Master's Thesis

Radboud University Nijmegen

The online optimization of brain-computer interface stimulus parameters, a simulation

---

Master's programme in Artificial Intelligence: Intelligent Technology

Author:
Chiara Thöni
chiara.thoeni@ru.nl

Supervisor:
dr. Michael Tangermann
michael.tangermann@donders.ru.nl

Second reader:
dr. Max Hinne
m.hinne@donders.ru.nl

August 20, 2023

Abstract

A brain-computer interface operates by presenting stimuli to elicit brain responses that can be acted upon. As the discriminability of these responses is subject to the stimulus characteristics, the presentation of the optimal stimuli could improve the performance of the interface. Unfortunately, these optima are subject-specific. Furthermore, the optimization of said stimuli is complicated by the potentially high level of non-independent and identically distributed noise that comes with the recorded responses. The aim of this thesis is to demonstrate that these challenges can be alleviated by introducing the optimization process with heteroskedatic modelling and the replication of existing designs. To this end, various Bayesian optimization algorithms are tested on simulations that are designed to resemble the aforementioned online optimization objective. The compared optimization algorithms differ in the surrogate models, replication schemes and selection strategies that have been used. As it is impossible to change the stimulus characteristics that have been used to record existing datasets, the simulated objective functions model the response discriminability as a function of the decoding parameters. To assess the effectiveness of the optimization algorithms, the algorithms are tested on one, two and seven-dimensional objective functions. The results suggest that a Bayesian optimization algorithm that is enriched with heteroskedastic Gaussian process regression and the location-based evaluation of existing designs significantly outperforms the random sampling baseline for all objective functions that have been considered. Even though the thesis is focused at brain-computer interfaces, application of the results is widespread since heteroskedastic noise can also be encountered with black-box optimization within other domains.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The information flow in the brain is characterized by electrical activity (Kolb et al., 2016, p. 111). This activity can be recorded and acted upon, for example by means of brain-computer interfaces (BCI) (Wolpaw, 2007). Applications of BCI include the control of (assistive) appliances, such as wheelchairs (Long et al., 2012) and prostheses (McFarland & Wolpaw, 2008), communication (Birbaumer et al., 1999) or language rehabilitation after a stroke (Musso et al., 2022). The interaction between the brain signals and the BCI application is co-established with brain signal recording devices. From these devices, electroencephalography (EEG) seems to be a suitable candidate for this recording task due to its non-invasive nature and high temporal resolution (Kolb et al., 2016, p. 111; Gui et al., 2010). Yet, EEG recordings suffer from non-stationarity and a low signal-to-noise ratio (SNR) (Gui et al., 2010; Sosulski et al., 2022). To classify, or regress on, the recorded signals, machine learning methods are trained on the subject-specific EEG recordings. In the case of classification, the subject of this thesis, the machine learning algorithms could aim to categorize the event-related potentials (ERPs) that can be elicited during a BCI paradigm. These type of brain responses can be classified based on their spatio-temporal patterns and amplitude (Sosulski et al., 2022). Research suggests that the characteristics of the stimuli that are used to elicit ERPs, such as the stimulation speed (Gonsalvez et al., 2007; Höhne & Tangermann, 2012; Sugi et al., 2018), stimulation intensity (Gonsalvez et al., 2007) and stimulation pattern (Allison & Pineda, 2006), influence the single-trial ERP discriminability. While the drawbacks that come with the recorded EEG signals make the decoding task non-trivial, the characteristics of the stimuli that are used to elicit the ERPs could be exploited to assist in this task.

Whereas the "optimal" stimulation parameters can be taken from literature, research has been conducted into the automatic, subject-specific optimization of the stimulus characteristics (Sosulski et al., 2022). To reflect the aim of the optimization, the objective function could be modelled after a measurement of the ERP discriminability, such as the amplitude of the potential or the classification score (Sosulski et al., 2022). Yet, such an objective function can be considered a *black box*: there is no available analytical solution for the optimum or information about the gradient. Due to the absence of this information, the search for the global optimum is restricted to making queries to the objective function for different parameter values (Brochu et al., 2010; Frazier, 2018). When the optimization of stimulus characteristics is considered, the objective function is queried by performing a trial of a BCI experiment with the stimuli of interest (Sosulski et al., 2022). The execution of the aforementioned trials is considered an expensive operation due to their time-consuming nature. As a consequence, the number of samples that can be evaluated during the optimization process is limited, which renders brute-force techniques such as grid search inapplicable. Additionally, the outcome that results from the evaluation of the objective function can be noisy (Brochu et al., 2010; Sosulski et al., 2022), which could make the greedily selected optimum unreliable. For these reasons, better-informed techniques that are applicable to black-box objective functions are applied. A popular technique for solving such

expensive, black-box optimization problems, which has also been used by Sosulski et al. (2022), is Bayesian optimization (BO) (Brochu et al., 2010; Frazier, 2018; Jones et al., 1998; Snoek et al., 2012).

Bayesian optimization algorithms consist at least of two components. The first component is a Bayesian statistical model that is used to construct a belief about the shape of the objective function. Next, the algorithm can make suggestions for informed samples by striking a balance between exploration and exploitation of the constructed representation. The exact tradeoff between exploration and exploitation is determined by the second component of the algorithm, the acquisition function (Frazier, 2018; Lam et al., 2016).

From the definition above, it becomes clear that the performance of the Bayesian optimization algorithms depends on the quality of the representation of the objective function. More specifically, the algorithm's functioning is hampered when the statistical model fails to sufficiently capture the objective function. The latter could happen if the observational noise is substantially larger than the variance of the objective function or the assumptions of the Bayesian statistical model are violated (Sosulski et al., 2022). When EEG recordings are considered, both causes could occur. Firstly, an objective function that is based on the discriminability of the ERP is a noisy estimator due to the low SNR within the recorded EEG signals. Secondly, the non-stationarities within the recorded data prevent the data (and noise) from being independent and identically distributed (iid), which makes the application of some Bayesian statistical models, such as Gaussian processes, suboptimal (Williams & Rasmussen, 2006, p. 8). Lastly, depending on the nature of the objective function, the distribution of data and the associated noise could be skewed due to ceiling effects (e.g., if the AUC $\in [0, 1]$ is used as an objective function), which hampers the performance of statistical models that model the data and associated noise as a Gaussian distribution (Sosulski et al., 2022).

The aim of this thesis is to investigate how the challenges outlined above could be addressed to improve the online optimization strategy of BCI stimulus parