows that the marginal likelihood tends to decrease as we increase the number of data points. The maximization of the log-marginal likelihood in order to find the hyper-parameters of the kernel and the likelihood variance is referred to as type II maximum likelihood (MLII). This approach for learning the hyper-parameters of a Gaussian process is an instance of a Bayesian model selection. Bayesian model selection is a method for selecting among different model instances that uses the rules of probability. The probability, in Bayesian model selection, is computed by integrating over the unknown values in that model. In the case of a Gaussian process, the integration is over the function space characterized by the kernel. In the case of Matérn$_{5/2}$ kernel, equation (2.13) can be differentiated with respect to the hyper-parameters and optimized with gradient based methods.

Finally, we address the computational complexity involved in both training a Gaussian process (learning the hyper-parameters $\boldsymbol{\theta}$) and producing posterior predictions of the black box objective function values, given an optimal set of hyper-parameters $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$. The computational bottleneck is the computation of the inverse matrix $\mathbf{K}_y^{-1}$ that appears both in the marginal likelihood in equation (2.13) and the posterior predictions in equations (2.8)–(2.9). The computational complexity of this step is cubic in the number of data points, i.e. $\mathcal{O}(N^3)$. This inverse matrix has to be recomputed every time the hyper-parameters in $\boldsymbol{\theta}$ change. However, it can also be stored and be used multiple times for computing posterior predictions in equations (2.8)–(2.9). The inverse step $\mathbf{K}_y^{-1}$ has the same complexity as computing the log-determinant $\log|\mathbf{K}_y|$ in equation (2.13) but both steps can be solved from a Cholesky decomposition of the covariance matrix of noisy observations which has cubic complexity. In fact, the log determinant is computed as twice the sum of logs of the diagonals in the Cholesky decomposition, i.e. $\log |\mathbf{K}_y| = 2\sum_i \log(\mathbf{L}_{ii})$. Where $\mathbf{L}_{ii}$ is the i-th element in the diagonal of the Cholesky matrix $\mathbf{L}$. It also holds that $\mathbf{K}_y = \mathbf{L}\mathbf{L}^T$, hence the inverse can be computed from the Cholesky decomposition. The computational complexity of posterior predictions is dominated by the matrix multiplication of the full covariance term $\mathbf{K}_y^{-1}$ times the cross-covariance term $k(\mathbf{x}_\star, \mathbf{X})$ and its transpose (as in equation (2.9)). This operation has quadratic complexity in the number of data points, i.e. $\mathcal{O}(N^2)$. These complexities will be of crucial importance when attempting to scale the Bayesian optimization approach to the high-dimensional problems featuring more than 20 dimensions.

2.3 Acquisition function

Now that we have described a suitable model for the response surface as a Gaussian process, we describe the second component in our Bayesian optimization algorithm, namely the acquisition function. The acquisition function defines a utility that is based on the values of the GP response surface. In this thesis we are mainly interested in the acquisition functions that use Gaussian processes. Therefore, in the reminder we will assume a GP for the response surface and its predictions as inputs for the acquisition function. Acquisition functions trade off exploration and exploitation during the search for the optimum. During exploration, objective evaluations concentrate where uncertainty in the response surface is high, while exploitation favors selecting input locations where predictions reach small values (in the case of a minimization problem). The acquisition function guides the search for the optimum and contributes to the data-efficiency of the optimization applied to the expensive objective f. It is assumed in general that the intermediate steps introduced for utility maximization are less computationally demanding than evaluating the true objective unconditionally. The general assumptions for the acquisition function are that i) acquisition functions usually have fixed parametric form, ii) Gaussian process predictions are cheap to compute and therefore that iii) acquisitions can be evaluated and optimized efficiently. The acquisition function lowers the evaluation budget required on the f, which may be limited, economically demanding, or time consuming to obtain.

2.3.1 Probability of improvement

The first strategy proposed in the literature of Bayesian optimization is that of *probability of improvement* (PI) [Kus64]. The strategy proposed here is to select points that maximize the probability of observing lower (or higher in a maximization of the objective function setting) function values than the ones explored during optimization. Given a Gaussian process predictive distribution with mean $\mu(\mathbf{x})$ and variance $\sigma^2(\mathbf{x})$ at $\mathbf{x}$, this acquisition function selects inputs according to the cumulative Gaussian density of the improvement $f_{\min} - \mu(\mathbf{x})$. Here the term $f_{\min} := \min\limits_{\mathbf{x} \in \mathbf{X}_t} f(\mathbf{x})$ is the current minimum observed in the exploration history and $\mathbf{X}_t = \{\mathbf{x}_1, ..., \mathbf{x}_t\}$ represents the collection of input variables selected up to iteration t in the Bayesian optimization algorithm. The expression for the probability of improvement acquisition is as follows

$$
\begin{aligned}
\mathbf{x}_{t+1} &= \arg\max_{\mathbf{x} \in \mathcal{X}} \; p(f(\mathbf{x}) \leq f_{\min}) && (2.14) \\
&= \arg\max_{\mathbf{x} \in \mathcal{X}} \; \Phi\left(\frac{f_{\min} - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right) && (2.15)
\end{aligned}
$$

Here Φ denotes the cumulative distribution of a standard Gaussian $\mathcal{N}(0,1)$. The consideration with this acquisition function is that small improvements with high probability are preferred to larger, but more uncertain gains. Samples concentrate where the current functional estimate predicts high values with confidence, often favoring pure exploitation of the posterior mean. Proposed alternative strategies have been empirically analyzed, which lower bound the minimum improvement to be at least ξ, that is $p(f(\mathbf{x}) \leq f_{\min} - \xi)$. Each positive value of the ξ parameter corresponds to a different exploration and exploitation trade-off [Liz08]. Setting a high minimum improvement pushes sampling towards less likely but more promising areas of the search space. Several empirical studies [Jon01, Liz08] have developed schedules for this parameter to control the trade off over iterations. Good practices drive the selection of $\mathbf{x}_{t+1}$ towards uncertainty reduction in the early stages (ξ large), and favor exploitation, $\xi \to 0$, later in the optimization.

2.3.2 Expected improvement

Another central utility function in Bayesian optimization is the *expected improvement* (EI) [Moč75, MTZ78]. Instead of selecting input locations according to their probability of improving the current optimum $f_{\min} := \min_{\mathbf{x}\in\mathbf{X}_t} f(\mathbf{x})$, another strategy proposed by Močkus is to formalize the improvement $I(\mathbf{x}) = \max\{0, f_{\min} - f(\mathbf{x})\}$ and then maximize its expected value with respect to the posterior prediction, i.e.

$$\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}\in\mathcal{X}} \underset{p(f|\mathbf{X},\mathbf{y})}{\mathbb{E}} \left[I(\mathbf{x})\right]. \tag{2.16}$$

The improvement $I(\mathbf{x})$ models positive differences between function values and our best observation $f_{\min}$, and the predictive distribution $p(f|\mathbf{X},\mathbf{y})$ is a short notation to denote equations (2.8)–(2.9). Given the best function value $f_{\min}$ observed so far we will obtain an improvement $I(\mathbf{x})$ if $f(\mathbf{x}) = f_{\min} - I(\mathbf{x})$. Therefore, by definition, the likelihood of the improvement is Gaussian distributed with the following form [Jon01]

$$\frac{1}{\sqrt{2\pi}\sigma(\mathbf{x})} \exp\left(-\frac{(f_{\min} - I(\mathbf{x}) - \mu(\mathbf{x}))^2}{2\sigma^2(\mathbf{x})}\right). \tag{2.17}$$

Here, the terms $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ are taken from the posterior Gaussian process predictions from equations (2.8) and (2.9), respectively. The analytical expression for the expected improvement is obtained evaluating the expectation $\mathbb{E}(I(\mathbf{x}))$ with respect to the above likelihood in equation (2.17) [Jon01]

$$EI(\mathbf{x}) = \int_{I=0}^{I=\infty} I \left[\frac{1}{\sqrt{2\pi}\sigma(\mathbf{x})} \exp\left(-\frac{(f_{\min} - I(\mathbf{x}) - \mu(\mathbf{x}))^2}{2\sigma^2(\mathbf{x})}\right)\right] dI \tag{2.18}$$

$$= \sigma(\mathbf{x})Z(\mathbf{x})\Phi(Z(\mathbf{x})) + \sigma(\mathbf{x})\phi(Z(\mathbf{x})), \tag{2.19}$$

$$Z(\mathbf{x}) = \frac{f_{\min} - \mu(\mathbf{x})}{\sigma(\mathbf{x})}, \tag{2.20}$$

where Φ and ϕ are the cumulative density function (CDF) and probability density function (PDF) of $\mathcal{N}(0,1)$, respectively. Equation (2.19) can be derived applying integration by parts to the integral in equation (2.18). The expected improvement acquisition function has been proven to find the global optimum when employed in a Bayesian optimization algorithm, that is the iterates from this method are dense [Loc97]. However, one observation is that this strategy is highly deceptive during early stages of the optimization. Small error bars around the best point, yield high utility for inputs in the neighborhood. Several examples in [Jon01] show that this method performs fairly exhaustive search around the best observed value $f_{\min}$ before starting the search for the global minimum. Therefore, also for this strategy a least-improvement parameter ξ can refine exploration performances over iterations with results similar to those shown for probability of improvement acquisition function.

2.3.3 Upper confidence bound

A more recent strategy for selecting the new input location during a Bayesian optimization iteration is the *upper confidence bound* (UCB) [SKKS10] acquisition function. Given the posterior Gaussian process predictions in equations (2.8)–(2.9), the upper confidence bound acquisition does not need any information regarding the best observation $f_{\min}$ obtained up to iteration t. The selection step for the next point can be written as

$$\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}\in\mathcal{X}} \; -\mu(\mathbf{x}) + \beta_t \sigma(\mathbf{x}). \tag{2.21}$$

Here the β_t parameter denotes the trade off between exploration and exploitation. In [SKKS10] a schedule for this parameter is proposed that achieves cumulative regret bounds. We highlight that this acquisition function is then maximized to select the next evaluation point for the BO algorithm. The value of this acquisition function is high for values of the posterior mean that are small. It will therefore prefer selecting points where the Gaussian process prediction has low mean $\mu(\mathbf{x})$ with small uncertainty and this is also referred to as the exploitation behaviour. The exploration behaviour, instead, will select locations $\mathbf{x}_{t+1}$ which feature a large value of the posterior standard deviation $\sigma(\mathbf{x})$ which represents the uncertainty in GP predictions. The parameter β_t guides the search in both these regions with high uncertainty or small posterior mean value and tuning this parameter can offer an increase in performances of the overall Bayesian optimization algorithm.

2.3.4 Other acquisition functions

So far we have presented the acquisition functions that will be used in this thesis for obtaining the experimental results. Here, we briefly describe other acquisition functions that can be employed in the Bayesian optimization algorithm. One example of acquisition that is based on the probability distribution of the minimizer $p(\mathbf{x}^*|\mathcal{D})$, where $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ denotes the data set, is *entropy search* (ES) [HS12]. The idea with this acquisition function is to select the input $\mathbf{x}_{t+1}$ that is expected to reduce the entropy of the distribution $p(\mathbf{x}^*|\mathcal{D})$ the most after evaluation. However, evaluating the largest decrease in differential entropy is intractable in continuous search spaces and requires approximations. The state of the art approximation to the method of entropy search is the proposed *predictive entropy search* (PES) [HLHG14]. Predictive entropy search exploits symmetry properties of the mutual information to derive an equivalent reformulation of the entropy search acquisition function in terms of the mutual information between the true global minimizer $\mathbf{x}^*$ and observation y. Experiments have shown equivalent or better performances with respect to discretized methods [VVW09] for entropy search, making predictive entropy search the state of the art method in entropy search approximations.

A different approach that uses the posterior distribution $p(f|\mathbf{X}, \mathbf{y})$ of the Gaussian process after having observed data is the *Thompson sampling* (TS) [Tho33] acquisition function. This simple strategy is defined in both continuous and descrete spaces for optimization. In our Bayesian optimization setting, we consider only the continuous spaces example which consists of selecting a sample function f_s from the posterior $f_s \sim p(f|\mathbf{X}, \mathbf{y})$. Then performing optimization of the acquisition function by selecting the minimizer of this sampled function as the next query point $\mathbf{x}_{t+1}$, that is $\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}\in\mathcal{X}} f_s(\mathbf{x})$. Here, evaluating the sample function $f_s(\mathbf{x})$ for each point can be done using spectral sampling techniques [Boc59, RR08, LGQCRFV10]. For a detailed explanation of the mentioned acquisition functions the reader is referred to the review in [SSW$^+$15].

One drawback of common acquisition functions such as expected improvement is to select one-step optimal points for the optimization. In fact expected improvement is one step optimal meaning that EI would lead to the optimal choice of $\mathbf{x}_{t+1}$ if we only had one BO iteration left.

Another acquisition function that has been presented recently is the *two-step look-ahead* [WF19] acquisition function. The two-step look-ahead approach extends the acquisition function to be optimal if there were two BO iterations left. The idea with the approach is to apply the envelope theorem [MS02] to estimate the gradient of this multi-step acquisition. Moreover, Monte Carlo variance reduction methods reduce the computational cost of evaluating the acquisition function and its gradient.

2.4 Challenges in high dimensions

Now that we have defined the main components of the Bayesian optimization algorithm, we mention the computational challenges that we encounter when trying to scale BO to high dimensions. Points in high dimensional spaces, namely with $D \geq 20$, become increasingly distant with the dimensionality D. In fact, the lower bound on their distance depends exponentially on the dimensinoality D of the input space [GKKW06]. Hence, the number N of data points required to cover the search space, and therefore to learn an objective function f, increases exponentially with the number of dimensions D of the search space $\mathcal{X}$ [SSW$^+$15]. This becomes challenging when using Gaussian processes to model the response surface. In fact, Gaussian processes scale cubically in the number of data points $\mathcal{O}(N^3)$. With an exponential number of data points the cubic complexity of the Gaussian process easily becomes intractable. Evaluating the acquisition function scales quadratically $\mathcal{O}(N^2)$ in the number of data points since they depend only on posterior GP predictions for computing the utility. However, it is not the number of data points to concern in the acquisition maximization, but rather the number of acquisition evaluations required to select the new point $\mathbf{x}_{t+1}$. In fact, acquisition functions are generally multi-modal non convex and therefore require an exponential number of evaluations in the number of dimensions to be maximized. In particular, common off-the-shelf numerical optimization techniques based on discretization [SLA12] and adaptive grids [BK10] require exponential computations in the number of dimensions D. This renders the maximization hard since the acquisition needs to be evaluated too many times and becomes the computational bottleneck of the overall optimization process. Gradient based methods for maximizing the acquisition function also suffer in high dimensional settings. With only a small evaluation budget, the learned response surface and the resulting acquisition function are characterized by vast flat regions interspersed with nonlinear landscapes [RLG+17]. This renders optimization with gradients of the acquisition function inherently challenging [GOH14]. The optimizer either stops because of absence of gradients or remains stuck in local optima. Therefore, the computational burden, the runtime cost and the expenses of performing inference with high-dimensional datasets become not negligible any more. Maximizing the acquisition function is not an easy task any more either. This renders the Bayesian optimization more expensive than and not suitable any more for the optimization of prohibitively costly black box objective functions.

In this setting, globally optimizing a high dimensional function is an hopeless task. In order to solve this problem we therefore need to introduce additional assumptions such as intrinsic low dimensionality of the black box objective f. This assumption basically implies that the true objective function effectively depends on a low dimensional representation of the data characterized by dimensionality $d \ll D$. This allows reformulating the problem as a low dimensional Bayesian optimization with feasible dimensionality d and solving it efficiently with off the shelf methods. The main challenge is therefore to reformulate the high dimensional problem as a low dimensional one. In the following chapters we address this aspect with two main approaches: i) a decomposition strategy that reformulates the high dimensional problem as a collection of independent low dimensional instances and ii) a compression strategy that uses an auto-encoder like probabilistic model to reformulate the Bayesian optimization problem in a low dimensional manifold of the original search space.

Chapter 3

High-dimensional Bayesian optimization with projections using quantile Gaussian processes

In this chapter, we address the high dimensional Bayesian optimization with low dimensional quantile Gaussian processes defined on independent subsets of dimensions. We have motivated in Section 2.4 that high dimensionality requires an exponential number of observation points in the number of dimensions in order to learn the response surface and an exponential number of acquisition function evaluation also in the number of dimensions D for selecting a new location for optimization. This makes Bayesian optimization in high dimensions a challenging task. What we propose here is to exploit the intrinsic low dimensionality of the objective function to reformulate the optimization problem as a collection of independent subproblems.

3.1 Related work

One common idea is to assume that the objective function is defined on a linear subspace of the original parameter space. By performing a linear mapping it is therefore possible to optimize the high dimensional objective function on a low dimensional linear subspace of the original data space. This linear subspace is characterized by dimensionality d with $d \ll D$, where D is the original dimensionality. In [WZH$^+$13] the optimization is performed on this linear embedding of the original search space because of the assumption that the objective lives only on a linear subspace of the input domain that is d-dimensional. Strong theoretical results show that performing Bayesian optimization under these assumptions is equivalent to learning and optimizing the true objective on a random embedding [WZH$^+$13]. However, robust implementation requires further deftness to account for box-constraints and non-injectivity of the mapping from the embedding to the original domain. Another method based on linear embeddings is the one presented in [GOH14]. In this work, the linear mapping necessary to reduce the dimensionality to the d-dimensional subspace is actively learned from data.

Another approach that has proven successful is the employment of decomposition strategies for the optimization problem at hand. The decomposition strategies consist of reformulating the high dimensional Bayesian optimization problem into a collection of subproblems of dimensionalities at most d. Again here the advantage with this reformulation is that the dimensionality d is much smaller than the dimensionality of the original space D, that is $d \ll D$. One example of this decomposition approach is the work presented in [KSP15]. In this work axis aligned projections of the original data space are considered for optimization as a decomposition strategy. Axis aligned projections coincide with the selection of subsets of variables from the original high dimensional space with a budget of up to d dimensions per projection. In [KSP15] an additive model is employed to compose the effects of the axis aligned projections on the underlying objective function. Basically, the objective function is assumed to be additive, that is

$$f(\mathbf{x}) = f_1(\mathbf{x}^{(1)}) + ... + f_z(\mathbf{x}^{(z)}), \tag{3.1}$$

with additive components $f_1, ..., f_z$ all defined on a partition of the original search space. By partition of the search space we mean a set of disjoint subsets $\mathcal{X}_1, ..., \mathcal{X}_z$ of variables or components $x^{(i)}$ (with up to d variables per each subset) such that their union forms the original data space $\mathcal{X}$. In [KSP15] each subproblem derived from the each axis aligned projection is optimized independently using the *Upper Confidence Bound* (UCB) acquisition function [SKKS10]. This approach scales the optimization of the acquisition linearly in the number of components by including specific structural assumptions about f. Another approach that is still based on additive components for optimization has been presented in [RSBC18] where the additive components are not defined any more on disjoint subsets of variables. In this work, there is no partition of the parameter space but rather different axis aligned projections are allowed to contain the same components or variables.

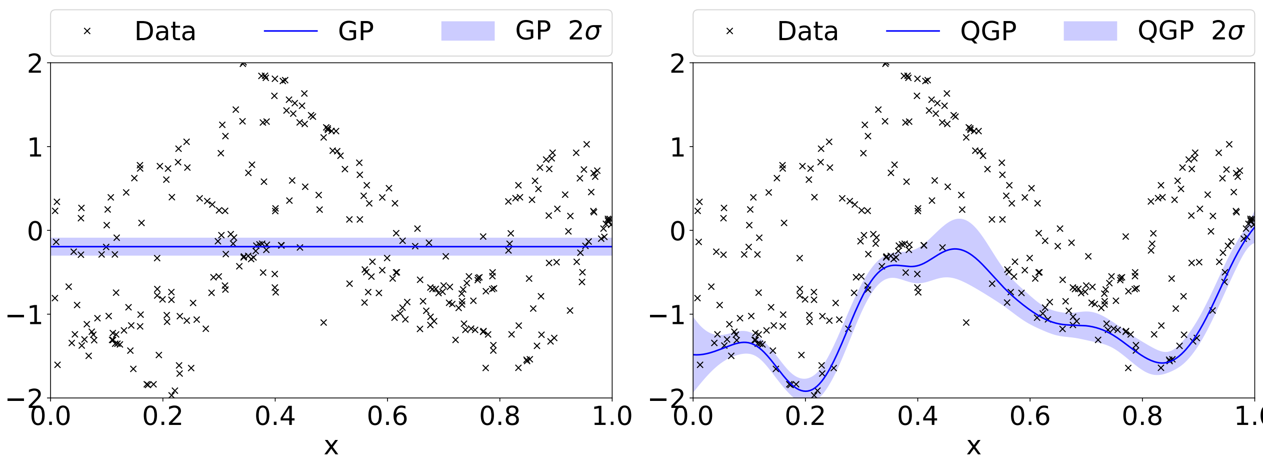
A promising direction in works related to axis aligned projections for Bayesian optimization is the automatic selection of the projection based on which dimensions are effectively relevant for the objective function optimization. In this case, the main problem is to identify a set of useful dimensions for optimization; that is dimensions on which the objective function actually depends. These dimensions are also referred to as active dimensions. In the work presented in [CCK12], the active dimensions are selected by performing a test. In this test, a squared exponential kernel, whose bandwidth depends on the number of active variables, is employed. This work provides strong theoretical results regarding the sample complexity for selecting the set of active dimensions which is logarithmic on the dimensionality of the high dimensional parameter space. Albeit these theoretical results this approach requires employing a budget of evaluations in order to discover the active dimensions prior to optimization.

Another way to exploit axis aligned projections is through the definition of tree structure dependencies among dimensions in the problem at hand. For instance, in a deep learning problem one may be interested in exploring multiple architectures of the deep network. These deep network architectures may be characterized by their own set of hyper-parameters, which may be shared across models. These shared and non shared parameters could be expressed in a decision tree having the hyperparameters as leaf nodes and different architectures as nodes of the tree. In the work presented in [JAGS17] this tree structure dependency is exploited to transfer information between overlapping decision variables. In particular, in [JAGS17] a novel surrogate model for the response surface is introduced that allows to combine independent Gaussian processes with a linear model that captures the tree-based dependency structure.

Another approach based on axis aligned projections is proposed in [UBC^{+}16] where a different decomposition of the input is performed by randomly selecting subsets of the input parameters and assigning a different model for each subset. Each model is then trained with a separate dataset to address the problem of inconsistencies introduced by axis-aligned projections.

Axis aligned projections cause multiple observations for the same input variable $\mathbf{x}$ which we refer to as *inconsistencies* or *ambiguities*. For instance, if we consider a two-dimensional function we can observe different function values at two distinct points that share the same first coordinate, that is the following input-function-value pairs can be observed $([x_1, x_2], f(x_1, x_2))$, $([x_1, \tilde{x}_2], f(x_1, \tilde{x}_2))$. Now, if we apply an axis-aligned projection on the first coordinate, that is we remove from the data points the x_2 and $\tilde{x}_2$ variables, respectively we end up with the following projected dataset: $([x_1], f(x_1, x_2)), ([x_1], f(x_1, \tilde{x}_2))$. Since the function values may be different $f(x_1, x_2) \neq f(x_1, \tilde{x}_2)$ for the two data points, we end up with multiple function values for the same projected input x_1 which we refer to as inconsistency or ambiguity. An example of this shortcoming is presented in Figure 3.1. As a result, optimization in these lower-dimensional spaces suffers from observation ambiguity or inconsistency when applying the same principle to noisy observations instead of noiseless function values. To amend the inconsistency issue in [UBC+16] the authors adopt multiple separate subsets of experiments resulting in a strategy for optimization that requires much data. This approach has shown to perform well in practice but has been characterized by a data inefficient strategy because of the definition of a different dataset for each model. In this work, there is no possibility to share information (data) across models.

In this dissertation, we propose a scalable method based on axis-aligned projections that overcomes issues inherent in this type of projections. We formulate an optimization approach based on independent sub-problems for each subset of $d \ll D$ parameters. This reduces the complexity of both learning a response surface and optimizing the acquisition function in a low dimensional space. We address the issue of inconsistencies with a sensible choice of the model by using a quantile Gaussian process (QGP) [BBC12] on each axis aligned projection. The quantile Gaussian process is a Gaussian process characterized by an asymmetric Laplace likelihood. The key idea behind the QGP is that we only model a lower τ-quantile of function values, which also allows us to retain a simpler explanation of the data. In fact, the overall result



(a) GP (b) QGP

Figure 3.1: (a) vanilla GP hyper-parameter learning can result in unrealistic estimates and largely useless generalization capabilities. This is a clear sign of underfitting where the data with inconsistency is modeled as noise. The resulting acquisition function from this model will favor pure exploitation leading to pre-mature convergence of the algorithm. (b) the Quantile GP overcomes the issue of inconsistencies by explicitly modeling only a quantile function.

with the quantile Gaussian process is an automatic selection of most promising observations for minimization where we can generalize with the response surface. The QGP maintains well-calibrated confidence bounds in posterior predictions in the presence of noisy multiplicity of outputs see Figure 3.1. A more detailed description of the model is provided in Section 3.3

3.2 High-dimensional Bayesian optimization with projections

We propose a novel Bayesian optimization algorithm based on axis-aligned projections that uses quantile regression models for learning a low-dimensional projection of the response surface. Under the assumption that the black-box function is effectively lower-dimensional, projections onto d-dimensional features tackle the curse of dimensionality for both the learning of the response surface and the maximization of the acquisition function.

We select a maximum of z possible axis-aligned projections that partition the D-dimensional input/parameter space, such that $\mathcal{X} = \bigcup_{i=1}^{z} \mathcal{X}_i$ and $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$ for $i \neq j$ and $i, j = 1, ..., z$.

This convention allows us to partition the dimensions into a maximum of z projections, which we refer to as *components*. We then define the projection as a set of d coordinates $proj(i) = \{p_{i,1}, ..., p_{i,d}\}$, to select from the original parameter space for $i = 1, ..., z$. Given an input vector $\mathbf{x} = [x_1, ..., x_D] \in \mathbb{R}^D$, we identify its i-th projection as $\mathbf{x}^{(i)} = [x_{p_{i,1}}, ..., x_{p_{i,d}}] \in \mathbb{R}^d$, that is from the high-dimensional vector of parameters we select the coordinates with indices $p_{i,1}, ..., p_{i,d}$. Essentially, the i-th axis aligned projection selects a subset of d dimensions of the high dimensional ambient space where the selected dimensions are those with indices $\{p_{i,1}, ..., p_{i,d}\}$. The projected vector is defined in the low-dimensional space $\mathcal{X}_i \subset \mathbb{R}^d$ and we say that this d-dimensional space is defined by the projection $proj(i)$. When performing Bayesian optimization from axis-aligned projected data, we consider the data set $\{\mathcal{X}_i, \mathcal{Y}\}$ with lower-dimensional inputs $\mathbf{x}^{(i)}$ and all observations.

Projections along the axes cause inconsistencies or ambiguities, that is multiple observation values $y_1, ..., y_m$ for the same input location. For instance, if we observe y_1, y_2 from a function with two-dimensional inputs, i.e., $y_1 = \hat{f}(x_1, x_2) + \varepsilon_1, y_2 = \hat{f}(x_1, \tilde{x}_2) + \varepsilon_2$, with $\varepsilon_1, \varepsilon_2$ being additive[1] identically distributed and independent Gaussian noise, and we plot them w.r.t. the first coordinate we obtain multiple output values in correspondence to x_1, that is $(x_1\ ,\ \hat{f}(x_1, x_2))$ and $(x_1\ ,\ \hat{f}(x_1, \tilde{x}_2))$. This multiplicity is not caused by noise, but it originates from the variation of the unobserved parameter x_2 in the original domain. The effect of these inconsistencies/ ambiguities is illustrated in Figure 3.1. A standard GP would model this multiplicity of outputs as additive Gaussian noise. This modeling may result in a mis-interpretation of data as unstructured noise. Figure 3.1 shows an example of this shortcoming. As can be seen, the presence of inconsistencies negatively affects the hyper-parameter optimization resulting in unrealistic estimates of the characteristic lengthscales, kernel variance and measurement noise variance. This shortcoming also affects the capability of generalizing with the learned Gaussian process model. In fact, the inconsistencies are interpreted as measurement noise and the model is incapable to generalize showing clear signs of underfitting. A particular sign of underfitting is the large value of the characteristic lengthscales even in the presence of slopes in the data.

[1]By additive we mean that the measurement noise is added to the function value

Algorithm 2 Quantile-Gaussian process based Bayesian optimization. Main steps of a Bayesian optimization algorithm with projections. The inner loop iterates over the z components and trains a different quantile-GP model for each projection. The training of each quantile-GP model in line 5 is performed by maximizing the marginal likelihood in equation (3.5) and selects the kernel lengthscale hyper-parameters, the kernel variance and the measurement noise variance. The update in line (8) in the outer loop concatenates all the selected updates $\mathbf{x}_{t+1}^{(i)}$.

```
1:  **Inputs:** $\mathbf{X}_0 = \{\mathbf{x}_1, ..., \mathbf{x}_{N_0}\} \in \mathbb{R}^{N_0 \times D}$, $\mathbf{y}_0 = \{y_1, ..., y_{N_0}\} \in \mathbb{R}^{N_0}$
2:  **Set:** $d \ll D$
3:  **for** $t = 0, 1, 2, ..., T_{\text{end}} - 1$ **do**
4:      **for** $i = 1, ..., z$ **do**
5:          Train $i$-th Quantile-GP model $f_i | \mathbf{X}_t^{(i)}, \mathbf{y}_t$
6:          Select $i$-th update $\mathbf{x}_{t+1}^{(i)} = \underset{\mathbf{x}^{(i)} \in \mathcal{X}_i}{\text{argmax}}\ \alpha(\mathbf{x}^{(i)} | f_i)$
7:      **end for**
8:      Update input with all components $\mathbf{x}_{t+1} = \cup_{i=1}^{z} \mathbf{x}_{t+1}^{(i)}$
9:      Observe objective value $y_{t+1} = f(\mathbf{x}_{t+1}) + \varepsilon_{t+1}$
10:     Augment Data set $\mathbf{X}_t \cup \{\mathbf{x}_{t+1}\}$, $\mathbf{y}_t \cup \{y_{t+1}\}$
11: **end for**
12: **Return** $\mathbf{x}^* = \arg\min \mathbf{y}_t$
```

For our purpose of obtaining reliable posterior predictions after training, we are interested in removing such inconsistencies, e.g., by automatically selecting only the best (lowest) observations for each parameter sub-configuration. Selecting extreme (small) observations is intuitive in our minimization context, and we validate this choice with empirical results. Quantile regression provides a method for function estimation that effectively embodies this notion of automatic selection. We detail the steps of our method with quantile GP models for each projection in Algorithm 2. Note that in Algorithm 2, in line 5, the variable $\mathbf{X}_t^{(i)}$ collects all the projected data points in the i-th projection, that is $\mathbf{X}_t^{(i)} = \{\mathbf{x}_1^{(i)}, ..., \mathbf{x}_{N_t}^{(i)}\}$, where the N_t variable denotes the number of data points available at the t-th iteration and the i-th projection takes values in $i = 1, ..., z$.

3.3 Quantile Gaussian process regression

We are interested in modeling a proportion of the data with a Gaussian process. This proportion is referred to as quantile, τ, and defines the probability $p(y \leq \mu_\tau) = \tau$ of observations y to be below the functional estimate μ_τ [TLSS06]. Quantile regression [KH01] is comprised of techniques for estimating and expressing μ_τ through conditional quantile functions. The basic intuition behind quantile regression is that minimizing the $l1$-loss function, $\sum_{i=1}^{N_t} |y_i - \mu_\tau(\mathbf{x}_i)|$ yields a functional estimator of the median, which corresponds to quantile $\tau = 0.5$. For an arbitrary τ, direct estimation of the quantile functions is obtained by minimizing a tilted loss function (pinball loss)

$$l_\tau(\xi) = \begin{cases} \tau\xi & \text{if } \xi \geq 0 \\ (\tau - 1)\xi & \text{if } \xi < 0, \end{cases} \tag{3.2}$$

where $\xi = y_i - \mu_\tau(\mathbf{x}_i)$ for $i = 1, ..., N_t$. The regression problem that optimizes the cumulative loss $\sum_{i=1}^{N_t} l_\tau(y_i - \mu_\tau(\mathbf{x}_i))$ consistently produces τ-th quantile function estimates [TLSS06].

In our predictions, we model the uncertainty of our estimate μ_τ as a posterior probability over function values derived in a Bayesian framework. The quantile-Gaussian process model (QGP) [BBC12] allows for such a formulation and computation of posterior predictions through approximate inference via Expectation Propagation (EP) [Min01]. We introduce a standard Gaussian process prior over quantile functions, i.e., $\mu_\tau \sim \mathcal{GP}(m, k)$, and reformulate the tilted loss in equation (3.2) in terms of a renormalized reward

$$\mathbf{R}(y_i, \mu_\tau) = Z_i \exp\left(l_\tau(\mu_\tau, y_i)\right), \tag{3.3}$$

where Z_i is a normalizing constant, and $\mathbf{R}(y_i, \mu_\tau)$ evaluates the Asymmetric Laplace Distribution (ALD). The basic intuition behind this definition is also displayed in Figure 3.2 for a quantile $\tau = 0.1$. The reward represents the likelihood $p(y_i|\mu_\tau, \mathbf{x}_i, \boldsymbol{\theta}_{GP})$ of each input $\mathbf{x}_i$ and model hyper-parameters $\boldsymbol{\theta}_{GP}$, which includes the characteristic lengthscales of the kernel, the kernel variance and the measurement noise variance. The joint distribution of the probabilistic model

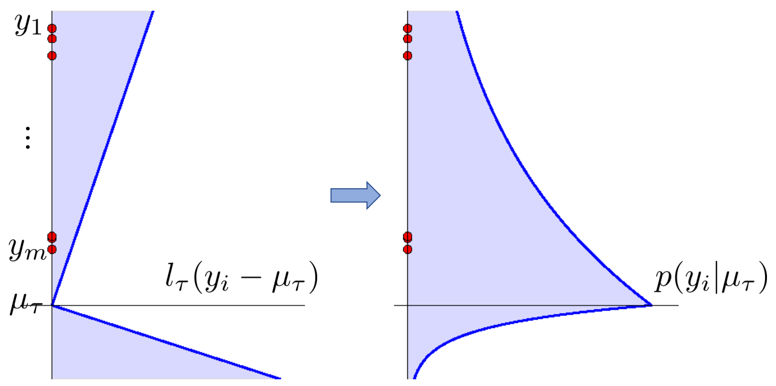


Figure 3.2: Two different approaches to quantile regression. Red dots show inconsistencies, i.e. different observations $y_1, ..., y_m$ for the same input $\mathbf{x}$. Left (direct estimation): the blue line shows the tilted loss for $\tau = 0.1$, the observations y above the functional estimate generate a loss 10-times smaller than those that appear at the same distance below μ_τ. Right (Bayesian formulation): the blue line represents how likely data points are given the quantile model μ_τ.

of the quantile Gaussian process is the following

$$p(\mathbf{y}_t, \mu_\tau | \mathbf{X}_t, \boldsymbol{\theta}_{GP}) = p(\mathbf{y}_t | \mu_\tau, \mathbf{X}_t, \boldsymbol{\theta}_{GP}) p(\mu_\tau | \boldsymbol{\theta}_{GP}) \tag{3.4}$$

Where $p(\mathbf{y}_t, \mu_\tau | \mathbf{X}_t, \boldsymbol{\theta}_{GP})$ is the joint distribution of observations and function values given the inputs $\mathbf{X}_t$ and hyper-parameters $\boldsymbol{\theta}_{GP}$. The distribution $p(\mathbf{y}_t | \mu_\tau, \mathbf{X}_t, \boldsymbol{\theta}_{GP})$ is the asymmetric Laplace likelihood and the distribution $p(\mu_\tau | \boldsymbol{\theta}_{GP})$ is the Gaussian process prior.

We perform training of the quantile Gaussian process hyper-parameters via a type-II maximum likelihood approach [RW06], that is we select the characteristic lengthscale hyper-parameters of the kernel by maximizing the marginal likelihood at each Bayesian optimization iteration, that is

$$\arg\max_{\boldsymbol{\theta}_{GP}} \int p(\mathbf{y}_t | \mu_\tau, \mathbf{X}_t, \boldsymbol{\theta}_{GP}) p(\mu_\tau | \boldsymbol{\theta}_{GP}) d\mu_\tau. \tag{3.5}$$

Here the vector $\mathbf{y}_t$ is the collection of all observations encountered during optimization up to iteration t, the μ_τ is the functional estimate modeled as a Gaussian process and the matrix $\mathbf{X}_t$ is the collection of all locations selected during the optimization up to iteration t. The vector $\boldsymbol{\theta}_{GP}$ includes all the hyper-parameters of the Gaussian process, that is characteristic lengthscales and kernel variance. The integral in equation (3.5) is intractable and we approximate it via

Expectation propagation.

3.4 Expectation propagation

Expectation propagation (EP) [Min01] is an approximate inference method based on local approximations used when both the posterior predictions and the marginal likelihood are analytically intractable due to non Gaussian likelihood $p(\mathbf{y}_t|\mu_\tau, \mathbf{X}_t, \boldsymbol{\theta}_{GP})$. The non conjugate prior for this likelihood renders the computation of the integral in equation (3.5) intractable and expectation propagation provides an approximation for both the marginal likelihood and the posterior predictions. We preferred expectation propagation to the Laplace approximation because the Laplace approximation provides a more coarse solution than EP for approximating the intractable posterior with a Gaussian distribution [VGS+20]. We also considered EP compared to variational methods. In our problem we considered the true density of the posterior to be multimodal. In this case, expectation propagation allows to find an average of the modes. Variational methods, instead, would pick one of the modes. Therefore, we preferred EP in order to perform mode averaging on the intractable posterior.

Expectation propagation for Gaussian processes expresses the likelihood, $p(\mathbf{y}_t|\mu_\tau, \mathbf{X}_t, \boldsymbol{\theta}_{GP})$, with a product of unnormalized Gaussian distributions in the latent variable $\mu_{\tau,i}$ (where the i in $\mu_{\tau,i}$ denotes the i-th functional estimate value of μ_τ in correspondence to the input $\mathbf{x}_i$, for each data point that is for $i = 1, ..., N_t$) called *local likelihood approximations* $\tilde{\pi}_i = \tilde{Z}_i \mathcal{N}(\mu_{\tau,i}; \tilde{\mu}_i, \tilde{\sigma}_i^2)$, in our case

$$p(\mathbf{y}_t|\mu_\tau, \mathbf{X}_t, \boldsymbol{\theta}_{GP}) \approx \prod_{i=1}^{N_t} \tilde{\pi}_i := \prod_{i=1}^{N_t} \tilde{Z}_i \mathcal{N}(\mu_{\tau,i}; \tilde{\mu}_i, \tilde{\sigma}_i^2). \tag{3.6}$$

In our setting, the rewards are independent for each observation y_i collected during the optimization so that the model likelihood $p(\mathbf{y}_t|\mu_\tau, \mathbf{X}_t, \boldsymbol{\theta}_{GP})$ factorizes over the rewards, that is

$$p(\mathbf{y}_t|\mu_\tau, \mathbf{X}_t, \boldsymbol{\theta}_{GP}) = \prod_{i=1}^{N_t} \mathbf{R}(y_i, \mu_{\tau,i}). \tag{3.7}$$

Expectation propagation approximates each of these N_t-likelihood factors with a local Gaussian approximation; we therefore apply an approximation with N_t local Gaussian likelihoods. Each local approximation is characterized by the *site parameters*: $\tilde{Z}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2$, for $i = 1, ..., N_t$, where the effect of the normailzation constants, $\tilde{Z}_i$, on the marginal likelihood can be expressed as a function of the site parameters $\tilde{\mu}_i, \tilde{\sigma}_i^2$, [RW06]. These site parameters are are the targets of the EP algorithm and are updated in an iterative process until convergence. The convergence guarantee for the expectation propagation procedure has not been proven but rather conjectured [RW06] for log-concave likelihoods, such as the asymmetric Laplace distribution and it has been reported [RW06] that EP works relatively well for Gaussian process models.

The update of each local approximation is performed such that each $\tilde{\pi}_i$ will contribute to the posterior as the original likelihood in $p(\mathbf{y}_t|\mu_\tau, \mathbf{X}_t, \boldsymbol{\theta}_{GP})$, still retaining nice properties of analytical integration against Gaussian distributions. We denote the product of the local likelihood approximations with the following Gaussian distribution

$$\prod_{i=1}^{N_t} \tilde{\pi}_i = \mathcal{N}(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}) \prod_{i=1}^{N_t} \tilde{Z}_i, \tag{3.8}$$

where the vector $\tilde{\boldsymbol{\mu}}$ contains the means $\tilde{\mu}_i$ of the unnormalized local likelihood approximations and the covariance matrix is a diagonal matrix containing the variances of each unnormalized local likelihood approximation on the main diagonal, that is $\tilde{\Sigma}_{ii} = \tilde{\sigma}_i^2$, for $i = 1, ..., N_t$. Expectation propagation approximates the true posterior $p(\mu_\tau|\mathbf{X}_t, \mathbf{y}_t)$ with the approximation $q(\mu_\tau|\mathbf{X}_t, \mathbf{y}_t)$, which is defined as follows

$$q(\mu_\tau|\mathbf{X}_t, \mathbf{y}_t) := \frac{1}{Z_{EP}} p(\mu_\tau|\boldsymbol{\theta}_{GP}) \prod_{i=1}^{N_t} \tilde{\pi}_i = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{3.9}$$

$$\boldsymbol{\Sigma} = (\mathbf{K}^{-1} + \tilde{\boldsymbol{\Sigma}}^{-1})^{-1} \tag{3.10}$$

$$\boldsymbol{\mu} = \boldsymbol{\Sigma}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{\mu}}. \tag{3.11}$$

The approximate posterior is obtained applying the rule of a product of two Gaussians. The two Gaussians involved in the product are the Gaussian process prior $p(\mu_\tau|\boldsymbol{\theta}_{GP})$ and the product of local likelihoods approximations $\prod_{i=1}^{N_t} \tilde{\pi}_i$. The result of a product of two Gaussians is an unnormalized Gaussian distribution. This product is then renormalized with the marginal likelihood factor Z_{EP}. This implies that the posterior approximation is distributed as a Gaussian. The term Z_{EP} is expectation propagation's approximation to the normalization constant in Bayes' rule for deriving the posterior distribution, that is $Z_{EP} = q(\mathbf{y}_t|\mathbf{X}_t)$, where $q(\mathbf{y}_t|\mathbf{X}_t)$ is the approximate marginal likelihood. Moreover the probability $p(\mu_\tau|\boldsymbol{\theta}_{GP})$ is the Gaussian process prior, $\boldsymbol{\mu}$ is the mean vector of the approximate posterior and is derived from the rule of the product of two Gaussians (the Gaussian prior of the GP and the Gaussian product of local likelihood approximations). Similarly, the covariance matrix $\boldsymbol{\Sigma}$ of the approximate posterior is derived from the same rules of products of Gaussians and the $\mathbf{K}$ matrix is the kernel matrix of the Gaussian process prior, that is $\mathbf{K} = [k(\mathbf{x}_i, \mathbf{x}_j)]_{i,j}$ for $i, j = 1, ..., N_t$.

The idea with expectation propagation is to start from the approximate posterior as defined in equations (3.9)–(3.11) and remove the contribution of a single local likelihood approximation $\tilde{\pi}_i$ (the i-th component) to obtain the *cavity distribution*. Then, expectation propagation combines the cavity distribution with the exact likelihood $p(y_i|\mu_\tau, \mathbf{X}_t, \boldsymbol{\theta}_{GP})$ of the i-th observation y_i, resulting in a non Gaussian marginal. As an additional step, EP projects the non Gaussian marginal to a Gaussian distribution with moment matching. Finally, the site parameters $\tilde{Z}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2$ of the i-th local likelihood approximation are computed such that the posterior have the same marginal as that computed with the moment matching at the previous step. These steps are performed for each i-th local likelihood approximation, that is for $i = 1, ..., N_t$. This whole procedure is repeated until convergence.

To describe the above steps more in detail, here we report the probabilities derived from the main steps of expectation propagation. First we consider the i-th marginal of the approximate posterior described in equations (3.9)–(3.11)

$$q(\mu_{\tau,i}|\mathbf{X}_t, \mathbf{y}_t) = \mathcal{N}(\mu_{\tau,i}; \mu_i, \sigma_i^2), \tag{3.12}$$

that is the i-th component of the approximate posterior where the μ_i and the σ_i^2 are the i-th components of $\boldsymbol{\mu}$ and Σ, respectively such that $\sigma_i^2 = \Sigma_{ii}$. The first step in expectation propagation is therefore to compute the i-th cavity distribution $q_{-i}(\mu_{\tau,i})$. The i-th cavity distribution $q_{-i}(\mu_{\tau,i})$ is obtained removing the contribution of the i-th local likelihood approximation from the approximate posterior described in equations (3.9)–(3.11), that is

$$q_{-i}(\mu_{\tau,i}) := \mathcal{N}(\mu_{\tau,i}; \mu_{-i}, \sigma_{-i}^2), \tag{3.13}$$

$$\sigma_{-i}^2 = (\sigma_i^{-2} - \tilde{\sigma}_i^{-2})^{-1}, \tag{3.14}$$

$$\mu_{-i} = \sigma_{-i}^2(\sigma_i^{-2}\mu_i - \tilde{\sigma}_i^{-2}\tilde{\mu}_i). \tag{3.15}$$

Expressions (3.14)–(3.15) can be verified by just multiplying the i-th cavity distribution by the i-th local likelihood approximation $\tilde{Z}_i\mathcal{N}(\mu_{\tau,i}; \tilde{\mu}_i, \tilde{\sigma}_i^2)$ and using the rules for computing the product of two Gaussian distributions to obtain the approximate marginal posterior in equation (3.12). The second step is to multiply the cavity distribution by the exact i-th likelihood factor $p(y_i|\mu_\tau, \mathbf{X}_t, \boldsymbol{\theta}_{GP})$ to obtain a non-Gaussian marginal, that is the current posterior approximation with i-th contribution from the original likelihood

$$h_i = q_{-i}(\mu_{\tau,i})\ p(y_i|\mu_\tau, \mathbf{X}_t, \boldsymbol{\theta}_{GP}). \tag{3.16}$$

A third step is to project the combination h_i of the marginal cavity distribution $q_{-i}(\mu_{\tau,i})$ with the exact likelihood $p(y_i|\mu_\tau, \mathbf{X}_t, \boldsymbol{\theta}_{GP})$ to an un-normalized Gaussian $\hat{q}_i(\mu_{\tau,i})$ via moment matching:

$$\hat{q}_i(\mu_{\tau,i}) := \hat{Z}_i\mathcal{N}(\hat{\mu}_i, \hat{\sigma}_i^2) = \text{proj}\left[h_i\right]. \tag{3.17}$$

Here we have introduced the projection operator proj, that matches the first two moments of the distribution h_i to a Gaussian distribution $\hat{q}_i$. Since the distribution is an un-normalized Gaussian also the zeroth moment that is the normalizing constant should match. Moreover note that since the $\hat{q}_i(\mu_{\tau,i})$ is an un-normalized Gaussian the moment matching operation minimizes the KL divergence between the h_i distribution and the un-normalized approximation $\hat{q}_i(\mu_{\tau,i})$, that is $KL(h_i||\hat{q}_i)$. The fourth and last step removes the contribute of the cavity distribution from $\hat{q}_i(\mu_{\tau,i})$ to update the set of site parameters $\tilde{Z}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2$. This process is iterated for all factors of the likelihood which coincide with the number of points collected by the optimization up to iteration t, that is for $i = 1, ..., N_t$. Finally, this loop is repeated until convergence.

Algorithm 3 Expectation Propagation: in each update of site i, the expectation propagation procedure computes the i-th marginal of the approximate posterior in equations (3.9)–(3.11). Line 5 computes the *cavity distribution*, $q_{-i}(\mu_{\tau,i})$. Expectation propagation then applies the contribution of the i-th original likelihood to the cavity q_{-i} to obtain the i-th *hybrid distribution*: h_i. The hybrid distribution is then projected to an un-normalized Gaussian, $\hat{q}_i$, via moment matching (by minimizing $\mathrm{KL}(h_i \| \hat{q}_i)$). The local approximation is then obtained removing the cavity term in Line 8

```
1: **Initialize:** $\tilde{\sigma}_i^{-2}\tilde{\mu}_i = 0,\ \tilde{\sigma}_i^{-2} = 0$ for $i = 1, ..., N_t$
2: **for** $j = 0, 1, 2, ...$until convergence **do**
3:    **for** $i = 1, ..., N_t$ **do**
4:       Compute the $i$-th marginal of the approximate posterior distribution: $q(\mu_{\tau,i}|\mathbf{X}_t, \mathbf{y}_t)$
5:       Evaluate the $i$-th cavity distribution: $q_{-i}(\mu_{\tau,i}) = \mathcal{N}(\mu_{\tau,i}; \mu_{-i}, \sigma^2_{-i})$
6:       Multiply the cavity by the $i$-th true likelihood: $h_i = q_{-i}(\mu_{\tau,i})\ p(y_i|\mu_\tau, \mathbf{X}_t, \boldsymbol{\theta}_{GP})$
7:       Project to an un-normalized Gaussian distribution: $\hat{q}_i = \mathrm{proj}\,[h_i]$
8:       Remove the contribute of the cavity distribution from $\hat{q}_i(\mu_{\tau,i})$: $\hat{q}_i/q_{-i}(\mu_{\tau,i})$
9:       Update: $\tilde{\mu}_i, \tilde{\sigma}_i^2$
10:   **end for**
11: **end for**
12: **Return:** $\tilde{\mu}_i, \tilde{\sigma}_i^2$ for $i = 1, ..., N_t$
```

After convergence, each local approximation $\tilde{\pi}_i$ will contribute to the posterior as the original likelihood in $p(\mathbf{y}|\mu_\tau, \mathbf{X}_t, \boldsymbol{\theta}_{GP})$, still retaining nice properties of analytical integration against Gaussian distributions. Algorithm 3 shows the main steps involved in the approximate inference procedure (the original implementation of the algorithm uses the natural parameters, $\tilde{\tau}_i = \tilde{\sigma}_i^{-2}$ and $\tilde{\nu}_i = \tilde{\sigma}_i^{-2}\tilde{\mu}_i$ and these are initialized to zero). Further details on the expectation propagation algorithm as approximate inference method are provided in [Min01][RW06].

3.4.1 Posterior Gaussian process predictions

We have described the entire expectation propagation procedure with the derivation of the steps necessary to update the site parameters. Now we detail the equations for obtaining the posterior predictions given the convergence of the outer loop of Algorithm 3. The posterior predictions can be derived in closed form and resemble the equations for Gaussian process posterior predictions in expressions (2.8)–(2.9). The only difference is in the noise term: instead of being an identity matrix it becomes a diagonal matrix, which contains the site parameters. The posterior mean is derived as follows

$$\begin{aligned}\mathbb{E}[\mu_{\tau,\star}|\mathbf{X}_t,\mathbf{y}_t,\mathbf{x}_\star] &= k(\mathbf{x}_\star,\mathbf{X}_t)\mathbf{K}^{-1}\boldsymbol{\mu} &(3.18)\\ &= k(\mathbf{x}_\star,\mathbf{X}_t)\mathbf{K}^{-1}(\mathbf{K}^{-1}+\tilde{\boldsymbol{\Sigma}}^{-1})^{-1}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{\mu}} &(3.19)\\ &= k(\mathbf{x}_\star,\mathbf{X}_t)(\mathbf{K}+\tilde{\boldsymbol{\Sigma}})^{-1}\tilde{\boldsymbol{\mu}}. &(3.20)\end{aligned}$$

Here, $\mu_{\tau,\star}$ is the value of the functional estimate μ_τ at the test location $\mathbf{x}_\star$ and the term $k(\mathbf{x}_\star,\mathbf{X}_t)$ evaluates the cross-covariance between the test point $\mathbf{x}_\star$ and the training inputs $\mathbf{X}_t$. The first equality in equation (3.18) is obtained combining the Gaussian process predictive mean in equation (2.8) with the posterior mean $\boldsymbol{\mu}$ defined in equation (3.11). The second equality in expression (3.19) is obtained expanding the definition of the approximate posterior mean in equation (3.11). The last equality in expression (3.20) is obtained from well-known matrix identities [PP08]. The posterior variance is defined as follows

$$\begin{aligned}\mathbb{V}[\mu_{\tau,\star}|\mathbf{X}_t,\mathbf{y}_t,\mathbf{x}_\star] &= k(\mathbf{x}_\star,\mathbf{x}_\star)-k(\mathbf{x}_\star,\mathbf{X}_t)[\mathbf{K}^{-1}-\mathbf{K}^{-1}(\mathbf{K}^{-1}+\tilde{\boldsymbol{\Sigma}}^{-1})^{-1}\mathbf{K}^{-1}]k(\mathbf{x}_\star,\mathbf{X}_t)^T &(3.21)\\ &= k(\mathbf{x}_\star,\mathbf{x}_\star)-k(\mathbf{x}_\star,\mathbf{X}_t)(\mathbf{K}+\tilde{\boldsymbol{\Sigma}})^{-1}k(\mathbf{x}_\star,\mathbf{X}_t)^T. &(3.22)\end{aligned}$$

The kernel value $k(\mathbf{x}_\star,\mathbf{x}_\star)$ describes the covariance at the test input as a function of $\mathbf{x}_\star$. In equation (3.21), there are two terms after the self covariance $k(\mathbf{x}_\star,\mathbf{x}_\star)$ in square brackets. The first term is due to the variance of the functional estimator $\mu_{\tau,\star}$ at test point $\mathbf{x}_\star$ if we condition on the posterior μ_τ; that is the first term can be summarized by the following expression

$$\mathbb{E}_{p(\mu_{\tau,\star}|\mathbf{X}_t,\mathbf{x}_\star,\mu_\tau)}[(\mu_{\tau,\star} - \mathbb{E}[\mu_{\tau,\star}|\mathbf{X}_t,\mathbf{x}_\star,\mu_\tau])^2] = k(\mathbf{x}_\star,\mathbf{x}_\star) - k(\mathbf{x}_\star,\mathbf{X}_t)\mathbf{K}^{-1}k(\mathbf{x}_\star,\mathbf{X}_t)^T. \tag{3.23}$$

The second term in the square brackets of equation (3.21) is due to the fact that the expectation $\mathbb{E}[\mu_{\tau,\star}|\mathbf{X}_t,\mathbf{x}_\star,\mu_\tau]$ on the left hand side of equation (3.23) depends on μ_τ. Therefore, we add the variance term derived from the posterior over μ_τ

$$\begin{aligned}\mathbb{E}_{q(\mu_\tau|\mathbf{X}_t,\mathbf{y}_t)}[(\mathbb{E}[\mu_{\tau,\star}|\mathbf{X}_t,\mathbf{x}_\star,\mu_\tau]-\mathbb{E}[\mu_{\tau,\star}|\mathbf{X}_t,\mathbf{y}_t,\mathbf{x}_\star])^2] =& \\ k(\mathbf{x}_\star,\mathbf{X}_t)\mathbf{K}^{-1}(\mathbf{K}^{-1}+\tilde{\boldsymbol{\Sigma}}^{-1})^{-1}\mathbf{K}^{-1}k(\mathbf{x}_\star,\mathbf{X}_t)^T.&\end{aligned} \tag{3.24}$$

Finally, we obtain equation (3.22) from equation (3.21) by applying the matrix inversion lemma [RW06]. For more detailed derivation of the posterior Gaussian process predictions with expectation propagation the reader is referred to [RW06].

3.5 Experiments

In this section, we assess the quantile Gaussian process model for Bayesian optimization on axis-aligned projections. In our analysis, we dedicate a set of experiments to validate our choice of the quantile τ with empirical evidence. In high-dimensional settings, we test our approach on the commonly assumed additivity property by imposing and violating this assumption. In addition, we include an empirical analysis of performances when the axis-aligned assumption is violated, that is under an arbitrary rotation of the original domain.

Performing Bayesian optimization on subsets of dimensions, we define fixed groups and update each partition component in parallel during one optimization step. We avoid over-fitting to a single acquisition function comparing performances across a set of acquisitions: *expected improvement* (EI), [Moč75], *upper confidence bound* (UCB) [SKKS10], and *probability of improvement* (PI) [Kus64]. Note that for the improvement based acquisition functions, namely expected improvement and probability of improvement, we need the value of the true optimum

$f(\mathbf{x}^*)$ of the objective function. Since we do not access the true optimum either before or during the optimization, we compute an approximation of it by selecting the best noisy observation collected up to iteration t, in other words $y_{\min} := \min \mathbf{y}_t$. This change in the definition of the improvement based acquisition functions favors aggressive exploitation behavior by probability of improvement. For the Gaussian process model in each baseline, we select the Matérn$_{5/2}$ kernel.

In our setting, we set the exploration exploitation trade off β_t parameter of UCB acquisition function in equation (2.21) to $\sqrt{3}$, which is a common choice in Bayesian optimization literature [WMHD17] and has been shown to perform well in practice [MKD20]. The procedure performed for the maximizing of the acquisition function is identical for each baseline: i) we first apply a random search optimization step with 5000 samples drawn uniformly at random, ii) then we select the 100 locations corresponding to the highest acquisition function values and perform gradient based optimization from these 100 starting locations. For the gradient based maximization of the acquisition function we employ the L-BFGS-B scipy optimizer [BLNZ95, ZBLN97].

Each Bayesian optimization progression curve displays the mean and standard error of the immediate logarithmic regret $\log_{10} |f(\mathbf{x}_{\text{best}}(t)) - f_{\min}|$, where $f_{\min}$ is the true minimum of f, that is the objective function evaluated at the true minimizer $\mathbf{x}^*$, therefore $f_{\min} = f(\mathbf{x}^*)$, and $\mathbf{x}_{\text{best}}(t)$ is the location of the lowest value of the objective function encountered during the optimization up to iteration t, that is $\mathbf{x}_{\text{best}}(t) \in \arg\min_{i=1:t} f(\mathbf{x}_i)$. Mean and standard error are computed over 20 experiments with different random initialization. The initialization of the starting points is drawn uniformly at random in the interval $[0, 1]^D$. All optimization experiments start with a budget of 10 data points. The true measurement noise variance to be learned as a parameter in our experiments is set to the value of 10^{-4}.

3.5.1 Sensitivity analysis

The use of a quantile Gaussian process introduces the quantile parameter τ as an additional hyper-parameter. This value models the proportion of observations that are modeled by the quantile Gaussian process and, consequently, the shape of the response surface from posterior predictions. Here, we evaluate the sensitivity of Bayesian optimization to different choices of the quantile parameter τ.

We restrict our selection of quantiles to a maximum of $\tau = 0.5$ since we require our model to be sensible with respect to low observations. Intuitively, in a minimization setting, proportions of the data below the median represent good indicators of the location of a minimum and we therefore expect performances to deteriorate for $\tau > 0.5$ (a symmetric argument applies for maximization problems).

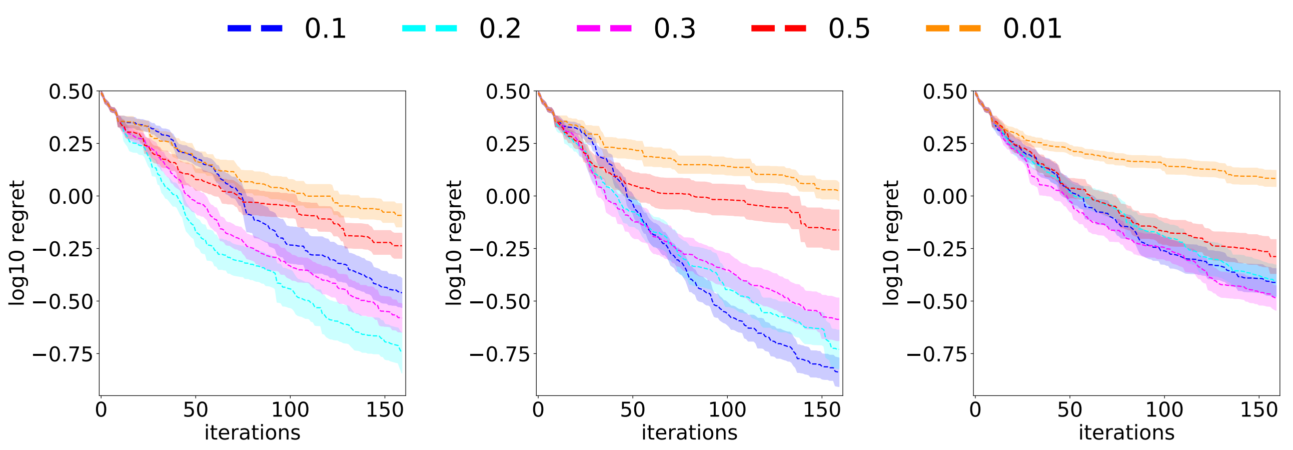
We use the Hartmann benchmark function, which is defined as

$$f(\mathbf{x}) = -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{6} \mathbf{A}_{i,j}(x_j - \mathbf{P}_{i,j})^2\right), \tag{3.25}$$

where $\alpha = [1.0, 1.2, 3.0, 3.2]^T$ and $\mathbf{A},\ \mathbf{P} \in \mathbb{R}^{4\times 6}$ are fixed given matrices. The Hartman benchmark objective function has a total of six effective dimensions. We lift the dimensionality of this objective function into a high-dimensional input space of dimensionality $D = 60$. Relevant dimensions are distributed uniformly at random over the 60 dimensions, and care is taken to ensure that all relevant dimensions are not contained in the same group. In fact, if all the relevant dimensions were contained in the same group we would only obtain standard Bayesian optimization in that group. With our approach, we avoid this scenario by imposing that the relevant dimensions are distributed across different groups and therefore different projections.

Figure 3.3 shows the progression of the immediate regret collected during the independent optimization runs. Error bars represent the standard error over 20 runs from different initializations. We evaluate performances in terms of best observed value at termination of the algorithm and *data efficiency* of each baseline which denotes the steepness of the descent in the succession towards the optimum. The collected results show good performances for moderate values of the quantile such as $0.1, 0.2, 0.3$ while extreme values such as 0.5 and 0.01 retain a much slower descent.

We observe that the extremely small quantile tends to overfit to lowest observations and reduces generalization capabilities. This renders exploration of the Bayesian optimization algorithm



(a) EI (b) UCB (c) PI

Figure 3.3: We show results on a set of quantile choices, $\tau = [0.01, 0.1, 0.2, 0.3, 0.4, 0.5]$, and compare performances across different acquisition functions: 3.3(a) *EI*, 3.3(b) *UCB*, 3.3(c) *PI*. Performances for small values, $\tau = [0.1, 0.2, 0.3]$, lead to similar convergence results both in terms of data efficiency and in final optimal guess. Error bars show the standard error over a set of 20 independent restarts.

expensive in the number of function evaluations and increases the number of local optima of the optimization landscape. Large quantiles also correspond to poor performances. Selecting $\tau = 0.5$, the quantile GP models the median which is sensible with respect to mid-range outputs. The lowest observations are treated as outliers and the resulting response surface landscape fails to capture downhill slopes relevant for global minimization.

In our set of experiments, modeling the 0.1 proportion of observations proves effective, and we identify the choice of $\tau = 0.1$ as our best configuration for the remaining experiments on synthetic data. In the subsequent we also introduce a Gamma hyper-prior for the lengthscales *Ga(shape*$=1$*, scale*$=1$*)* in each baseline to enforce exploration during optimization.

3.5.2 Additive high-dimensional objectives

In our second experiment, we assess the scalability of Bayesian optimization with quantile Gaussian processes by comparing to a set of baselines for high-dimensional optimization. We include *random embeddings* (REMBO) [WZH$^+$13], *random search* (RS) [BB12], *additive models* (Add-GP) [KSP15] and *Lipschitz continuous optimization with Gaussian processes* (GP-Lip).

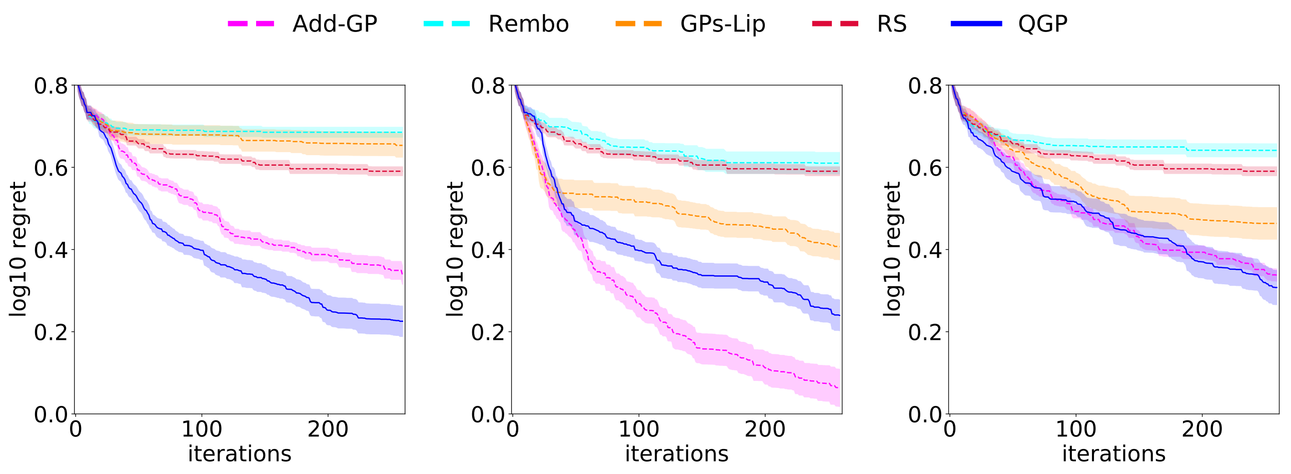
The algorithm REMBO: Random EMbeddings Bayesian Optimization [WZH$^+$13] performs standard Bayesian optimization in a low-dimensional box constrained space (embedding), i.e. $\mathbf{z} \in [z_{\min}, z_{\max}]^d$ and then projects the selected location to the original input space via a linear mapping $\mathbf{x} = \mathbf{A}\mathbf{z}$. The matrix $\mathbf{A}$ has entries sampled from standard Gaussian, i.e. $[\mathbf{A}]_{i,j} \sim \mathcal{N}(0, 1)$ and the values $z_{\min}, z_{\max}$ are provided such that there is a probability that the optimum selected in the embedded space is also an optimum in the high dimensional ambient space

Random Search [BB12] simply selects a set of T_{end} locations $\{\mathbf{x}_i\}_{i=0:T_{\text{end}}-1}$ uniformly at random in the high-dimensional input space and evaluates the objective functions on these locations without any adaptive search strategy.

Add-GP [KSP15] learns a d-dimensional Gaussian process for each addend of the sum $f_1, ..., f_z$. Each additive function component f_i for $i = 1, ..., z$ is defined on a subset of dimensions of the input space with at most d dimensions. Each component then independently optimizes an acquisition function and updates the corresponding set of d-coordinates.

The last baseline GP-Lip manually applies axis-aligned projections in a partition of the input space and resolves the inconsistencies by selecting lowest observations in pairs of points that violate Lipschitz continuity assumption. It then learns plain GP response surface (instead of a quantile Gaussian process) in each axis-aligned projection. This baseline compares with the automatic selection applied implicitly by the QGP in the presence of inconsistencies. The Lipschitz constant is the maximum element of a set of $5 \cdot 10^6$ gradients evaluated on a random selection of input locations for each benchmark function.

As an objective function for this set of experiments with additivity assumption satisfied we choose the Michalewicz function. We select this benchmark function because its valleys are aligned with the axis and therefore it is suitable for optimization algorithms based on axis-aligned projections [SSW+15]. This benchmark function has effective dimensionality $d_{ef} = 10$ and satisfies the additivity assumption. It is a sum of one-dimensional components f_i, each of which



(a) EI (b) UCB (c) PI

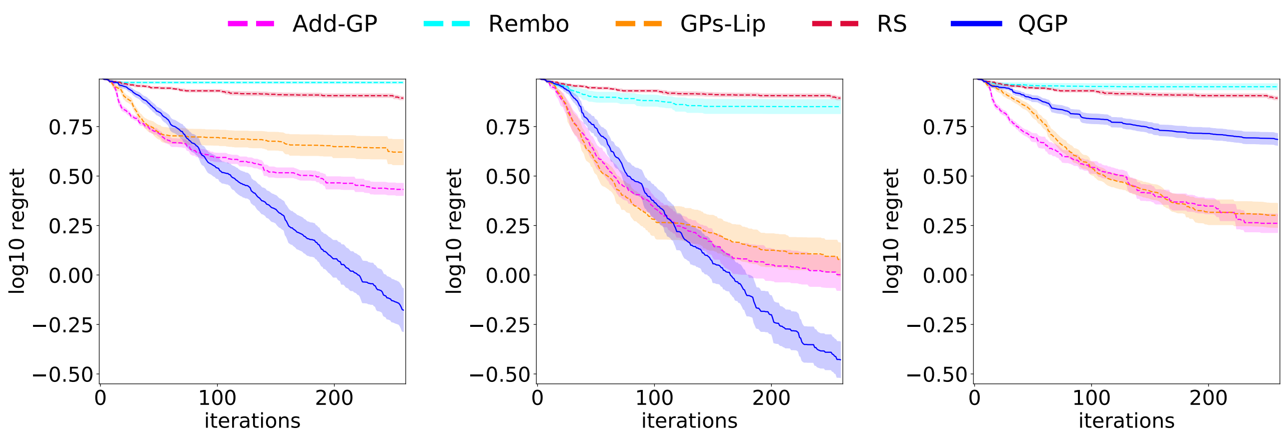
Figure 3.4: Results with Michalewicz benchmark function showing convergence results under additivity assumption with error bars showing once the standard error. The figures show comparison across different acquisition functions: 3.4(a) *EI*, 3.4(b) *UCB*, 3.4(c) *PI*. Convergence results of QGP recover best results after Add-GP which complies with Michalewicz additive properties.

is defined as $f_i(x_i) = -\sin(x_i)\sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$, that is

$$f(\mathbf{x}) = -\sum_{i=1}^{d_{ef}} f_i(x_i) = -\sum_{i=1}^{d_{ef}} \sin(x_i)\sin^{2m}\left(\frac{ix_i^2}{\pi}\right) \tag{3.26}$$

with parameter $m = 0.5$ for $i = 1, ..., d_{ef}$. To assess scalability to high dimensions, we test the optimization in a $D = 100$-dimensional input space and optimize components of dimensionality $d = 10$, where the relevant dimensions are distributed uniformly randomly (with replacement) across the 10 components of the partition by enforcing that all the relevant dimensions are not contained in a single component. In this experiment we emphasize that Add-GP conforms to the properties of the objective function and is therefore a reference baseline for good performances.

Figure 3.4 shows the progression of all optimization algorithms. Overall, we see that optimization with axis-aligned projections with the QGP model is an effective and competitive method for Bayesian optimization when f is decomposable as sum of low-dimensional components. We also note that both QGP and Lip-GP show consistent gap in data efficiency and optimization, which motivates the quantile Gaussian process as a model in the presence of inconsistencies for effective optimization along projections.



(a) EI (b) UCB (c) PI

Figure 3.5: We compare results on an effectively 10-dimensional, non-additive, objective with all acquisition functions: 3.5(a) *EI*, 3.5(b) *UCB*, 3.5(c) *PI*. We show performances with functions that are effectively lower-dimensional and non additive and assess performances of the QGP model.

3.5.3 Non-additive high-dimensional objective

This experiment analyzes the performance of the quantile Gaussian process Bayesian optimization in a high-dimensional search space, when we no longer make any assumptions on additive decomposability of the objective. More specifically, we define the objective,

$$f(\mathbf{x}) = 10\sin(x_1)\prod_{i=1}^{d_{ef}}\sin(x_i). \tag{3.27}$$

Again, we select the effective dimensionality $d_{ef} = 10$. We optimize 10-dimensional components in a fixed partition of a 100-dimensional input space avoiding condensing all relevant dimensions in a single group.

Figure 3.5 shows that the quantile Gaussian process model attains the best observation at termination with respect to other baselines in both EI and UCB acquisition functions. For PI the QGP model shows a slower progression than Add-GP and GP-Lip during the early stages of optimization. Other baselines, such as REMBO, flatten out quite early. We explain poor performances of the REMBO baseline by noting that it performs exploration only on a d-dimensional space. Using a linear mapping it can only span at most d directions in the D-dimensional space, and this heavily restricts exploration. We observe that even relaxing assumptions on additivity, the additive model still maintains similar performances both in terms of progress and value at termination for most acquisition functions considered remaining however suboptimal on exploration with expected improvement and upper confidence bound. Overall the QGP results are competitive for different properties of the black-box function and prove robust with respect to model hyper-parameter τ. These results highlight the QGP as a good model for optimization with projected data.

3.5.4 Rotated high-dimensional objective

Our last experiment analyses the performances of the QGP-Bayesian optimization approach under arbitrary rotation of the high-dimensional domain. In particular, we consider the rotated additive Michalewicz benchmark function, $g(\mathbf{x}) = f(\mathbf{U}\mathbf{x})$, where $\mathbf{U} \in \mathbb{R}^{D\times D}$ is an arbitrary orthogonal matrix, obtained by applying orthonormalization of a random matrix $\mathbf{B}$, where $[\mathbf{B}]_{i,j} \sim \mathcal{N}(0,1)$, and f is defined as in Section 3.5.2. We maintain the same model selection procedure for the GP models in each baseline, i.e. with Gamma hyper-prior *Ga(shape=*1*, scale=*1*)* on the kernel lengthscales and the marginal likelihood defined as in equation (3.5).

Figure 3.6 shows the results obtained with each baseline. Performances of both axis-aligned projections-based baselines, namely QGP model and Add-GP, clearly deteriorate w.r.t. the original experiment in Section 3.5.2. The random projection-based baseline REMBO, instead, shows steeper descent and better optimum at termination of optimization. Moreover, lengthscale kernel-parameters for the QGP model become shorter. By averaging over the 20 random initializations and the 250 iterations, we observe that at least[2] 95% of lengthscales have decreased from the axis-aligned experiment. The variance (over random initializations and iterations) of the lengthscale parameters also becomes smaller. In particular, the average variance (averaging over all lengthscales) for the rotated objective experiment decreases by a factor[3] of 0.014. This is a sign of a more highly nonlinear response surface, characterized by many local minima and therefore harder to optimize. The QGP model, however, still retains

[2]This is the minimum percentage with respect to the different acquisition functions: we observe 97% with UCB, 96% with EI and 95% with PI acquisition.

[3]0.014 with PI, 0.003 with EI and 0.006 with UCB.

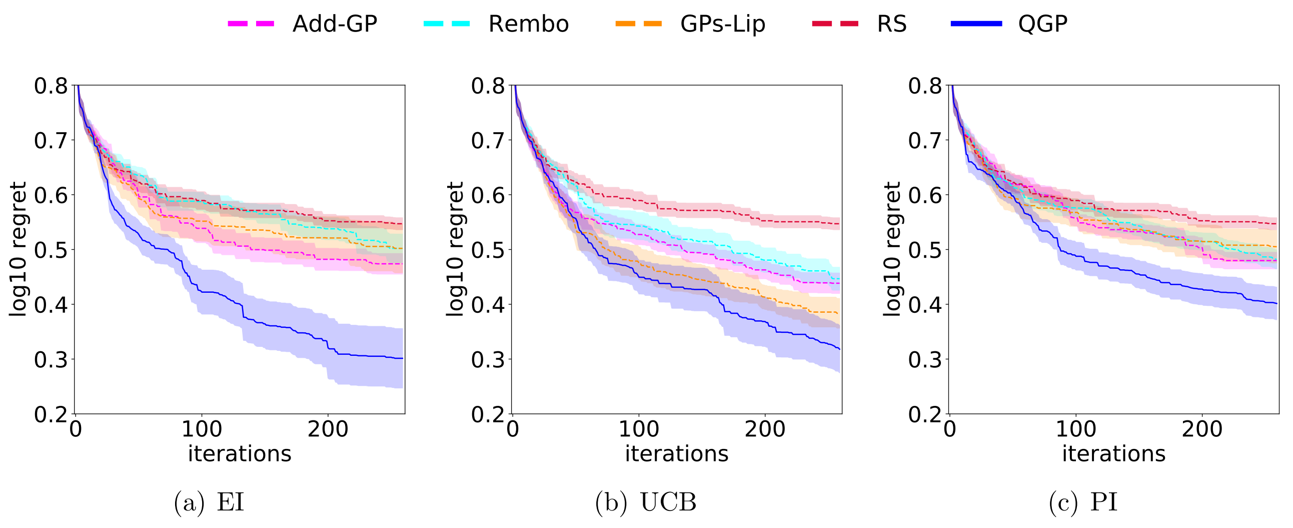


Figure 3.6: Results with Michalewicz benchmark function showing convergence results under arbitrary rotation of the high-dimensional space. We report a comparison across different acquisition functions: 3.6(a) *EI*, 3.6(b) *UCB*, 3.6(c) *PI*. Convergence results of QGP recover steepest descent and better optimum at termination on all acquisitions.

best performances also w.r.t. random projection-based methods on all acquisitions.

3.6 Summary

We proposed a framework for scaling Bayesian optimization to high dimensions by using axis-aligned projections. We considered a quantile regression approach that allows for generalizations from projected data and we empirically showed low sensitivity of quantile Gaussian process-based-Bayesian optimization w.r.t. the choice of the quantile parameter τ. Based on experimental results, we argue that modeling extreme functions from projected data maintains good indicators of the optimum location.

One observation is that quantile Gaussian process-based-Bayesian optimization features sensible modeling of the response surface from unstructured data and has an effective update strategy on all. We acknowledge that careful modeling and corresponding complexity of the learning is also an important trade-off to consider. The quantile GP approximates the Gaussian process posterior with expectation propagation, which becomes computationally involved for a large number of data points.

To address this downside, future work will tackle computational efficiency with sparse GP methods and extend applicability to a large number of data points in short time. Future work will also investigate whether to concentrate Bayesian optimization updates on projections that matter and neglect those that leave the objective function unchanged. Analysis of the Gaussian process hyper-parameters could allow for introducing an on-line selection strategy based on the optimization history.

Chapter 4

High-dimensional Bayesian optimization using low-dimensional feature spaces

In this chapter, we provide a method for high dimensional Bayesian optimization in a low dimensional feature space. As we discussed in Section 2.4, attempting to globally optimize a high dimensional black box function is an intractable task. Points in the space become increasingly distant with the dimensionality D of the search space and this renders optimization inherently hard. One possible solution to this problem is to introduce further assumptions about the black box objective function. In particular we consider the assumption that the function is intrinsically low dimensional. In other words, an intrinsically low dimensional function depends on a small set of parameters that have much smaller dimensionality $d \ll D$ than the original search space $\mathcal{X}$. This will be a central assumption on the derivation of our method.

4.1 Related work

One popular way to solve the high dimensional Bayesian optimization problem is to translate it into a collection of sub-problems. These sub-problems can either be optimized independently or may be characterized by a dependency structure among them. In [MKD20] the problem is decomposed into axis-aligned projections and then a probabilistic model based on quantile regression is employed in order to i) address challenges from these type of projections and ii) achieve successful optimization results from independent optimization of each projection. In [KSP15] the axis aligned projections are composed into an additive model. In other words, the function is assumed to be decomposable into additive components $f_1, ..., f_m$ each defined on a disjoint subset of variables. This structural assumption allows optimizing the m sub-problems independently. An extension of this approach is proposed in [RSBC18] where the axis aligned projections in the additive model are not constrained to be disjoint subsets of variables any more. In this last work the projections are rather characterized by the presence of the same variables in multiple components.

High dimensional optimization problems are often characterized by lower intrinsic dimensionality. In particular, this assumption can be exploited by defining a feature mapping which maps the original D-dimensional data onto a $d \ll D$ feature space. One instance of this approach is in [WZH+13], where the feature mapping is defined as a random linear mapping in order to reduce the dimensionality of the optimization problem. Another approach that is based on linear embeddings is presented in [GOH14]. In this work, the linear dimensionality reduction is actively learned from data. Albeit these methods produce good performances in practice, they remain confined to the linear feature mapping case. Nonlinear embeddings, instead of linear ones, could achieve higher compression rates. In our work we focus on this nonlinear setting for optimization in feature space.

Variational autoencoders (VAEs) [RMW14, KW14] offer the possibility to apply nonlinear feature mapping and achieveing higher compression rates than with linear feature maps. Applying the VAEs in the context of Bayesian optimization has been proposed in [GBWD+18, GLJL14, KPHL17, GHL17]. In [GBWD+18] the idea is to define a low dimensional latent space characterized by continuous variables and learn it with a variational autoencoder. This VAE models jointly the low dimensional representation and a property predictor which describes the objective function values in the latent space. It is then possible to perform Bayesian optimization with Gaussian processes once learnt offline i) the feature mapping from the high dimensional space into the latent space and ii) the reconstruction mapping from the latent space back into the data space. Both the encoder and the decoder in [GBWD$^+$18] are modeled with deep neural networks and are trained with hundreds of thausands of chemical compound examples. A characteristic of this approach is that the amount of data necessary for learning a meaningful representation substantially exceeds small evaluation budgets that often constrain a Bayesian optimization procedure. The method proposed in [GBWD$^+$18] requires both large amounts of data and learning the model offline without the possibility to update the learnt feature space during optimization. However, in the specific application of automatic discovery of molecules, where libraries of existing compounds are readily available prior to optimization, this approach makes much sense. In our work we propose a probabilistic model that allows for a small evaluation budget in order to learn the feature mapping and the reconstruction mapping. We use a probabilistic model based on Gaussian processes which allows us to learn online during optimization and ensures superior data efficiency with respect to variational autoencoders based approaches [GBWD$^+$18, GLJL14, KPHL17, GHL17].

VAEs were used to derive a low dimensional representation of the high dimensional Bayesian optimization problem [LGDL18]. In this work, the uncertainty related to the latent space representation is propagated through the response surface via Gaussian processes latent variable models [Law05, TL10, LQC06]. One aspect with [LGDL18] that differs from our approach is that the latent space representation is not learned specifically for the regression task. In other words, in our work we learn a low dimensional representation $\mathcal{Z}$ jointly with the response surface. This allows learning feature representations that are suitable for the regression task at hand. Gradient-based methods [ATOF18] have been used to learn a lower-dimensional Riemannian manifold for optimization and sampling.

Another example where nonlinear embeddings are employed is the modeling of non-stationary objective functions. Informally, stationarity can be thought of as the function looking similar at all input locations $\mathbf{x}$ [RW06]. Therefore non-stationary functions are characterized by abrupt changes for small variations of input space such as discontinuities. In this context, a hierarchical composition of GPs, referred to as *deep GPs* [DL13, SD17, DDGL16, Dam15, HL14] is especially useful when the response surface is characterized by sudden variations or has constraints. An extensive investigation on the employment of deep GP models in Bayesian optimization is proposed in [DDGL16, HBB$^+$19]. In our work, we also exploit the idea of learning highly nonlinear functions through the composition of simpler functions [LBH15], but we focus on the deterministic dimensionality reduction and the optimization in feature space.

In this chapter, we propose a Bayesian optimization algorithm for high-dimensional optimization, which learns a nonlinear feature mapping $\boldsymbol{h}$ to reduce the dimensionality of the inputs, and a reconstruction mapping $\boldsymbol{g}$ based on Gaussian processes to evaluate the true objective function, jointly, see Figure 4.1. This allows us to optimize the acquisition function in a lower-dimensional feature space, so that the overall BO routine scales to high-dimensional problems that possess an intrinsic lower dimensionality. Finally, we formulate a constrained maximization of the acquisition function in feature space to prevent meaningless reconstructions.

4.2 Bayesian optimization in low-dimensional feature spaces

In this section, we exploit the intrinsic low-dimensionality assumption of the high dimensional black-box objective function by performing optimization on a nonlinear *feature space* $\mathcal{Z} \subset \mathbb{R}^d$, where $d \ll D$. We refer to the high dimensional search space $\mathcal{X} \subset \mathbb{R}^D$ as the *data space* and the low dimensional $\mathcal{Z}$ as the *feature space*. The idea with our approach is to compress the high-dimensional inputs $\mathbf{x}$ into a low dimensional manifold with an *encoder* $\boldsymbol{h}$ and then perform standard Bayesian optimization in this $\mathcal{Z}$ space. This implies learning the response surface with a Gaussian process model in the low dimensional space $\mathcal{Z}$ and maximizing the acquisition function to select an optimal candidate $\mathbf{z}_{t+1} \in \mathcal{Z}$ to observe the black box function at. However, the black box function cannot be evaluated at the selected point $\mathbf{z}_{t+1}$ in feature space but requires us to map it back to the original high dimensional space $\mathbf{x}_{t+1} \in \mathcal{X}$, where the objective function is defined. This is achieved by introducing a *decoder* model $\boldsymbol{g}$ that maps the selected $\mathbf{z}_{t+1}$ back into the original data space $\mathcal{X}$. Algorithm 4 summarizes the main steps of our Bayesian optimization approach in feature space. In the following, we detail the model (see Figure 4.1) for jointly learning the feature map $\boldsymbol{h}(\cdot)$, the low-dimensional response surface in feature space, and the reconstruction mapping $\boldsymbol{g}(\cdot)$.

Algorithm 4 Key steps of Bayesian optimization in feature space. The response surface learning and acquisition function maximization are performed in feature space $\mathcal{Z}$ with dimensionality $d \ll D$. The reconstruction step in line 9 allows us to run experiments with the original objective function, f.

```
1: Inputs: $\mathbf{X}_0 = \{\mathbf{x}_1, ..., \mathbf{x}_{N_0}\} \in \mathbb{R}^{N_0 \times D}$, $\mathbf{y}_0 = \{y_1, ..., y_{N_0}\} \in \mathbb{R}^{N_0}$
2: for $t = 0, 1, 2, ..., T_{\text{end}} - 1$ do
3:     Response surface learning
4:         $f = f_Z \circ \boldsymbol{h}$
           {Composition of a feature map and a low-dimensional response surface}
5:         $\mathbf{Z}_t = \boldsymbol{h}(\mathbf{X}_t)$
           {Dimensionality reduction}
6:         $p(f_Z|\mathbf{Z}_t, \mathbf{y}_t)$
           {Low-dimensional surface learning in feature space}
7:     Optimal input selection $\mathbf{x}_{t+1}$
8:         $\mathbf{z}_{t+1} = \underset{\mathbf{z}\in\mathcal{Z}}{\operatorname{argmax}}\ \alpha(\mathbf{z})$
           {Acquisition function maximization in feature space}
9:         $\mathbf{x}_{t+1} := \boldsymbol{g}(\mathbf{z}_{t+1})$
           {Reconstruction of high-dimensional input}
10:    Evaluation
11:        $y_{t+1} = f(\mathbf{x}_{t+1}) + \varepsilon$
           {Evaluation of the high-dimensional objective function with measurement noise}
12:    $\mathbf{X}_t \cup \{\mathbf{x}_{t+1}\}, \quad \mathbf{y}_t \cup \{y_{t+1}\}$
13: end for
14: Return $\mathbf{x}^* = \arg\min \mathbf{y}_t$
    {Computed minimizer of the objective function $f$}
```

4.3 Probabilistic model

In this section, we detail the components of the probabilistic model employed for performing Bayesian optimization on a low dimensional feature space. We refer to the encoder model $\boldsymbol{h}\colon \mathbb{R}^D \to \mathbb{R}^d$ as the *feature mapping* and to the decoder $\boldsymbol{g}\colon \mathbb{R}^d \to \mathbb{R}^D$ as the *reconstruction mapping*. Our idea with the feature mapping and the reconstruction mapping is to learn a nonlinear embedding from data that

- i) is useful for optimization
- ii) can be learned in a data efficient way
- iii) can be trained online during the optimization process (at each Bayesian optimization iteration t).

In our model, we compose the feature mapping jointly with the response surface. In particular, we express the true objective function $f : \mathbb{R}^D \to \mathbb{R}$ as a composition of a feature mapping $\boldsymbol{h}$ and a function $f_Z\colon \mathcal{Z} \to \mathbb{R}$ so that $f = f_Z \circ \boldsymbol{h}$ and therefore $f(\mathbf{x}) := f_Z(\boldsymbol{h}(\mathbf{x}))$. In this way we learn features $\mathbf{z} = \boldsymbol{h}(\mathbf{x})$ that are useful for the regression task of the response surface. We train this model of the response surface jointly with the reconstruction model such that the features learned from the training are useful for two distinct tasks i) learning a response surface and ii) reconstructing the original inputs from the feature space $\mathcal{Z}$ into data space $\mathcal{X}$. Figure 4.1 shows the structure of the described probabilistic model.

This structure of the probabilistic model allows us to achieve higher compression rates with respect to linear encoder-decoder structured models. In particular, it allows us to learn the response surface and maximize the acquisition function in a feature space $\mathcal{Z}$ that is feasible for Bayesian optimization. Both the response surface model and the reconstruction model are based on Gaussian processes and therefore achieve superior data efficiency with respect to variational autoencoders, which instead require much more data for learning an embedding. Moreover GP models can be trained online at each Bayesian optimization iteration by maximizing the log

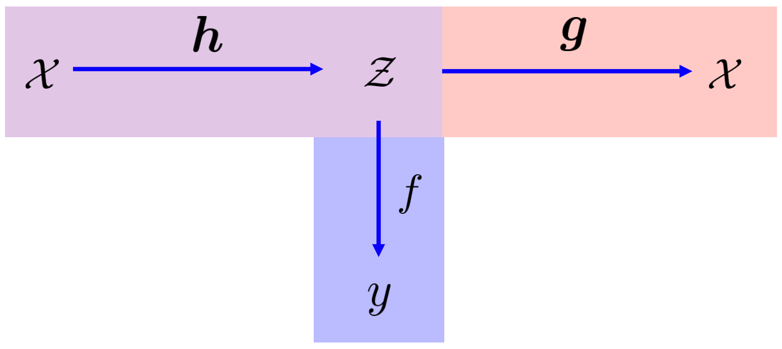

$$\begin{tikzcd} \mathcal{X} \arrow[r, "\boldsymbol{h}"] & \mathcal{Z} \arrow[r, "\boldsymbol{g}"] \arrow[d, "f"] & \mathcal{X} \\ & y & \end{tikzcd}$$

Figure 4.1: Structure of the model for the optimization in feature space. The feature space $\mathcal{Z}$ is learned from data and is useful for two different tasks: i) the regression of the response surface and ii) the reconstruction into the original data space $\mathcal{X}$.

marginal likelihood as shown in Section 2.2 in Chapter 2. Figure 4.1 describes the structure of the probabilistic model adopted in this chapter.

4.3.1 Manifold Gaussian process for response surface learning in feature space

In our Bayesian optimizatin in feature space, we require the response surface to predict the value of the black box objective function f with uncertainty associated to each prediction. We use GPs for modeling the response surface as also introduced in Section 2.2 of Chapter 2. In our optimization in feature space, we aim at learning both the feature mapping $\boldsymbol{h}$ and the low dimensional response surface f_Z jointly. These two components are described in lines 5–6 of Algorithm 4 and are part of a single learning problem. Therefore, we need a Gaussian process that accomplishes two tasks at once: i) learning useful representations $\mathbf{z}$ of inputs $\mathbf{x}$ for the regression task at hand and ii) learning the response surface f_Z in feature space. A manifold Gaussian process (MGP) [CPRD16, WHSX16] addresses these tasks by composing two mappings: the deterministic feature mapping $\boldsymbol{h}$, characterized by parameters $\boldsymbol{\theta}_h$ and a GP $f_Z \sim \mathcal{GP}(m, k)$ with kernel hyper-parameters and noise parameters $\boldsymbol{\theta}_k = \{\sigma_f, l_1, ..., l_d, \sigma_n\}$ (signal variance, characteristic length-scales and measurement noise variance, respectively). The GP models the relationship between feature vectors $\mathbf{z}$ and noisy function evaluations $y \in \mathbb{R}$ in observation space. The overall composition $f = f_Z \circ \boldsymbol{h}$ of the feature mapping and the GP in feature space is still a Gaussian process (because $\boldsymbol{h}$ is a deterministic mapping) so that

$$f \sim \mathcal{GP}(m_M, k_M) \tag{4.1}$$

$$m_M(\mathbf{x}) = m(\boldsymbol{h}(\mathbf{x})) \tag{4.2}$$

$$k_M(\mathbf{x}, \mathbf{x}') = k(\boldsymbol{h}(\mathbf{x}), \boldsymbol{h}(\mathbf{x}')) \tag{4.3}$$

where the function $m_M(\mathbf{x})$ denotes the mean function and $k_M(\mathbf{x}, \mathbf{x}')$ denotes the covariance function of the manifold GP. Given high-dimensional training inputs $\mathbf{X}_t$ and corresponding observations $\mathbf{y}_t$ of the objective function, we find model parameters $\{\boldsymbol{\theta}_h, \boldsymbol{\theta}_k\}$ that maximize the marginal likelihood (evidence) $\{\boldsymbol{\theta}_h^*, \boldsymbol{\theta}_k^*\} \in \underset{\boldsymbol{\theta}_h, \boldsymbol{\theta}_k}{\arg\max}\, p(\mathbf{y}_t|\mathbf{X}_t, \boldsymbol{\theta}_h, \boldsymbol{\theta}_k)$. By maximizing this objective function for training the manifold GP, we learn a low dimensional embedding as a by-product of the supervised regression framework.

Unsupervised methods for dimensionality reduction such as PCA [Pea01, Jol03] or variational autoencoders [RMW14, KW14] achieve high compression rates but solve an orthogonal task to that of learning a response surface. This is because the objective function being optimized for training these methods differs from the one used in the regression framework. As a result, the representation learned from unsupervised methods may not be optimal for supervised regression task [WSD15]. We highlight that in both PCA and variational autoencoder techniques the objective is solely to compress data and not learning a function in a low-dimensional representation of the input space. This renders PCA and variational autoencoders not suitable for learning a function in a feature space. With the manifold Gaussian process, instead, the feature space learned from the supervised regression framework is optimal (locally), in a marginal likelihood sense, for the regression task at hand. In other words, the manifold Gaussian process is able to learn a function (the response surface) jointly with a low-dimensional representation of its inputs (feature space). The dimensionality reduction, with the manifold Gaussian process, is a by-product of the regression framework.

We consider a multi-layer feed-forward neural network with sigmoid activation function as the feature mapping $\boldsymbol{h}$. This choice of architecture allows us to learn a nonlinear embedding of the original dataset $\mathcal{X}$. Because of the sigmoid activation function, the resulting feature space is characterized by the set $\mathcal{Z} = [0,1]^d$. Neural networks as an explicit feature map within a manifold Gaussian process have already been applied successfully for modeling non-smooth responses in robot locomotion [CPRD16, CCTM15]. Deep networks have also proven successful for learning the orientation of pictures from high-dimensional images [WHSX16]. With a Gaussian likelihood, the MGP posterior predictive distribution at a test point $\mathbf{x}_\star \in \mathcal{X}$ is Gaussian distributed with mean and variance given by

$$\begin{aligned}\mathbb{E}[f(\mathbf{x}_\star)] &= m_M(\mathbf{x}_\star) + k_M(\mathbf{x}_\star, \mathbf{X}_t)\mathbf{K}_{My}^{-1}(\mathbf{y}_t - m_M(\mathbf{X}_t)) \\ &= m(\mathbf{z}_\star) + k(\mathbf{z}_\star, \mathbf{Z}_t)\mathbf{K}_{Zy}^{-1}(\mathbf{y}_t - m(\mathbf{Z}_t))\end{aligned} \tag{4.4}$$

$$\begin{aligned}\mathbb{V}[f(\mathbf{x}_\star)] &= k_M(\mathbf{x}_\star, \mathbf{x}_\star) - k_M(\mathbf{x}_\star, \mathbf{X}_t)\mathbf{K}_{My}^{-1}k_M(\mathbf{X}_t, \mathbf{x}_\star) \\ &= k(\mathbf{z}_\star, \mathbf{z}_\star) - k(\mathbf{z}_\star, \mathbf{Z}_t)\mathbf{K}_{Zy}^{-1}k(\mathbf{Z}_t, \mathbf{z}_\star),\end{aligned} \tag{4.5}$$

respectively, with $\mathbf{z}_\star := \boldsymbol{h}(\mathbf{x}_\star)$ and $\mathbf{Z}_t := \boldsymbol{h}(\mathbf{X}_t)$ as also described in line 5 of Algorithm 4. Here, the training dataset is defined as follows $\mathbf{X}_t = \{\mathbf{x}_1, ..., \mathbf{x}_{N_t}\}$ and its representation in feature space is $\mathbf{Z}_t = \{\mathbf{z}_1 = \boldsymbol{h}(\mathbf{x}_1), ..., \mathbf{z}_{N_t} = \boldsymbol{h}(\mathbf{x}_{N_t})\}$, where N_t is the size of the datasets at the current Bayesian optimization iteration t. Moreover, $k_M(\mathbf{x}_\star, \mathbf{X}_t) = k(\mathbf{z}_\star, \mathbf{Z}_t) = [k(\mathbf{z}_\star, \mathbf{z}_i)]_{i=1}^{N_t}$ is a row vector of cross covariance terms. The inverse terms are defined as $\mathbf{K}_{My} := k_M(\mathbf{X}_t, \mathbf{X}_t) + \sigma_n^2\mathbf{I}$ and $\mathbf{K}_{Zy} := k(\mathbf{Z}_t, \mathbf{Z}_t) + \sigma_n^2\mathbf{I}$. By definition it holds that $k_M(\mathbf{X}_t, \mathbf{X}_t) = k(\mathbf{Z}_t, \mathbf{Z}_t)$, and $m_M(\mathbf{X}_t) = m(\mathbf{Z}_t)$, where $m(\mathbf{Z}_t) = [m(\mathbf{z}_i)]_{i=1}^{N_t}$ computes the prior mean function evaluated at the embedded training inputs $\mathbf{Z}_t$. In the remainder of this chapter we will assume the mean function to be the zero function that is $m(\mathbf{z}) = 0$ for all $\mathbf{z} \in \mathcal{Z}$. Given the equations (4.4)–(4.5) we are allowed to compute the posterior predictions both from inputs in data space $\mathbf{x} \in \mathcal{X}$ and from inputs in feature space $\mathbf{z} \in \mathcal{Z}$. Equations (4.4)–(4.5) redefine the posterior Gaussian process predictions previously defined in Section 2.2 of Chapter 2 by equations (2.8)–(2.9). Specifically, we define the posterior mean as $\mu(\mathbf{x}) := \mathbb{E}[f(\mathbf{x})]$ and the posterior variance as $\sigma^2(\mathbf{x}) := \mathbb{V}[f(\mathbf{x})]$.

One advantage of the manifold Gaussian process is that we obtain a GP in the high dimensional space $\mathcal{X}$ together with a response surface in the low dimensional feature space $\mathcal{Z}$. This low dimensional response surface allows us to perform standard Bayesian optimization in feature space instead of the original data/parameter space. In particular, we maximize the acquisition function in feature space as described in line 8 of Algorithm 4. The candidate point to be evaluated next $\mathbf{z}_{t+1}$ is in feature space and cannot be evaluated on the black-box objective function f whose domain is $\mathcal{X}$. Therefore, we need to map the maximizer of the acquisition function $\mathbf{z}_{t+1} \in \mathcal{Z}$ back to the original data/parameter space into the vector $\mathbf{x}_{t+1} \in \mathcal{X}$ as described in line 9 of Algorithm 4. In the following, we detail the probabilistic model for the reconstruction mapping $\boldsymbol{g}$ employed in our Bayesian optimization in feature space.

4.3.2 Input reconstruction with manifold multi-output Gaussian processes

In this subsection, we detail the model for the reconstruction from feature space to data space as also described in figure Figure 4.1 with the $\boldsymbol{g}$ mapping. This is a key step in our Bayesian optimization in feature space, which requires decoding the low dimensional features $\mathbf{z} \in \mathcal{Z}$ into the high dimensional data $\mathbf{x} \in \mathcal{X}$. This reconstruction step is also summarized in line 9 of Algorithm 4 and is crucial in order to evaluate the true black box objective function f at the selected new location $\mathbf{x}_{t+1}$. We model the reconstruction mapping with a vector valued function $\boldsymbol{g} = \{g_i\}_{i=1}^{D}$, which has components $g_i \colon \mathcal{Z} \to \mathcal{X}_i$ for $i = 1, ..., D$. Each component g_i maps low dimensional vectors in feature space to the i-th coordinate of the high dimensional data, i.e. $g_i(\mathbf{z}) = \tilde{x}^{(i)} \in \mathcal{X}_i$. Here, we denote the reconstructions with $\tilde{\mathbf{x}} \in \mathcal{X}$ with components $\tilde{x}^{(i)} \in \mathcal{X}_i$ taking values in each component set $\mathcal{X}_i$ for $i = 1, ..., D$. Multi-output Gaussian processes (MOGPs)[ÁRL11, ÁL11, BCS+09, WKG11, ÁL09, ORR+08, STJ05, BF05] define a prior in the space of vector valued functions and explicitly model output correlations. A MOGP is defined as

$$\boldsymbol{g} \sim \mathcal{GP}(\mathbf{m}, \mathbf{K}) \tag{4.6}$$

It is fully specified my a mean vector valued function $\mathbf{m} \colon \mathcal{Z} \to \mathbb{R}^D$ and a positive, semi-definite matrix-valued covariance function $\mathbf{K} \colon \mathcal{Z} \to \mathbb{R}^{D \times D}$, which computes the correlation between observations in the same output coordinate and cross-correlations between the D different outputs.

The model that we adopt for the multi-output GP is the *intrinsic coregionalization model* (ICM) [Goo97, Wac13], which is characterized by a covariance matrix with Kronecker structure. This model allows trading off the number of parameters with expressiveness of the vector valued function. In the intrinsic coregionalization model the covariance function expresses the correlation between different output dimensions thus facilitating information sharing across different tasks (dimensions of the output). One successful application of the ICM has been in robotics for learning inverse dynamics [WKVC09]. The advantages in selecting the intrinsic coregionalization model for modeling the reconstruction mapping is that it requires fewer parameters than the linear model of coregionalization [ÁRL11] and is particularly suitable for exploiting the properties of the Kronecker product for efficient training of the MOGP and posterior prediction computation.

In our reconstruction model, we are interested in ensuring that the set of the reconstructions is exactly $\mathcal{X}$. In other words, if we defined the data space to be the hypercube $\mathcal{X} = [0,1]^D$, then the reconstructions from the multi-output GP will need to belong in this specific interval. Multi-output Gaussian processes do not guarantee such a reconstruction property since they map to the whole set of real numbers. In order to address this issue, we introduce a strictly monotonic output squashing function Ψ, to constrain the values returned by the MOGP to the $[0,1]^D$ hypercube. This idea was initially introduced in the context of warped Gaussian processes [SGR04]. Since the squashing function is strictly monotonic we can define a corresponding inverse transformation Ψ^{-1} that we apply to the data $\mathbf{x}$ in input to our model. The output resulting from our MOGP model for any given test point is then squashed through the transformation Ψ. However, the reconstruction provided by the MOGP for any test point $\mathbf{z}_\star$ is a probability distribution on the value of the reconstruction $\tilde{\mathbf{x}}_\star$, that is $p(\tilde{\mathbf{x}}_\star|\mathbf{X}_t, \tilde{\mathbf{X}}_t, \mathbf{z}_\star)$. In fact, the distribution is the multi-output Gaussian process prediction which provides a mean and a covariance matrix for the reconstruction $\tilde{\mathbf{x}}_\star$. Here the set $\tilde{\mathbf{X}}_t$ denotes the training set of the reconstruction model that is $\tilde{\mathbf{X}}_t = \{\tilde{\mathbf{x}}_1, ..., \tilde{\mathbf{x}}_{N_t}\}$, where $\tilde{\mathbf{x}}_i = \mathbf{x}_i$ for $i = 1, ..., N_t$. Since we require our reconstruction to be a point in data space we evaluate the expectation with respect to this distribution of the transformed outputs, i.e.

$$\mathbf{x}_{t+1} = \mathbb{E}_{\tilde{\mathbf{x}}_\star \sim p(\tilde{\mathbf{x}}_\star | \mathbf{X}_t, \tilde{\mathbf{X}}_t, \mathbf{z}_\star)}[\Psi(\tilde{\mathbf{x}}_\star) | \mathbf{X}_t, \tilde{\mathbf{X}}_t, \mathbf{z}_\star] \tag{4.7}$$

In our work, we select the Gaussian cumulative density function as our strictly monotonic squashing function $\Psi := \Phi$ for warping the outputs of our reconstruction model [SGR04]. The motivation for choosing such a squashing function is twofold: i) the inverse mapping Ψ^{-1} of the cumulative density function is a well known function and is defined as the Probit function and ii) the computation of the expectation $\mathbb{E}_p[\Psi(\tilde{\mathbf{x}}_\star) | \mathbf{X}_t, \tilde{\mathbf{X}}_t, \mathbf{z}_\star]$ with respect to the distribution $p(\tilde{\mathbf{x}}_\star | \mathbf{X}_t, \tilde{\mathbf{X}}_t, \mathbf{z}_\star)$ at test reconstructions can be derived in closed form [RW06]. The derivation of the analytical result can be summarized in the following steps

$$\mathop{\mathbb{E}}_{\tilde{x}_i}\left[\Phi\left(\frac{\tilde{x}_i - m_i}{v_i}\right)\middle| \mathbf{X}_t, \tilde{\mathbf{X}}_t, \mathbf{z}_\star\right] = \int_{-\infty}^{+\infty} \Phi\left(\frac{\tilde{x}_i - m_i}{v_i}\right) \mathcal{N}(\tilde{x}_i | \mu_i, \sigma_i^2)\ d\tilde{x}_i \tag{4.8}$$

$$\Phi(x) = \int_{-\infty}^{x} \mathcal{N}(y|\ 0, 1)\ dy \tag{4.9}$$

where m_i is a constant and v_i is a positive constant. The terms μ_i and σ_i^2 denote the i-th components of the posterior mean and posterior variance obtained from the reconstruction predictions $p(\tilde{\mathbf{x}}_\star | \mathbf{X}_t, \tilde{\mathbf{X}}_t, \mathbf{z}_\star)$ for $i = 1, ..., D$. By joining the two expressions in equations (4.8)–(4.9), we obtain the following expression, here we rewrite the expectation in shorter form by omitting the terms after the conditioning

$$\mathop{\mathbb{E}}_{p}\left[\Phi\left(\frac{\tilde{x}_i - m_i}{v_i}\right)\right] = \frac{1}{2\pi\sigma_i v_i} \int_{-\infty}^{+\infty} \int_{-\infty}^{\tilde{x}_i} \exp\left(-\frac{(y - m_i)^2}{2v_i^2} - \frac{(\tilde{x}_i - \mu_i)^2}{2\sigma_i^2}\right)\ dy\ d\tilde{x}_i \tag{4.10}$$

$$= \frac{1}{2\pi\sigma_i v_i} \int_{-\infty}^{+\infty} \int_{-\infty}^{\mu_i - m_i} \exp\left(-\frac{(z + w)^2}{2v_i^2} - \frac{w^2}{2\sigma_i^2}\right)\ dw\ dz. \tag{4.11}$$

This last integral in equation (4.11) is obtained applying the following substitution to the integral in equation (4.10) $z = y - \tilde{x}_i + \mu_i - m_i$ and $w = \tilde{x}_i - \mu_i$. The second integral in expression (4.11) can be rewritten as an incomplete integral over a joint Gaussian. Therefore, we can compute the outer integral (the one between $-\infty$ and $+\infty$) by simply marginalizing the joint Gaussian over one of its variables, namely the variable w. With this marginalization we obtain the solution as

$$\mathop{\mathbb{E}}_{p}\left[\Phi\left(\frac{\tilde{x}_i - m_i}{v_i}\right)\right] = \frac{1}{\sqrt{2\pi(v_i^2+\sigma_i^2)}}\int_{-\infty}^{\mu_i - m_i} \exp\left(-\frac{z^2}{2(v_i^2+\sigma_i^2)}\right)\, dz \tag{4.12}$$

$$= \Phi\left(\frac{\mu_i - m_i}{\sqrt{v_i^2+\sigma_i^2}}\right). \tag{4.13}$$

Equation (4.13) is again a cumulative density function of a Gaussian distribution and can be easily evaluated at the point $(\mu_i - m_i)/\sqrt{v_i^2+\sigma_i^2}$. We apply this equation to each dimension of the output that is for indices $i = 1, ..., D$.

Intrinsic coregionalization model

The intrinsic coregionalization model [Goo97, Wac13] is the result of a linear mapping applied to a set of P latent functions $u_i \colon \mathcal{Z} \to \mathbb{R}$ for $i = 1, ..., P$ that are assumed to be *sample paths*, i.e. sample functions independently drawn from the same Gaussian process prior $\mathcal{GP}(m_c, k_c)$. Here the mean function and covariance function are defined as $m_c \colon \mathcal{Z} \to \mathbb{R}$ and $k_c \colon \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$, respectively. The intrinsic coregionalization model applies two distinct transformations, one from a d-dimensional space to a P-dimensional space, and a second one that is from the P-dimensional space to a D-dimensional one. In our setting, the d-dimensional set characterizes the feature space which is our $\mathcal{Z}$ space. The P-dimensional space is the collection of P independent GP sample paths and the D-dimensional space is our data space $\mathcal{X}$. The first transformation is characterized by $\boldsymbol{u}(\mathbf{z}) = \{u_i(\mathbf{z})\}_{i=1}^{P}$. Each point in the training set in feature space is evaluated on all the GP sample paths obtaining a P-long vector of evaluations for each training input, that is $\mathbf{z}_i \in \mathbf{Z}_t \to \boldsymbol{u}(\mathbf{z}_i)$, where $\boldsymbol{u}(\mathbf{z}_i) = [u_1(\mathbf{z}_i), ..., u_P(\mathbf{z}_i)]^T$, for $i = 1, ..., N_t$. Note that $\boldsymbol{u}$ is a vector valued function. Then a linear map is applied to the result of this vector valued function $\boldsymbol{g}(\mathbf{z}) = \mathbf{A}\boldsymbol{u}(\mathbf{z})$. Here $\mathbf{A} \in \mathbb{R}^{D\times P}$ is the linear mapping that couples the independent vector and parameterizes the ICM model. As a result, $\boldsymbol{g}$ is an MOGP

$$\boldsymbol{g} \sim \mathcal{GP}(\mathbf{m}, \mathbf{K}) \tag{4.14}$$

$$\mathbf{m} = \mathbf{A}\mathbf{m}_c \tag{4.15}$$

$$\mathbf{K}(\mathbf{z}, \mathbf{z}') = \mathbf{A}\mathbf{A}^T \otimes k_c(\mathbf{z}, \mathbf{z}') \tag{4.16}$$

where equation (4.15) denotes the mean function and equation (4.16) the covariance function of the multi-output Gaussian process in equation (4.14). Here, $\mathbf{m}_c = [m_c]_{i=1}^P$ is obtained by repeating the single-valued mean function m_c in a P-vector. Moreover, k_c is the covariance function for the GP prior, $\otimes$ is the Kronecker product and the matrix $\mathbf{A}\mathbf{A}^T$ is the coregionalization matrix. Note that k_c may differ from the covariance function k used for the response surface f_Z. The kernel of the intrinsic coregionalization model is thus characterized by a set of parameters $\boldsymbol{\theta}_c = \{\sigma_{fc}^2, l_{1c}, ..., l_{dc}, \mathbf{A}, \sigma_{nc}^2\}$ which consist on signal variance σ_{fc}^2 of the kernel k_c, a set of d characteristic length-scales $l_{1c}, ..., l_{dc}$, the component $\mathbf{A}$ of the coregionalization matrix $\mathbf{A}\mathbf{A}^T$ and the noise variance parameter σ_{nc}^2.

Reconstruction Model

Here we address the reconstruction task that we defined in line 9 of Algorithm 4. For this purpose we define the manifold MOGP with intrinsic coregionalization model (mMOGP). This joint model is characterized by the same feature mapping $\boldsymbol{h}$ used for learning the response surface in feature space with the manifold GP which has been introduced in Section 4.3.1. We apply the same trick as seen in Section 4.3.1 to obtain the mMOGP's covariance function which combines the feature mapping $\boldsymbol{h}$ and the covariance function of the intrinsic coregionalization model k_c, that is $k_{MO}(\mathbf{x}, \mathbf{x}') = k_c(\boldsymbol{h}(\mathbf{x}), \boldsymbol{h}(\mathbf{x}'))$. The resulting Gaussian process is $\boldsymbol{g} \circ \boldsymbol{h} \sim \mathcal{GP}(\mathbf{0}, \mathbf{B} \otimes k_{MO})$, where the covariance function is specified by the coregionalization matrix $\mathbf{B} = \mathbf{A}\mathbf{A}^T$. Here, we have assumed without loss of generality a prior zero-mean vector function for the manifold MOGP. We can interpret this model as an auto-encoder, where the MGP plays the role of the encoder with $\boldsymbol{h} : \mathcal{X} \to \mathcal{Z}$ and the MOGP the role of the decoder, which maps low-dimensional features back into data space.

4.3.3 Joint training

The joint training of the MGP, which models the response surface, and the mMOGP, which is used for the reconstruction (see also Figure 4.1), is performed by maximizing a rescaled version of the log-marginal likelihood. We first write the joint model of the response surface and the mMOGP

$$\begin{aligned} p(\mathbf{y}, \mathbf{X}_t, f, \boldsymbol{g}) &= p(\mathbf{y}, f)\ p(\mathbf{X}_t, \boldsymbol{g}) && (4.17)\\ &= p(\mathbf{y}|f)p(f)\ p(\mathbf{X}_t|\boldsymbol{g})p(\boldsymbol{g}) && (4.18)\end{aligned}$$

Here $\mathbf{y}$ is the vector of noisy observations and f is the response surface in feature space. The variable $\mathbf{X}_t$ denotes the training data and $\boldsymbol{g}$ is the multi-output reconstruction mapping. Both probabilities $p(\mathbf{y}|f)$ and $p(\mathbf{X}_t|\boldsymbol{g})$ are Gaussian likelihoods. The probabilities $p(f)$ and $p(\boldsymbol{g})$ are the Gaussian process priors of the response surface and the multi-output reconstruction, respectively. The joint distribution on the left hand-side of equation (4.17) is then marginalized with respect to f and $\boldsymbol{g}$ to obtain the following marginal likelihood

$$\mathcal{L} \propto -\mathbf{y}^T\mathbf{K}_{Zy}^{-1}\mathbf{y} - \log|\mathbf{K}_{Zy}| - \frac{1}{D}\left(\mathbf{x}_V^T\mathbf{K}_V^{-1}\mathbf{x}_V + \log|\mathbf{K}_V|\right) + \text{const.} \qquad (4.19)$$

Here, $\mathcal{L}$ comprises terms from both the MGP and mMOGP models, where $\mathbf{K}_{Zy}$ is defined in equations (4.4)–(4.5) and denotes the covariance matrix obtained from the training inputs in $\mathcal{Z}$ space. The covariance matrix of the mMOGP $\mathbf{K}_V = \bar{\mathbf{K}} + \sigma_{nc}^2\mathbf{I}$ is obtained by evaluating the Kronecker product $\bar{\mathbf{K}} = \mathbf{B} \otimes k_c(\mathbf{Z}_t, \mathbf{Z}_t)$ with the MOGP kernel k_c evaluated at the training inputs $\mathbf{Z}_t$ in feature space. The vector $\mathbf{x}_V$ is a concatenation of the columns of the training data $\mathbf{X}_t \in \mathbb{R}^{N_t \times D}$. The maximizers $[\boldsymbol{\theta}_h^*, \boldsymbol{\theta}_k^*, \boldsymbol{\theta}_c^*]$ of the log-marginal likelihood are the parameters $\boldsymbol{\theta}_h^*$ of the feature mapping $\boldsymbol{h}$ (which is shared between the MGP and the mMOGP), the hyper-parameters $\boldsymbol{\theta}_k^*$ of the kernel k of the manifold GP and the hyper-parameters $\boldsymbol{\theta}_c^*$ of k_c including the component $\mathbf{A}$ of the coregionalization matrix $\mathbf{B}$ for the mMOGP, respectively.

We balance the contribution of the two components of the log-marginal likelihood involved in training by introducing a $1/D$ factor that multiplies the log-marginal likelihood of the manifold

MOGP (the term in brackets in equation (4.19)). The two components are the log-marginal likelihood of the manifold GP (first and second term on the right hand side of equation (4.19)) and the log-marginal likelihood of the MOGP in feature space (term in brackets in equation (4.19)). The two log-marginal likelihood terms are characterized by different dimensions of the covariance matrices involved in their computation, namely $\mathbf{K}_{Zy}$ and $\mathbf{K}_V$. In order to balance the two terms we would expect the dimensions of these two matrices to be the same. The dimensions of the matrix $\mathbf{K}_V$ are $N_tD \times N_tD$, while the dimensions of the matrix $\mathbf{K}_{Zy}$ are $N_t \times N_t$. In order to achieve the same dimensions we could repeat the $\mathbf{K}_{Zy}$ matrix D times in a block diagonal fashion. This block diagonal would then have quadratic form equal to $D\mathbf{y}^T\mathbf{K}_{Zy}^{-1}\mathbf{y}$ and log determinant equal to $D \log |\mathbf{K}_{Zy}|$. The meaning of this operation is that we model and train the response surface D-times instead of a single time. Therefore, an equivalent rescaling is to divide the reconstruction terms $\mathbf{x}_V^T\mathbf{K}_V^{-1}\mathbf{x}_V$ and $\log |\mathbf{K}_V|$ by D. Optimization of equation (4.19) can be performed via gradient-based methods [BLNZ95, ZBLN97].

Modeling the black box function f with the response surface in feature space is an orthogonal task to that of reconstructing the original inputs in data space from features $\mathbf{z}$ in feature space $\mathcal{Z}$. However, by training these tasks jointly in our log-marginal likelihood in equation (4.19), they have a regularization effect on the optimization of the parameters $\boldsymbol{\theta}_h$ of the neural network encoder in the sense that the feature mapping $\boldsymbol{h}$ will not overfit to a single regression task: the parameters $\boldsymbol{\theta}_h$ will determine a feature space embedding that is useful for both the modeling of the objective function and the reconstruction of the original inputs in data space.

The major computational bottleneck for evaluating the log-marginal likelihood in equation (4.19) comes from the term $\mathbf{x}_V^T\mathbf{K}_V^{-1}\mathbf{x}_V$. The matrix involved in this quadratic form has size $N_tD \times N_tD$ and needs to be inverted. The computational complexity of this operation is cubic in the number of rows/columns of the matrix and would therefore require $\mathcal{O}(N_t^3D^3)$ operations which easily become intractable for moderate dimensionality D of the data space and training set size N_t. However, thanks to the intrinsic coregionalization model adopted for the reconstruction task, the matrix $\mathbf{K}_V$ has a particular Kronecker structure that can be exploited for efficient training. In our work, we reduce the computational complexity of this matrix inversion to $\mathcal{O}(N_t^3) + \mathcal{O}(D^3)$ by exploiting the properties of the Kronecker product, tensor algebra [RHB99] and structured

Gaussian processes [GSC13, Saa12] as shown in the following section.

4.3.4 Computationally efficient mMOGP

In our work, the reconstruction mapping is defined as the expectation, with respect to the posterior distrbution $p(\tilde{\mathbf{x}}_\star|\mathbf{X}_t, \tilde{\mathbf{X}}_t, \mathbf{z}_\star)$ of the MOGP, of a Gaussian cumulative density function evaluated at the test reconstruction $\tilde{\mathbf{x}}_\star$, that is $\mathbb{E}_p[\Phi(\tilde{\mathbf{x}}_\star)|\mathbf{X}_t, \tilde{\mathbf{X}}_t, \mathbf{z}_\star]$. This allows us to derive the reconstruction in closed form given the test predictions of the manifold multi output GP. We also train the mMOGP with a rescaled version of the log-marginal likelihood. However, both these two operations of computing the test predictions, which are essential for the reconstruction step, and evaluating the log-marginal likelihood require the computation of a Kronecker product. This Kronecker product enables modeling the correlation between arbitrary pairs of dimensions in the intrinsic coregionalization model. In the ICM model the full covariance matrix of all outputs can be written as

$$\mathbf{K}_V = \mathbf{B} \otimes k_c(\mathbf{Z}_t, \mathbf{Z}_t) + \sigma_{nc}^2 \mathbf{I}, \tag{4.20}$$

where $k_c(\mathbf{Z}_t, \mathbf{Z}_t)$ is the covariance matrix obtained from the training inputs $\mathbf{Z}_t$ in feature space, the $\mathbf{B}$ matrix is the coregionalization matrix of the intrinsic coregionalization model and the $\sigma_{nc}^2\mathbf{I}$ term denotes the additive spherical Gaussian noise that affects the observations of the reconstructions. Inverting the full covariance matrix $\mathbf{K}_V$ in equation (4.20) scales cubically in the number of dimensions D and the number of training data points N_t, i.e. $\mathcal{O}(N_t^3 D^3)$. Therefore, this inversion operation easily becomes intractable in high dimensional spaces even for small N_t. Even the storage of the full covariance matrix in memory becomes quite challenging in high dimensions since the number of elements contained in the matrix scales quadratically with dimension D and training set size N_t, that is $\mathcal{O}(N_t^2 D^2)$. Here we propose an efficient implementation of the ICM. In particular, we exploit the properties of the Kronecker product and apply results from structured GPs [GSC13, Saa12] that allow for efficient computation of the log-marginal likelihood in equation (4.19) and efficient posterior predictions with the manifold multi output Gaussian process decoder. We reduce the complexity of inverting the matrix $\mathbf{K}_V$ to a sum of cubic terms, that is $\mathcal{O}(N_t^3) + \mathcal{O}(D^3)$. Moreover we reduce the storage requirements for our algorithm for training and posterior prediction to $\mathcal{O}(N_t D)$ memory space. We are interested in the full covariance matrix under the assumption of a Gaussian likelihood for the multi-output observations, i.e. $\bar{\mathbf{K}} + \sigma_{nc}^2 \mathbf{I}$, where $\bar{\mathbf{K}} = \mathbf{B} \otimes k_c(\mathbf{Z}_t, \mathbf{Z}_t)$. We first express the full covariance matrix in terms of its eigendecomposition, i.e. $\bar{\mathbf{K}} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$, where $\mathbf{Q}$ is an orthogonal matrix whose columns are the eigenvectors of $\bar{\mathbf{K}}$ and $\mathbf{\Lambda}$ is a diagonal term that contains the eigenvalues of $\bar{\mathbf{K}}$ in its main diagonal. This allows expressing the inverse of the covariance from noisy targets as

$$\left(\bar{\mathbf{K}} + \sigma_{nc}^2 \mathbf{I}\right)^{-1} = \mathbf{Q}\left(\mathbf{\Lambda} + \sigma_{nc}^2 \mathbf{I}\right)^{-1} \mathbf{Q}^T, \tag{4.21}$$

where both $\mathbf{\Lambda}$ and $\sigma_{nc}^2 \mathbf{I}$ are diagonal and can be trivially inverted. However, the eigendecomposition of the $N_t D \times N_t D$ matrix $\bar{\mathbf{K}}$ would still be cubic in the product between the number of dimensions D and the number of training data N_t. Here, we exploit the properties of the Kronecker product and therefore express the eigendecomposition itself as a Kronecker product, i.e.

$$\bigotimes_{l=1}^{W} \mathbf{K}_l = \bigotimes_{l=1}^{W} \mathbf{Q}_l \bigotimes_{l=1}^{W} \mathbf{\Lambda}_l \Big(\bigotimes_{l=1}^{W} \mathbf{Q}_l \Big)^T, \tag{4.22}$$

where each term of the Kronecker product on the left-hand side $\mathbf{K}_l \in \mathbb{R}^{G_l \times G_l}$ has eigendecomposition $\mathbf{K}_l = \mathbf{Q}_l \mathbf{\Lambda}_l \mathbf{Q}_l^T$ for $l = 1, ..., W$, where W is number of factors in the Kronecker product. In our intrinsic coregionalization model we have $W = 2$, because the coregionalization matrix $\mathbf{B}$ Kronecker multiplies $k_c(\mathbf{Z}_t, \mathbf{Z}_t)$, the covariance matrix of the observations; see equation (4.20). Thus, from (4.21)–(4.22), we are allowed to invert the covariance from noisy targets by separately decomposing the covariance matrix $k_c(\mathbf{Z}_t, \mathbf{Z}_t) = \mathbf{Q}_k \mathbf{\Lambda}_k \mathbf{Q}_k^T$ and the coregionalization matrix $\mathbf{B} = \mathbf{Q}_b \mathbf{\Lambda}_b \mathbf{Q}_b^T$, which require $\mathcal{O}(N_t^3)$ and $\mathcal{O}(D^3)$ time, respectively; see line 5 of Algorithm 5.

Algorithm 5 Efficient computation of the inverse for matrices that have a Kronecker structure and spherical additive noise. Subroutine `matvecmul`: fast matrix-vector multiplication for matrices that can be expressed as a Kronecker product. Here the function `eigh` returns the eigen-decomposition of a matrix.

```
1: Input matrices: $\{\mathbf{K}_l \in \mathbb{R}^{G_l \times G_l}\}_{l=1}^{W}$
2: Input vector: $\mathbf{x}_V \in \mathbb{R}^{N_V}, \quad N_V = \prod_{l=1}^{W} G_l$
3: Input variable: $\sigma_{nc}^2$
4: for $l = 1, 2, ..., W$ do
5:     $\mathbf{\Lambda}_l, \mathbf{Q}_l, = \texttt{eigh}(\mathbf{K}_l)$ {Eigen-decomposition of each input matrix}
6: end for
7: $\mathbf{s} = \texttt{matvecmul}(\bigotimes_{l=1}^{W} \mathbf{Q}_l^T, \mathbf{x}_V)$ {Fast matrix-vector multiplication}
8: $\mathbf{D} = \bigotimes_{l=1}^{W} \mathbf{\Lambda}_l + \sigma_{nc}^2 \mathbf{I}$ {Diagonal term with eigenvalues and noise}
9: $\mathbf{w} = \mathbf{D}^{-1}\mathbf{s} = [s_i / D_{i,i}]_{i=1}^{N_V}$ {Standard matrix-vector multiplication}
10: Return $\mathbf{r} = \texttt{matvecmul}\left(\bigotimes_{l=1}^{W} \mathbf{Q}_l, \mathbf{w}\right) = \left(\bigotimes_{l=1}^{W} \mathbf{K}_l + \sigma_{nc}^2 \mathbf{I}\right)^{-1} \mathbf{x}_V$
    {Fast matrix vector multiplication of an inverse with Kronecker structure and a vector}
1: Procedure $\texttt{matvecmul}(\bigotimes_{l=1}^{W} \mathbf{K}_l, \mathbf{x})$
2: Input matrices: $\{\mathbf{K}_l \in \mathbb{R}^{G_l \times G_l}\}_{l=1}^{W}$
3: Input vector: $\mathbf{x} \in \mathbb{R}^{N_V}, \quad N_V = \prod_{l=1}^{W} G_l$
4: $\mathbf{r} = \mathbf{x}$ {Initialize result}
5: for $l = W, W-1, ..., 1$ do
6:     $\mathbf{R} = \texttt{reshape}(\mathbf{r}, [G_l, N_V / G_l])$ {Reshape results}
7:     $\mathbf{Z} = \mathbf{K}_l \mathbf{R}$
       Matrix-tensor product
8:     $\mathbf{r} = \texttt{vec}(\mathbf{Z}^\top)$ {Reshape results}
9: end for
10: Return $\mathbf{r} = \left(\bigotimes_{l=1}^{W} \mathbf{K}_l\right) \mathbf{x}$
    {Fast matrix vector multiplication for matrix with Kronecker structure}
```

Now that we are able to perform efficiently the inversion of matrix $\mathbf{K}_V$ defined in equation (4.20) we address the challenge of storing this matrix in memory. In particular, storing this inverse matrix and multiplying it by a vector still requires $\mathcal{O}(N_t^2 D^2)$ space and run time, respectively. Therefore this computational step becomes the main bottleneck for efficient manifold multi-output GP training and posterior GP predictions. Ideally we wish to be able to compute the matrix multiplication of the $\mathbf{K}_V$ matrix times a vector without evaluating the Kronecker product. This can be achieved by representing the expensive matrix-vector multiplication as a sequence of small matrix-tensor multiplications [RHB99]. In particular, we are interested in efficiently evaluating

$$\mathbf{r} = \left(\bigotimes_{i=1}^{W} \mathbf{K}_i \right) \mathbf{x}. \tag{4.23}$$

We first represent the multiplication of a matrix with Kronecker structure by a vector as a tensor product. A *tensor* $\mathbf{T}_{i_1,\dots,i_V}$ can be interpreted as an extension of matrices to objects where elements are indexed using a set of V indices: $i_1, \dots, i_V$, where the number V is referred to as the *order* of the tensor (instead of usual matrix notation which only requires two indices: i, j). Given the definition of the Kronecker product, we express the left-hand side of (4.22) as a tensor

$$\left[\bigotimes_{l=1}^{W} \mathbf{K}_l \right]_{i,j} = [\mathbf{K}_1]_{i_1,j_1} \cdot \dots \cdot [\mathbf{K}_W]_{i_W,j_W} \,, \tag{4.24}$$
$$1 \le i_l, j_l \le G_l \,, \quad 1 \le i,j \le \prod_{l=1}^{W} G_l.$$

The right-hand side of (4.24) coincides with a tensor $\mathbf{T}^K_{i_1,j_1,\dots,i_W,j_W}$, and a similar tensor-representation can be obtained for the $\prod_{l=1}^{W} G_l$-long vector $\mathbf{x}$, i.e. $\mathbf{T}^X_{j_W,\dots,j_1}$. A tensor product between the tensors $\mathbf{T}^K_{i_1,j_1,\dots,i_W,j_W}$ and $\mathbf{T}^X_{j_W,\dots,j_1}$ applies a contraction along the indices of the second tensor, i.e.

$$\sum_{j_1} \dots \sum_{j_W} \mathbf{T}^K_{i_1,j_1,\dots,i_W,j_W} \mathbf{T}^X_{j_W,\dots,j_1}. \tag{4.25}$$

This tensor contraction can be expressed in terms of a sequence of tensor-transposed matrix-tensor products

$$\Big(\bigotimes_{l=1}^{W} \mathbf{K}_l\Big)\mathbf{x} = \operatorname{vec}\Big(\Big(\mathbf{K}_1 \cdots \big(\mathbf{K}_W \mathbf{T}^X\big)^\top\Big)^\top\Big) \tag{4.26}$$

$$\mathbf{K}_l \mathbf{T}^X = \sum_{k=1}^{G_l} [\mathbf{K}_l]_{i_1,k}\, \mathbf{T}^X_{k,j_2,\ldots,j_W}. \tag{4.27}$$

Here, the function vec(·) returns the vectorized form of a matrix by stacking its columns vertically. The tensor transposition $\top$ applies a cyclic permutation to the order of the indices in a tensor. As a result, the right-hand side in (4.26) allows us to evaluate the expensive matrix-vector product without computing and storing the Kronecker product. Algorithm 5 shows the main steps of the efficient matrix inversion and matrix-vector multiplication for matrices that are characterized by a Kronecker structure. The matrix vector multiplication subroutine is expressed as a sequence of tensor-transpose matrix-tensor products.

4.4 Constrained acquisition function maximization

We have addressed the probabilistic modeling for our Bayesian optimization using low dimensional feature spaces. In particular we have defined a joint probabilistic model for both the response surface learning and the input reconstruction task in data space. The steps covered so far coincide with lines 4–6 and 9 of Algorithm 4, respectively. Now we take care of the maximization of the acquisition function in a low-dimensional feature space $\mathcal{Z}$ of the original data/parameter space $\mathcal{X}$.

One problem that arises at this stage with the mMOGP decoder is that locations $\mathbf{z}_{t+1}$ in feature space, which are too far away from data, will be mapped back to the mMOGP prior in data space. Under a standard zero-mean assumption, the high-dimensional reconstruction of these locations will be almost zero for all coordinates, $\mathbf{g}(\mathbf{z}_{t+1}) \approx \mathbf{0}$. This implies that the reconstructions through the expectation of the monotonic squashing function in equation (4.7) coincide with $\mathbf{x}_{t+1} = \mathbf{1}_D 0.5$ where $\mathbf{1}_D$ is a D-dimensional vector of all ones. The acquisition function is a key driver for exploration in Bayesian optimization. This means that when attempting exploration in feature space the reconstructions will collapse to a single point in data/parameter space. This characteristic of the algorithm is problematic and we address this limitation by restricting the search space of the acquisition function maximization in feature space. In particular, we introduce a constraint based on the Lipschitz continuity of the manifold multi output Gaussian process posterior. This will ensure that candidates $\mathbf{z}_{t+1} \in \mathcal{Z}$ selected in feature space will not collapse to a single point.

We want to leverage the information from observed data for the multi-output Gaussian process mapping and exploit it when optimizing the acquisition function in feature space. Leveraging information from observed data implies bounding the distance in feature space between the optimization variable and the training data in feature space. This can be achieved by upper bounding the Euclidean distance

$$\mathrm{dist}(\mathbf{z}, \mathbf{Z}_t) = \min_{1 \le i \le N_t} \|\mathbf{z}_i - \mathbf{z}\|_2 \tag{4.28}$$

in feature space between the optimization variable $\mathbf{z}$ and the embedded training data $\mathbf{Z}_t = \{\mathbf{z}_1, ..., \mathbf{z}_{N_t}\}$. Recall that here N_t is the number of data points available at Bayesian optimization iteration t. The desired upper bound is obtained by exploiting the Lipschitz continuity property of the multi-output posterior mean for which

$$|[\boldsymbol{\mu}(\mathbf{z})]_i - [\boldsymbol{\mu}(\mathbf{z}')]_i| \le L\|\mathbf{z} - \mathbf{z}'\|. \tag{4.29}$$

Here, L denotes the Lipschitz constant of the posterior mean $\boldsymbol{\mu}$ of the manifold multi-output Gaussian process. For common kernels, such as Matérn$_{52}$ and squared exponential, the posterior mean is Lipschitz continuous. Now we define the upper bound that needs to be satisfied in our maximization of the acquisition function

$$\mathrm{dist}(\mathbf{z}, \mathbf{Z}_t) \le \frac{\mu_{\max}(\mathbf{z}^*)}{L}. \tag{4.30}$$

This constraint allows us to specify how far from the data we can move in feature space without falling back to the prior on all coordinates of the reconstruction. Here $\mathbf{z}^*$ is the closest point in the training set $\mathbf{Z}_t$ to the decision variable $\mathbf{z}$. One can see $\mathbf{z}^*$ as the minimizer of the Euclidean distance expressed in equation (4.28). The numerator $\mu_{\max}(\mathbf{z}^*)$ on the right hand side of equation (4.30) is the maximum component-wise of the posterior mean of the mMOGP evaluated at $\mathbf{z}^*$, i.e. the maximum component-wise in $\boldsymbol{\mu}(\mathbf{z}^*)$.

We estimate the Lipschitz constant as the maximum norm of the Jacobian of the posterior mean of the manifold multi-output Gaussian process [GDHL16]

$$L = \max_{\mathbf{z}\in\mathcal{Z}} \|\nabla_{\mathbf{z}}\boldsymbol{\mu}(\mathbf{z})\|. \tag{4.31}$$

This maximization problem returns as a solution a valid Lipschitz constant [GDHL16] for the multi output posterior mean for any choice of norm in equation (4.31). The Jacobian of the posterior mean is represented by a $D \times d$ matrix and we adopt the max norm $\|\nabla_{\mathbf{z}}\boldsymbol{\mu}(\mathbf{z})\|_{\max} = \max|\mu^{'}_{i,j}|$ for $i = 1, ..., D$ and $j = 1, ..., d$. This specific choice of matrix norm is motivated by the equivalence relationship with other example matrix norms, such as 2-norm $\|\cdot\|_2$, Frobenius $\|\cdot\|_F$ and trace norm $\|\cdot\|_*$ for which it holds that [GVL96, HJ12], for any given value of the matrix $\nabla_{\mathbf{z}}\boldsymbol{\mu}(\mathbf{z})$

$$\|\nabla_{\mathbf{z}}\boldsymbol{\mu}(\mathbf{z})\|_{\max} \leq \|\nabla_{\mathbf{z}}\boldsymbol{\mu}(\mathbf{z})\|_2 \leq \|\nabla_{\mathbf{z}}\boldsymbol{\mu}(\mathbf{z})\|_F \leq \|\nabla_{\mathbf{z}}\boldsymbol{\mu}(\mathbf{z})\|_*. \tag{4.32}$$

Lower values of valid Lipschitz constants L allow for exploration in larger regions of the feature space that still satisfy the nonlinear constraint in equation (4.30).

4.5 Experiments

In this section, we report the results obtained on i) a set of high dimensional benchmark functions and ii) a real world application. Here, the objective functions to optimize in both these categories of problems are characterized by intrinsic low dimensionality. In particular, we are interested in i) assessing the benefits of adopting a probabilistic model structure as presented in Figure 4.1; ii) analyzing the benefits of applying a constrained maximization of the acquisition function with the constraint defined in equation (4.30). Our purpose for this section is to compare empirical performances across i) different characterization of the feature spaces, for instance linear subspaces versus nonlinear subspaces; ii) different properties of the objective function to be optimized such as additivity or non-additivity; iii) a real world optimization problem.

Approaches

In our experimental evaluation we consider our approach (MGPC-BO) [MDK20] and compare it to a set of different baselines which include random embeddings Bayesian optimization (REMBO) [WZH+13], additive models Bayesian optimization (ADD-BO) [KSP15], one recently proposed VAE-based model (VAE-BO) [GBWD+18] and the quantile Gaussian process baseline (QGP-BO) [MKD20] proposed in Chapter 3. The REMBO baseline performs Bayesian optimization on a linear subspace of the inputs in data space. ADD-BO assumes an additive structure (across dimensions) of the objective function f and therefore decomposes the problem into additive components and optimizes each component independently. The baseline VAE-BO learns a feature space for Bayesian optimization with deep networks offline. Finally, the QGP-BO baseline decomposes the problems into independent components each characterized by a QGP response surface as described in Chapter 3. We also include a version of our model presented in Figure 4.1 (HMGPC-BO) that uses a hierarchical ICM for the input reconstruction mapping $\boldsymbol{g}$. The hierarchical intrinsic coregionalization model partitions the data space into low-dimensional disjoint subsets, i.e. $\{\mathcal{X}_1, ..., \mathcal{X}_Q\}$, with $\mathcal{X}_i \subset \mathbb{R}^3$, and assumes statistical independence between reconstructions of different subsets, i.e. $\tilde{\mathbf{x}}^{(i)} \perp \tilde{\mathbf{x}}^{(j)}$, where $\tilde{\mathbf{x}}^{(i)} \in \mathcal{X}_i$, $\tilde{\mathbf{x}}^{(j)} \in \mathcal{X}_j$ for $i \neq j$. To each subset in the partition is applied a different ICM with different kernel hyper-parameters that is different kernels k_c^i and different coregionalization matrices $\mathbf{B}^i$ for $i = 1, ..., Q$.

We also compare our baselines MGPC-BO and HMGPC-BO with a version of the algorithm that features the same probabilistic model structure but without the nonlinear Lipschitz constraint in equation (4.30). This will allow us to assess the beneficial effect of the nonlinear constraint introduced in Section 4.4. We therefore compare our baselines MGPC-BO and HMGPC-BO with their constraint free version MGP-BO and HMGP-BO, respectively. Moreover, we also compare our approach with a different parametrization of the covariance function of the decoder $\boldsymbol{g}$. In particular, the baselines DMGP-BO and DMGPC-BO assume a single kernel k_c for the reconstruction mapping while HMGP-BO and HMGPC-BO assume different kernels $\{k_c^1, ..., k_c^Q\}$, one for each subset of the partitioning of the data space. Here, DMGPC-BO and DMGP-BO denote the same baseline with and without Lipschitz regularization, respectively. For all the approaches we specify the dimensionality d_{fs} of a feature space where the optimization is performed. This is the intrinsic dimensionality that is assumed by the algorithm during optimization. Note that this value may differ from the true intrinsic dimensionality d of the objective functions proposed in the following i.e. $d_{fs} \neq d$.

Acquisition functions

We evaluate the performances of all baselines across a set of common acquisition functions: *expected improvement* (EI) [Moč75], *upper confidence bound* (UCB) [SKKS10] and *probability of improvement* (PI) [Kus64]. The motivation for selecting these acquisition function is that we wish to test performances of our Bayesian optimization approach on a range of different decision strategies: aggressive exploitation (PI), aggressive exploration (UCB) and one-time-step optimal selection (EI). Note that the improvement based acquisition function PI is not necessarily prone to aggressive exploitation. In fact this acquisition function is based on the improvement with respect to the true optimum $f(\mathbf{x}^*)$. However, since we do not have access to the true optimum prior or during the optimization, we approximate it with the best noisy observation obtained up to iteration t, that is $y_{\min} := \min \mathbf{y}_t$. This change in the definition of the acquisition function favours aggressive exploitation behaviour. The substitution with the best noisy observation $y_{\min} := \min \mathbf{y}_t$ instead of the true optimum $f(\mathbf{x}^*)$ for improvement based acquisition functions is applied also to the expected improvement acquisition.

In our experiments, we set the β_t parameter of UCB acquisition function in equation (2.21) to $\sqrt{3}$. This choice of the parameter β_t is common in Bayesian optimization [WMHD17]. We apply an identical algorithm for the maximization of the acquisition function to all baselines:

we first perform a random search step with 5000 samples drawn uniformly at random, then we select the most promising 100 locations and we apply gradient based optimization from these 100 starting locations. For box constrained acquisition maximization we use the L-BFGS-B optimizer [BLNZ95, ZBLN97]. For constrained acquisition maximization with nonlinear Lipschitz constraints we use a trust-region interior point method [BHN99].

Model parameters

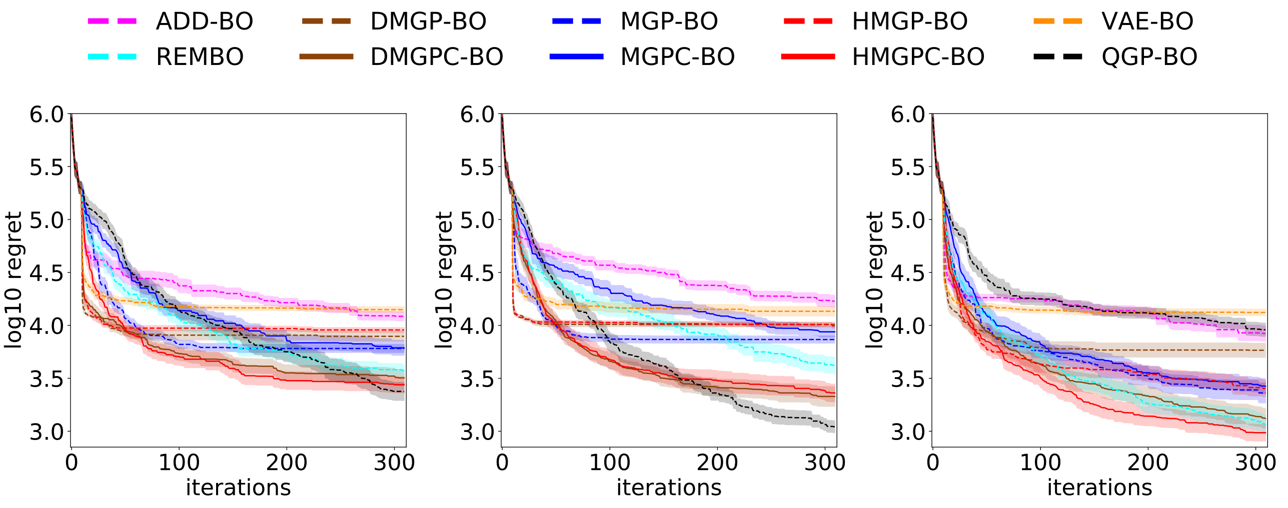
In our experiments, we select the $Mat\acute{e}rn_{5/2}$ kernel, defined in equation (2.12) of Section 2.2 as the covariance function for the Gaussian processes in each baseline. For the neural network employed in the encoder, the architecture was a single hidden layer with 20 units, and as the activation function we use the sigmoid activation.

Experiment setup

Each Bayesian optimization progression curve shows the mean and standard error of the immediate logarithmic regret $\log_{10} |f(\mathbf{x}_{\text{best}}(t)) - f_{\text{min}}|$, where f_{min} is the true optimum of f, that is the objective function evaluated at the true minimizer $f_{\text{min}} = f(\mathbf{x}^*)$, and $\mathbf{x}_{\text{best}}(t)$ is the location of the lowest value of the objective function encountered during the optimization up to iteration t, that is $\mathbf{x}_{\text{best}}(t) \in \arg\min_{i=1:t} f(\mathbf{x}_i)$. Mean and standard error are computed over 20 experiments with different random initialization. The initialization of the starting points is drawn uniformly at random in the interval $[0, 1]^D$. All optimization experiments start with a budget of 10 data points and perform a total of 300 Bayesian optimization iterations. The true noise variance to be learned as a parameter in our experiments is set to the value of 10^{-4}.

4.5.1 Linear feature space

We consider benchmark functions that are defined in a $d = 10$-dimensional space. We map their input space to a $D = 60$-dimensional space using an orthogonal matrix $\mathbf{R}^{d\times D}$ so that the overall objective is $f(\mathbf{z}) = f(\mathbf{Rx})$. For each objective function, care is taken to ensure that



(a) EI (b) UCB (c) PI

Figure 4.2: Results with Rosenbrock objective function of Bayesian optimization in feature space. The objective function is characterized by a linear embedding to reach $D = 60$ dimensions. Baselines MGPC-BO, HMGPC-BO and DMGPC-BO (solid lines) apply nonlinearly constrained acquisition maximization and recover no worse regret at termination than the unconstrained versions MGP-BO, HMGP-BO and DMGP-BO.

there exists a location $\mathbf{x}^* \in [0,1]^D$ that maps to the actual minimizer $\mathbf{R}\mathbf{x}^* \in \arg\min_{\mathbf{z}} f(\mathbf{z})$ of the low-dimensional benchmark f.

Additive objective

We minimize the *Rosenbrock* benchmark function

$$f(\mathbf{z}) = \sum_{i=1}^{d-1} [100(z_{i+1} - z_i^2)^2 + (z_i - 1)^2] \tag{4.33}$$

in a $d_{fs} = 10$-dimensional feature space. Note that in this particular case the intrinsic dimensionality of the objective function is equal to the assumed dimensionality of the feature space, that is $d = d_{fs} = 10$. Figure 4.2 shows that HMGPC-BO baseline descends quickly to relatively low regret in the early stages of optimization and recovers better regret at termination than the unconstrained baseline HMGP-BO. The VAE-BO baseline features a steep descent but also flattens out quite early. This behaviour is due to a lack of exploration from the feature space to data space. In particular, the reconstruction mapping is insufficiently expressive and ends up mapping most feature space locations to similar values in data space. We highlight that the model of the VAE-BO baseline was trained on a budget of 500 inputs-observation pairs prior to starting the BO experiments. This additional budget, however, still does not allow the VAE-BO to compare well with baselines that learn a feature mapping and a reconstruction mapping during optimization. The baseline REMBO shows a competitive descent for two main reasons: i) the fact that the baseline conforms with the linear embedding assumption that characterizes the intrinsically low dimensional objective function and ii) the employment of an orthonormal linear mapping which is supposed to improve performances and again conforms to structural assumption about the linear embedding $\mathbf{R}$. The ADD-BO baseline suffers from the coupling effects of the linear dimensionality reduction $\mathbf{R}$. In fact, the underlying assumption with ADD-BO is that additive components can be optimized independently. However, due to the linear dimensionality reduction $\mathbf{R}$ this assumption is no longer satisfied. Therefore, albeit being an additive benchmark function, ADD-BO struggles to find a competitive solution. The QGP-BO baseline is a competitive baseline that reaches the best value at termination of the optimization for two out of three acquisition functions. The QGP-BO baseline, however, is characterized by a slower descent in the early iterations and is sensitive with respect to the choice of the acquisition function. The poor performances may be due to a too aggressive exploitation behavior with PI acquisition. Overall, Figure 4.2 highlights the fast learning of feature space representations that are effective for optimization with MGPC-BO, HMGPC-BO and DMGPC-BO baselines.

Non-additive objective

Here, we optimize the *product of sines* with intrinsic dimensionality $d = 10$

$$f(\mathbf{z}) = 10 \sin(z_1) \prod_{i=1}^{d} \sin(z_i) \tag{4.34}$$

and compare results when the additivity assumption is not satisfied. Figure 4.3 shows the regret curves obtained optimizing the objective on a $d_{fs} = 10$-dimensional feature space. Note that also in this case the dimensionality of the true objective function f is equal to the dimensionality assumed for the feature space in our optimization, that is $d = d_{fs} = 10$.

Solid lines describe the Lipschitz-regularized baselines MGPC-BO, HMGPC-BO and DMGPC-BO (with nonlinear constraint), while dashed lines are baselines that apply box-constrained maximization of the acquisition in feature space. The regrets of HMGP-BO, MGP-BO and DMGP-BO (box constrained baselines) flatten quite early in both improvement-based acquisition functions (EI and PI) since these acquisition functions highlight locations in feature space that are too far away from the training data. In this setting, the decoder $\boldsymbol{g}$ returns the same high-dimensional reconstruction, which prevents the Bayesian optimization algorithm from exploring. The constrained maximization of the acquisition function is beneficial for all our models. We also note that the REMBO baseline conforms to the intrinsic linear low-dimensionality assumption described in Section 4.5.1 and is the most competitive baseline especially for UCB acquisition function. However, the linear reconstruction mapping applied by REMBO also suffers from non-injectivity, and this slows down exploration in the high-dimensional space. The linear projection deteriorates performances of the additive model. ADD-BO assumes independence between axis-aligned projections of the high-dimensional space, while the linear mapping $\mathbf{R}$ couples all subsets of dimensions. This linear mapping, therefore, penalizes optimization with independent additive components. The VAE-BO approach requires much larger amounts of data to learn a meaningful reconstruction mapping than available in our experiment. Therefore, most locations in feature space are mapped to similar reconstructions. This explains the flat curve observed on all VAE-BO progressions with different acquisition functions. Again the performances of the QGP-BO depend on the choice of the acquisition function. With an aggressive exploration strategy, such as UCB, QGP-BO is very competitive. This highlights that the QGP model can be employed also when the relevant dimensions of the objective function are not axis-aligned. With improvement based acquisition functions, such as EI and PI, the optimization of the QGP-BO baseline remains stuck in local optima due to a more aggressive exploitation.

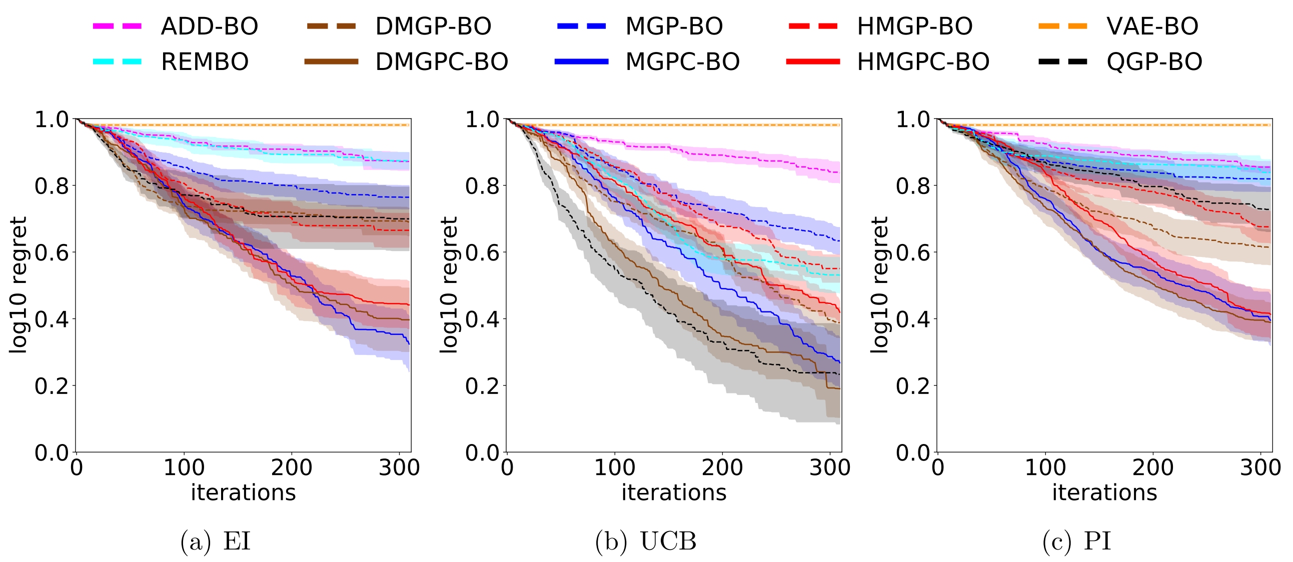
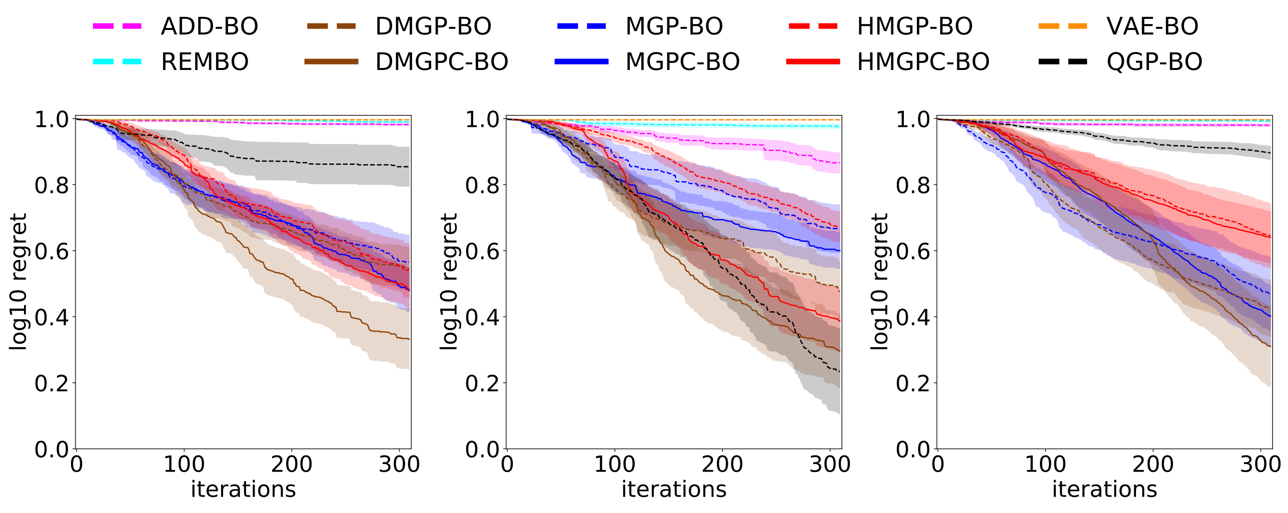


Figure 4.3: Optimization progression on product of sines characterized by linear embedding with EI 4.3(a), UCB 4.3(b) and PI 4.3(c). Baselines with Lipschitz nonlinear constraint, namely MGPC-BO, HMGPC-BO and DMGPC-BO learn low-dimensional representations of the objective that are useful for optimization. Curves and confidence bounds represent mean and standard error over 20 experiments with different restarts, respectively.

4.5.2 Nonlinear feature space with non-additive objective

We consider the product of sines function and, in this section, we apply a nonlinear dimensionality reduction to the objective function. We define a single-layer neural network mapping to elevate the dimensionality of the objective to $D = 60$, i.e. $f(\gamma(\mathbf{Rx}))$. Here γ is the sigmoid activation function. We select a dimensionality of the feature space as in previous sections $d_{fs} = 10$ which is equal to the intrinsic dimensionality of the objective function $d = 10$. Figure shows the progression of the regret over 300 Bayesian optimization iterations. We can observe consistent improvements of MGPC-BO, HMGPC-BO and DMGPC-BO with respect to VAE-BO which also assumes a nonlinear embedding for the objective. The performance of MGPC-BO, HMGPC-BO and DMGPC-BO also retain better regret at termination of the optimization than with box-constrained acquisition maximization (MGP-BO, HMGP-BO, DMGP-BO, respectively). Here, we apply a significance testing with the Wilcoxon signed-rank test [Wil92]. A Wilcoxon signed rank test is a nonparametric statistical hypothesis test. It is employed to assess whether two observed sets of samples are drawn from the same distribution. We apply the Wilcoxon signed rank test to assess whether the log-regrets, collected at the last BO iteration of two baselines, are sampled according to the same distribution. We perform 20 different runs of BO



(a) EI (b) UCB (c) PI

Figure 4.4: BO performances expressed as log regret of the product of sines function in a nonlinear embedding. Results are shown for EI 4.4(a), UCB 4.4(b) and PI 4.4(c). All our baselines with nonlinear constraint, namely MGPC-BO, DMGPC-BO and HMGPC-BO learn useful representations in feature space for optimization. There is highly significant difference of 0.014% between DMGPC-BO and ADD-BO.

per each baseline (20 different random initializations). Therefore, the sample of each baseline consists of the 20-log-regrets at termination of the BO iterations corresponding to that baseline. The Wilcoxon signed rank test assigns a p-value to a pair of samples. Small p-values (in the order of magnitude of 10^{-3}) correspond to highly significantly different samples. This means that the two baselines are significantly different in performances. We apply the Wilcoxson signed rank test between the best performing of our baselines, namely DMGPC-BO and the best competitive baseline that is ADD-BO. We observe a significance of at least 0.014% for all acquisition functions (largest p-value $p = 0.00014$ for UCB acquisition) meaning that our best baseline DMGPC-BO is highly significantly different than the ADD-BO baseline and attains better regret than ADD-BO at termination of the optimization.

Overall, we observe that the constrained maximization of the acquisition function is beneficial for the proposed model. The advantages with respect to ADD-BO, REMBO and VAE-BO baselines are more evident with the product of sines objective with nonlinear embedding while with the Rosenbrock we retain no worse regret. The QGP-BO baseline maintains competitive performances across product of sines and Rosenbrock objective functions for the UCB acquisition. When using improvement based acquisition functions the optimization in feature space retains similar performances while the QGP-BO suffers from too aggressive exploitation.

4.5.3 Sensitivity analysis on real data

Here we apply a sensitivity analysis with respect to the dimensionality of the feature space d_{fs} on a $D = 12$-dimensional real problem. We consider the Thomson problem of finding the lowest potential configuration of a set of electrons on a sphere [DMM04]. This is a central problem in physics and chemistry for identifying a structure with respect to atomic locations [DMM04]. The potential of a set of n_p electrons on a unit sphere is given by the objective

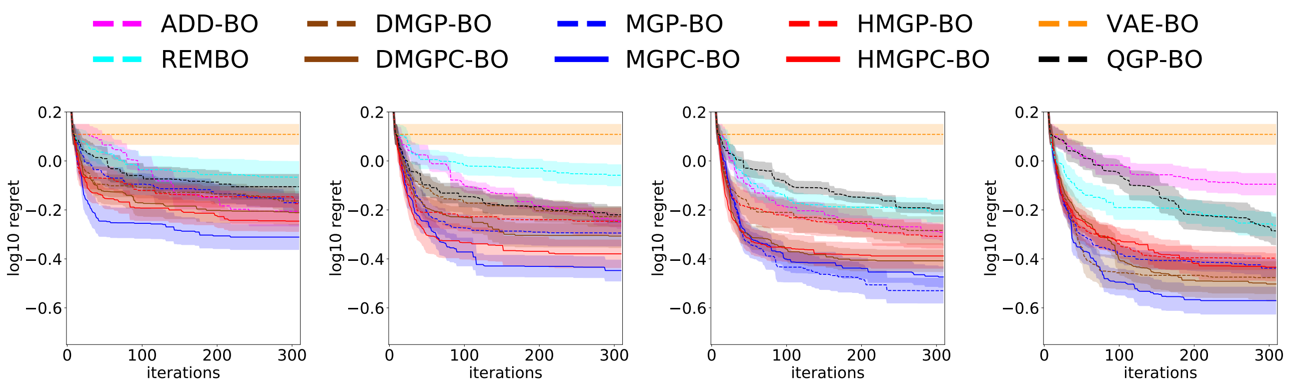
$$\sum_{i=1}^{n_p-1} \sum_{j=i+1}^{n_p} ((x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2)^{-\frac{1}{2}}. \tag{4.35}$$

This is a constrained minimization problem with constraints $x_i^2 + y_i^2 + z_i^2 = 1$ for $i = 1, ..., n_p$, which means that all electrons must lie on a unit sphere. We represent the variables of the problem as spherical coordinates with unit radius. This allows us to define two variables per point with a total number of $2n_p$ (azimuthal and polar angles) parameters to optimize within box constraints. For optimization, we select $n_p = 6$m which results in a $D = 12$-dimensional problem and we optimize it on low-dimensional feature spaces of dimensionalities $d_{fs} = 6, 4, 3, 2$ to observe the effect of this hyper-parameter in the optimization.

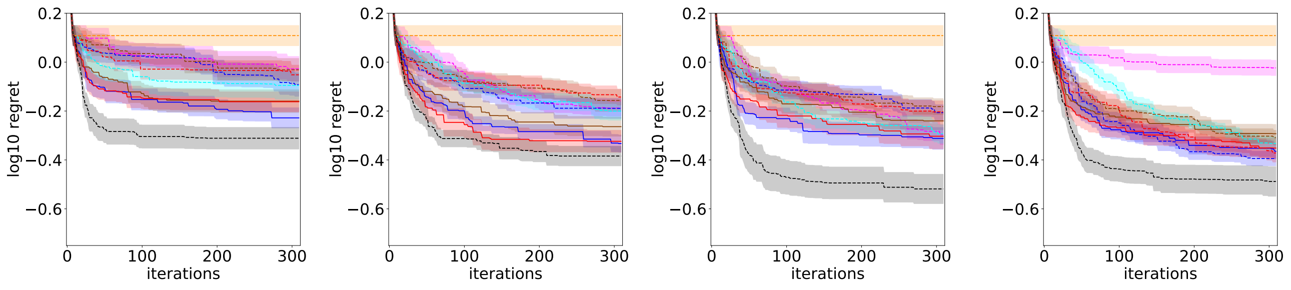
Figure 4.5 shows a comparison of our approaches with ADD-BO, REMBO and VAE-BO baselines on all three acquisition functions: expected improvement, upper confidence bound and probability of improvement. For the acquisition function PI, overall, we observe a deterioration of performances with diminishing dimensionality of the feature space. The regret clearly increases for our baselines when we select $d_{fs} = 2$ meaning that, with a high compression rate, the probabilistic model for MGPC-BO, DGPC-BO and HMGPC-BO learns less useful features for optimization. We observe the most competitive baseline to be ADD-BO which decomposes the 12-dimensional problems into D/d_{fs} sub-problems with dimensionality d_{fs}. A similar approach that decomposes the problem into independent subproblems is the QGP-BO baseline. This baseline retains similar regret to ADD-BO and performs better than the additive model for $d_{fs} = 6$. Another competitive baseline is REMBO, which uses a linear embedding for optimization. We highlight that for the REMBO baseline we perform optimization on a linear subspace within the interval $[-5/\sqrt{d_{fs}}, 5/\sqrt{d_{fs}}]$. This allows us to maintain the same ratio between i) coordinates that get reconstructed inside the data space and ii) coordinates that need to be projected onto the hypercube in data space because they happen to be outside the allowed interval in data space. By performing optimization within the interval $[-5/\sqrt{d_{fs}}, 5/\sqrt{d_{fs}}]$, this ratio is maintained constant across the different dimensionalities $d_{fs} = 2, 3, 4, 6$ of the linear feature space. The effect of this constant ratio is a meaningful deterioration of performance of optimization as we decrease the dimensionality d_{fs} of the linear feature space.

In Figure 4.5 we also report the results obtained on the sensitivity analysis with the expected improvement acquisition function. Overall, we observe a deterioration of performances for our baseline, namely MGPC-BO, HMGPC-BO and DMGPC-BO (solid lines). The best performances are retained by the QGP-BO baseline and reaches lower regret as we increase the dimensionality of the feature space d_{fs}. Both MGPC-BO and the hierarchical ICM model HMGPC-BO are competitive both in terms of data efficiency and value of the regret at termination of the optimization. In particular, by data efficiency we mean how quickly a good optimum is found by the algorithm. Therefore data efficiency translates into how steep is the descent for the optimization curve (steeper is more data efficient). Also REMBO progressively reaches lower regret values as we increase the dimensionality d_{fs} of the feature space. The baseline REMBO remains competitive for all dimensionalities of the feature space. VAE-BO fails to learn useful representations of data in feature space and maintains high regret over all dimensionalities $d_{fs} = 2, 3, 4, 6$ of the feature space.

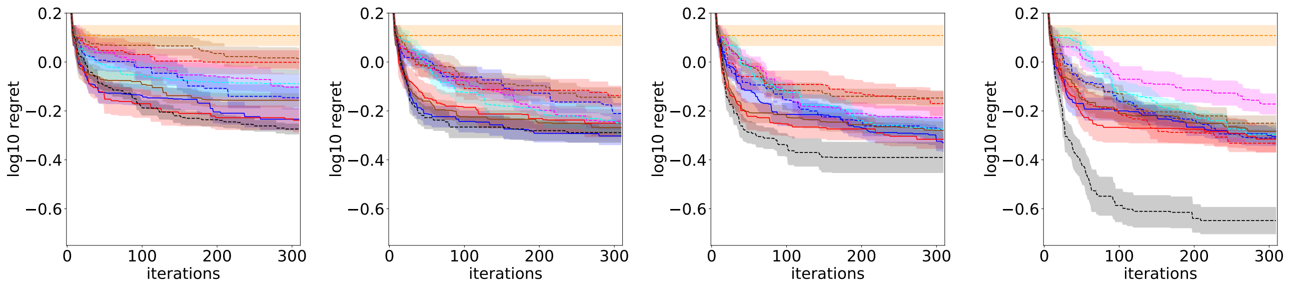
Finally, we analyze the results obtained with the upper confidence bound acquisition function. Here we do not observe a substancial deterioration of performances between $d_{fs} = 3$ and $d_{fs} = 2$. However, we still observe lower regret as we progress towards higher feature space dimensionalities, namely $d_{fs} = 4, 6$. Both REMBO and ADD-BO are competitive baselines for optimization progression and value of the regret at termination. Because of the poor learning of the embedding, VAE-BO does not perform comparatively well with the rest of the baselines also in this acquisition function. Overall, we observe QGP-BO as the best baseline in the comparison.



(a) $d_{fs} = 2$, PI (b) $d_{fs} = 3$, PI (c) $d_{fs} = 4$, PI (d) $d_{fs} = 6$, PI



(e) $d_{fs} = 2$, EI (f) $d_{fs} = 3$, EI (g) $d_{fs} = 4$, EI (h) $d_{fs} = 6$, EI



(i) $d_{fs} = 2$, UCB (j) $d_{fs} = 3$, UCB (k) $d_{fs} = 4$, UCB (l) $d_{fs} = 6$, UCB

Figure 4.5: Sensitivity analysis with respect to the dimensionality of the feature space d_{fs} on a real problem set. We test all approaches on a set of feature space dimensionalities $d_{fs} = 2, 3, 4, 6$. The performances of our baselines clearly deteriorate for $d_{fs} = 2$. Our baseline MGPC-BO show better performances than the best competing baseline ADD-BO and REMBO and reach the minimum in notably less iterations.

The choice of acquisition function UCB proves effective with the QGP-BO baseline in all our experiments. MGPC-BO and HMGPC-BO are competitive baselines in the comparison and perform better than ADD-BO, REMBO and VAE-BO.

We apply a significance test to the results obtained with PI acquisition function and compare our nonlinearly constrained baseline MGPC-BO with the competitive baselines (ADD-BO and REMBO) for each plot of Figure 4.5 at termination of the optimization. We select the Wilcoxon signed-rank test [Wil92], which does not assume that the difference between the sample populations is Gaussian. For feature space dimensionality $d_{fs} = 2$ we do not observe values significantly different since the p-value is $p = 0.135$. This is due to the deterioration of performances at $d_{fs} = 2$ for the optimization with probability of improvement acquisition function. For $d_{fs} = 3$ we observe a more significant difference between MGPC-BO and ADD-BO with p-value $p \leq 0.002$. With hyper-parameter values $d_{fs} \geq 4$ we observe significantly different baselines with significance at 0.6% (difference between MGPC-BO and ADD-BO for $d_{fs} = 4$ with p-value $p \leq 0.003$ and between MGPC-BO and REMBO for $d_{fs} = 6$ with p-value $p \leq 0.006$). Overall, we observe our constrained baselines to perform better than ADD-BO and REMBO and to reach the lowest value in notably less BO iterations.

4.5.4 Run-time complexity

The computational complexity of MGPC-BO is $\mathcal{O}(D^3 + N^3)$ due to the eigen-decomposition of both the coregionalization matrix (D^3) and kernel matrix (N^3). The baseline HMGPC-BO scales with $\mathcal{O}(d_{out}^3 Q + N^3 Q)$ with Q being the number of independent subsets of dimensions, i.e. $Q = D/d_{out}$, with d_{out} being a small constant value ($d_{out} = 3$). This baseline achieves faster computations when having small number of data points N, for large number of data points and large number of dimensions (both tending to infinity) the MGPC-BO results more efficient. The baseline DMGPC-BO instead has complexity $\mathcal{O}(d_{out}^3 Q + N^3)$, which is faster than the MGPC-BO. MGP-BO, DMGP-BO and HMGP-BO have the same complexity of MGPC-BO, DMGPC-BO and HMGPC-BO, respectively. The remaining baselines have all computational complexity $\mathcal{O}(N^3)$ due to the matrix inversion of the covariance matrix for GP training which is used in ADD-BO, REMBO and VAE-BO. In the QGP-BO the cubic complexity is obtained by computing the approximate posterior in the EP routine. Our baseline has an additional overhead of at least a linear term $d_{out}^3 Q$ which implies slower training times for our probabilistic model. This is a reasonable trade off for improved optimization performances and better data efficiency in our reconstruction model. In the comparison with the QGP-BO baseline we note that the performances of QGP-BO are dependent on the choice of the acquisition function while the MGPC-BO, HMGPC-BO and DMGPC-BO are competitive (better than ADD-BO, REMBO and VAE-BO) across all acquisitions.

4.6 Summary

In this chapter, we proposed a framework for efficient Bayesian optimization of intrinsically low-dimensional black-box functions based on nonlinear embeddings. In our model, a manifold GP learns useful low-dimensional feature representations of high-dimensional data by jointly learning the response surface and a reconstruction mapping. Our approach allows for optimizing acquisition functions in a low-dimensional feature space. Since exploration in feature space (driven by the acquisition function) does not necessarily mean exploration in the high-dimensional parameter space, we introduce a nonlinear constraint based on Lipschitz continuity of predictions of the reconstruction mapping, which encourages exploration in the vicinity of the training data and avoids un-identifiability issues in data space, which would otherwise hinder optimization.

Chapter 5

Conclusion

We have presented a spectrum of approaches for high dimensional Bayesian optimization that range from the decomposition strategies (in Chapter 3) to the feature space learning (in Chapter 4). In particular, we have shown empirical evidence that the quantile Gaussian process model is suitable for Bayesian optimization. This is possible when the optimization problem is reformulated as a collection of sub-problems defined on disjoint axis-aligned projections. We have also provided empirical evidence that the low intrinsic dimensionality assumption can be exploited by learning a nonlinear projection with manifold Gaussian processes. We presented a method that learns a feature space online during optimization and therefore allows exploiting all the data available up to iteration t. This is possible thanks to a probabilistic model that allows encoding, decoding and learning the response surface by means of manifold Gaussian processes.

In both our contributions we have presented a Bayesian optimization strategy that allows for exploiting all the data available at each Bayesian optimization iteration. In fact, the quantile Gaussian process uses all the data available and applies an automatic selection of the most promising observations according to the quantile. On the other hand, the optimization in feature space is characterized by a probabilistic model that can be learned online during optimization. This allows exploiting the data collected at each Bayesian optimization iteration.

In our work, we have shown that a sensitivity analysis with respect to a hyper-parameter can lead to an improvement of performances. In fact, with the sensitivity analysis in Chapter

3, we represented the problem of identifying the quantile τ not as a learning problem but as an optimization problem with respect to BO performances. The difference is that we no longer search hyper-parameters of the model that are able to explain the data but rather that improve the performances of the BO algorithm. Our conclusion is therefore that not always the hyper-parameter that explains the data well is the one that yields the better performances.

Thanks to the contributions provided in this thesis a number of problems may now be addressed without incurring in the curse of dimensionality. Among these problems we highlight: optimizing the 14 parameters of a random forest body part classifier [SFC$^+$11]. For instance, we could now address the problem of optimizing walking controllers for a simulated hexapod robot [LCRB20]. With our Bayesian optimization algorithms we could now optimize policy parameters such that the robot can walk to a target location while avoiding high joint velocities and height deviations. Another instance problem is selecting the most promising parameter configurations to improve the performances of integer programming solvers [HHLB11].

In conclusion, optimizing a high-dimensional objective function that has intrinsic low dimensionality is a feasible challenge. In this challenge, projections constitute a fundamental step for tackling the curse of dimensionality and scaling Bayesian optimization. Another important step is the definition of a probabilistic model that is sensible with respect to the choice of projection. For instance, the quantile Gaussian process is sensible choice in the presence of inconsistencies. On the other hand a manifold Gaussian process is essential for dimensionality reduction and both response surface learning and reconstruction mapping. Using projections and defining sensible probabilistic models is crucial and has proven effective in high-dimensional Bayesian optimization.

5.1 Future Work

One of the downsides of the quantile Gaussian process approach is the computational efficiency. The quantile Gaussian process approximates the marginal likelihood and the posterior predictions by means of expectation propagation. This step has computational complexity that is cubic in the number of data points and becomes computationally challenging for large numbers of data points. A possible solution to this problem is the employment of sparse Gaussian process regression methods that will reduce the computational burden of both the inference and marginal likelihood computation [SG06].

A challenging aspect with manifold Gaussian processes is that they are prone to overfitting in the presence of large number of parameters [CPRD16]. These are the parameters present in the deep network used as encoder. This behavior is due to the optimization of the marginal likelihood, which may get stuck in local optima during the optimization of the hyper-parameters. A possible alternative to the Bayesian model selection that maximizes the marginal likelihood is the maximization of the lower bound on the marginal likelihood. In particular, we could introduce uncertainty in the feature space and treat each input in $\mathscr{Z}$ space as a latent variable. We could then apply the same approximations that have been used for Gaussian processes with uncertain inputs for inference and training. This would allow defining a multi-output Gaussian process as the encoder instead of the parametric neural network and therefore avoiding over-parametrization of the probabilistic model.

By using axis aligned projections and by learning a feature space we have shown stratagems for dimensionality reduction. These stratagems allow us to transform input representations to make them amenable to Bayesian optimization. In this regard, nonlinear projection may extend the work of Benjamin Letham, Roberto Calandra et al. [LCRB20] who provide a deep analysis on the use of linear embeddings for high-dimensional Bayesian optimization. Another ramification of our work may concern Calandra's work on Bayesian gait optimization for bipedal locomotion [CGS^{+}14, CSPD16]. In fact, using manifold Gaussian processes for response surface learning would be beneficial for optimization. Manifold Gaussian processes are able to model the nonlinearities inherent in the bipedal walking motion [CPRD16].

Bibliography

[ÁL09] Mauricio A. Álvarez and Neil D. Lawrence. Sparse convolved Gaussian processes for multi-output regression. *Advances in Neural Information Processing Systems*, 2009.

[ÁL11] Mauricio A. Álvarez and Neil D. Lawrence. Computationally efficient convolved multiple output Gaussian processes. *Journal of Machine Learning Research*, 12:1459–1500, 2011.

[ÁRL11] Mauricio A. Álvarez, Lorenzo Rosasco, and Neil D. Lawrence. Kernels for vector-valued functions: a review. *Foundations and Trends in Machine Learning*, 2011.

[ATOF18] Gabriele Abbati, Alessandra Tosi, Michael A. Osborne, and Seth Flaxman. Adageo: adaptive geometric learning for optimization and sampling. *International Conference on Artificial Intelligence and Statistics*, 2018.

[BB12] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

[BBC12] Alexis Boukouvalas, Remi Barillec, and Dan Cornford. Direct Gaussian process quantile regression using expectation propagation. *International Conference on Machine Learning*, 2012.

[BBM04] Thomas Bartz-Beielstein and Sandor Markon. Tuning search algorithms for real-world applications: a regression tree based approach. *Congress on Evolutionary Computation*, 2004.

[BCS$^+$09] M. Yu Byron, John P. Cunningham, Gopal Santhanam, Stephen I. Ryu, Krishna V. Shenoy, and Maneesh Sahani. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *Advances in Neural Information Processing Systems*, 2009.

[Bet91] Bruno Betrò. Bayesian methods in global optimization. *Journal of Global Optimization*, 1(1):1–14, 1991.

[BF05] Phillip Boyle and Marcus Frean. Dependent Gaussian processes. *Advances in Neural Information Processing Systems*, 2005.

[BHBG07] Mustafa Baz, Brady Hunsaker, Paul J. Brooks, and Abhijit Gosavi. Automated tuning of optimization software parameters. *Technical Report, University of Pittsburgh*, 2007.

[BHN99] Richard H. Byrd, Mary E. Hribar, and Jorge Nocedal. An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900, 1999.

[BK10] Rémi Bardenet and Balázs Kégl. Surrogating the surrogate: accelerating Gaussian-process-based global optimization with a mixture cross-entropy algorithm. *International Conference on Machine Learning*, 2010.

[BLNZ95] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[Boc59] Salomon Bochner. *Lectures on Fourier integrals.* Princeton University Press, 1959.

[Bre01] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[BSK16] Felix Berkenkamp, Angela P. Schoellig, and Andreas Krause. Safe controller optimization for quadrotors with Gaussian processes. *International Conference on Robotics and Automation*, 2016.

[CCK12] Bo Chen, Rui Castro, and Andreas Krause. Joint optimization and variable selection of high-dimensional Gaussian processes. *International Conference on Machine Learning*, 2012.

[CCTM15] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

[CGS$^+$14] Roberto Calandra, Nakul Gopalan, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian gait optimization for bipedal locomotion. In *International conference on learning and intelligent optimization*, pages 274–290. Springer, 2014.

[CL11] Olivier Chapelle and Lihong Li. An empirical evaluation of Thompson sampling. *Advances in Neural Information Processing Systems*, 2011.

[CPRD16] Roberto Calandra, Jan Peters, Carl E. Rasmussen, and Marc P. Deisenroth. Manifold Gaussian processes for regression. *International Joint Conference on Neural Networks*, 2016.

[CSPD16] Roberto Calandra, André Seyfarth, Jan Peters, and Marc P. Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1):5–23, 2016.

[Dam15] Andreas Damianou. *Deep Gaussian processes and variational propagation of uncertainty.* PhD thesis, University of Sheffield, 2015.

[DDGL16] Zhenwen Dai, Andreas Damianou, Javier González, and Neil D. Lawrence. Variational auto-encoded deep Gaussian processes. *International Conference on Learning Representations*, 2016.

[DL13] Andreas Damianou and Neil D. Lawrence. Deep Gaussian processes. *International Conference on Artificial Intelligence and Statistics*, 2013.

[DMM04] Elizabeth D. Dolan, Jorge J. Moré, and Todd S. Munson. Benchmarking optimization software with COPS 3.0. *Technical Report, Argonne National Lab.*, 2004.

[Fra18] Peter I. Frazier. A tutorial on Bayesian optimization. *arXiv:1807.02811*, 2018.

[GBWD$^+$18] Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018.

[GDHL16] Javier González, Zhenwen Dai, Philipp Hennig, and Neil D. Lawrence. Batch Bayesian optimization via local penalization. *International Conference on Artificial Intelligence and Statistics*, 2016.

[GHL17] Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. Constrained Bayesian optimization for automatic chemical design. *arXiv:1709.05501*, 2017.

[GKKW06] László Györfi, Michael Kohler, Adam Krzyzak, and Harro Walk. *A Distribution-Free Theory of Nonparametric Regression.* Springer Science & Business Media, 2006.

[GLJL14] Javier González, Joseph Longworth, David C. James, and Neil D. Lawrence. Bayesian optimization for synthetic gene design. *NIPS Workshop in Bayesian Optimization*, 2014.

[GOH14] Roman Garnett, Michael A. Osborne, and Philipp Hennig. Active learning of linear embeddings for Gaussian processes. *Conference on Uncertainty in Artificial Intelligence*, 2014.

[Goo97] Pierre Goovaerts. *Geostatistics for Natural Resources Evaluation.* Oxford University Press, 1997.

[GSC13] Elad Gilboa, Yunus Saatçi, and John P. Cunningham. Scaling multidimensional inference for structured Gaussian processes. *Transactions on Pattern Analysis and Machine Intelligence*, 37(2):424–436, 2013.

[GVL96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

[HBB$^+$19] Ali Hebbal, Loic Brevault, Mathieu Balesdent, El-Ghazali Talbi, and Nouredine Melab. Bayesian optimization using deep Gaussian processes. *arXiv:1905.03350*, 2019.

[HHLB11] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. *International Conference on Learning and Intelligent Optimization*, 2011.

[HJ12] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 2012.

[HL14] James Hensman and Neil D. Lawrence. Nested variational compression in deep Gaussian processes. *arXiv:1412.1370*, 2014.

[HLHG14] José Miguel Hernández-Lobato, Matthew W. Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. *Advances in Neural Information Processing Systems*, 2014.

[HS12] Philipp Hennig and Christian J. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(1):1809–1837, 2012.

[JAGS17] Rodolphe Jenatton, Cedric Archambeau, Javier González, and Matthias Seeger. Bayesian optimization with tree-structured dependencies. *International Conference on Machine Learning*, 2017.

[Jol03] Ian T. Jolliffe. Principal component analysis. *Technometrics*, 2003.

[Jon01] Donald R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001.

[JSW98] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[KH01] Roger Koenker and Kevin F. Hallock. Quantile regression. *Journal of Economic Perspectives*, 15(4):143–156, 2001.

[KPHL17] Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. *International Conference on Machine Learning*, 2017.

[Kri51] Daniel G Krige. A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, 52(6):119–139, 1951.

[KSP15] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional Bayesian optimisation and bandits via additive models. *International Conference on Machine Learning*, 2015.

[Kus64] Harold J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 1964.

[KW14] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. *International Conference on Machine Learning*, 2014.

[Law05] Neil D. Lawrence. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research*, 6(Nov):1783–1816, 2005.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[LCLS10] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. *International Conference on World Wide Web*, 2010.

[LCRB20] Ben Letham, Roberto Calandra, Akshara Rai, and Eytan Bakshy. Re-examining linear embeddings for high-dimensional bayesian optimization. *Advances in Neural Information Processing Systems*, 33, 2020.

[LGDL18] Xiaoyu Lu, Javier González, Zhenwen Dai, and Neil D. Lawrence. Structured variationally auto-encoded optimization. *International Conference on Machine Learning*, 2018.

[LGQCRFV10] Miguel Lázaro-Gredilla, Joaquin Quiñonero-Candela, Carl E. Rasmussen, and Aníbal R. Figueiras-Vidal. Sparse spectrum Gaussian process regression. *Journal of Machine Learning Research*, 11:1865–1881, 2010.

[Liz08] Daniel J. Lizotte. *Practical Bayesian optimization.* PhD thesis, University of Alberta, 2008.

[Loc97] Marco Locatelli. Bayesian algorithms for one-dimensional global optimization. *Journal of Global Optimization*, 10(1):57–76, 1997.

[LQC06] Neil D. Lawrence and Joaquin Quiñonero-Candela. Local distance preservation in the GP-LVM through back constraints. *International Conference on Machine Learning*, 2006.

[MDK20] Riccardo Moriconi, Marc P. Deisenroth, and Sesh K.S. Kumar. High-dimensional Bayesian optimization using low-dimensional feature spaces. *Machine Learning*, pages 1–19, 2020.

[Min01] Thomas P. Minka. *A family of algorithms for approximate Bayesian inference.* PhD thesis, Massachusetts Institute of Technology, 2001.

[MKD20] Riccardo Moriconi, Sesh K.S. Kumar, and Marc P. Deisenroth. High-dimensional Bayesian optimization with projections using quantile Gaussian processes. *Optimization Letters*, 14(1):51–64, 2020.

[Moč75] Jonas Močkus. On Bayesian methods for seeking the extremum. *Optimization Techniques IFIP Technical Conference*, 1975.

[Moč94] Jonas Močkus. Application of Bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimization*, 4(4):347–365, 1994.

[MS02] Paul Milgrom and Ilya Segal. Envelope theorems for arbitrary choice sets. *Econometrica*, 70(2):583–601, 2002.

[MTZ78] Jonas Močkus, Vytautas Tiesis, and Antanas Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2(2):117–129, 1978.

[ORR$^+$08] Michael A. Osborne, Stephen J. Roberts, Alex Rogers, Sarvapali D. Ramchurn, and Nicholas R. Jennings. Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes. *International Conference on Information Processing in Sensor Networks*, 2008.

[Pea01] Karl Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[PP08] Kaare B. Petersen and Michael S. Pedersen. The matrix cookbook. *Technical University of Denmark*, 2008.

[RHB99] Ken F. Riley, Michael P. Hobson, and Stephen J. Bence. *Mathematical Methods for Physics and Engineering.* Cambridge University Press, 1999.

[RLG$^+$17] Santu Rana, Cheng Li, Sunil Gupta, Vu Nguyen, and Svetha Venkatesh. High dimensional Bayesian optimization with elastic Gaussian process. *International Conference on Machine Learning*, 2017.

[RMW14] Danilo J. Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and variational inference in deep latent Gaussian models. *International Conference on Machine Learning*, 2014.

[RR08] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems*, 2008.

[RSBC18] Paul Rolland, Jonathan Scarlett, Ilija Bogunovic, and Volkan Cevher. High-dimensional Bayesian optimization via additive models with overlapping groups. *International Conference on Artificial Intelligence and Statistics*, 2018.

[RW06] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[RY13] Daniel Revuz and Marc Yor. *Continuous martingales and Brownian motion*, volume 293. Springer Science and Business Media, 2013.

[Saa12] Yunus Saatçi. *Scalable Inference for Structured Gaussian Process Models*. PhD thesis, University of Cambridge, 2012.

[SD17] Hugh Salimbeni and Marc P. Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. *Advances in Neural Information Processing Systems*, 2017.

[SFC$^+$11] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *CVPR 2011*, pages 1297–1304. Ieee, 2011.

[SG06] Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. *Advances in Neural Information Processing Systems*, pages 1257–1264, 2006.

[SGR04] Edward Snelson, Zoubin Ghahramani, and Carl E. Rasmussen. Warped Gaussian processes. *Advances in Neural Information Processing Systems*, 2004.

[SKKS10] Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: no regret and experimental design. *International Conference on Machine Learning*, 2010.

[SLA12] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 2012.

[SSW$^+$15] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando De Freitas. Taking the human out of the loop: a review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[STJ05] Matthias Seeger, Yee-Whye Teh, and Michael I. Jordan. Semiparametric latent factor models. *Technical Report*, 2005.

[SWMW89] Jerome Sacks, William J. Welch, Toby J. Mitchell, and Henry P. Wynn. Design and analysis of computer experiments. *Statistical Science*, pages 409–423, 1989.

[Tho33] William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[TL10] Michalis Titsias and Neil D. Lawrence. Bayesian Gaussian process latent variable model. *International Conference on Artificial Intelligence and Statistics*, 2010.

[TLSS06] Ichiro Takeuchi, Quoc V. Le, Timothy D. Sears, and Alexander J. Smola. Nonparametric quantile estimation. *Journal of Machine Learning Research*, 7(Jul):1231–1264, 2006.

[UBC$^+$16] Doniyor Ulmasov, Caroline Baroukh, Benoit Chachuat, Marc P. Deisenroth, and Ruth Misener. Bayesian optimization with dimension scheduling: application to biological systems. *Computer Aided Chemical Engineering*, 2016.

[VGS$^+$20] Aki Vehtari, Andrew Gelman, Tuomas Sivula, Pasi Jylänki, Dustin Tran, Swupnil Sahai, Paul Blomstedt, John P Cunningham, David Schiminovich, and

Christian P Robert. Expectation propagation as a way of life: A framework for bayesian inference on partitioned data. *Journal of Machine Learning Research*, 21(17):1–53, 2020.

[VVW09] Julien Villemonteix, Emmanuel Vazquez, and Eric Walter. An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, 44(4):509, 2009.

[Wac13] Hans Wackernagel. *Multivariate Geostatistics: an Introduction with Applications*. Springer Science & Business Media, 2013.

[WF19] Jian Wu and Peter I. Frazier. Practical two-step lookahead Bayesian optimization. *Advances in Neural Information Processing Systems*, 2019.

[WHSX16] Andrew G. Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. *International Conference on Artificial Intelligence and Statistics*, 2016.

[Wil92] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, pages 196–202, 1992.

[WKG11] Andrew G. Wilson, David A. Knowles, and Zoubin Ghahramani. Gaussian process regression networks. *International Conference on Machine Learning*, 2011.

[WKVC09] Christopher Williams, Stefan Klanke, Sethu Vijayakumar, and Kian M. Chai. Multi-task gaussian process learning of robot inverse dynamics. *Advances in Neural Information Processing Systems*, 2009.

[WMHD17] James T. Wilson, Riccardo Moriconi, Frank Hutter, and Marc P. Deisenroth. The reparameterization trick for acquisition functions. *NIPS Workshop on Bayesian optimization*, 2017.

[WSD15] Niklas Wahlström, Thomas B. Schön, and Marc P. Deisenroth. From pixels to torques: policy learning with deep dynamical models. *arXiv:1502.02251*, 2015.

[WZH+13] Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, and Nando De Freitas. Bayesian optimization in high dimensions via random embeddings. *International Joint Conference on Artificial Intelligence*, 2013.

[ZBLN97] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.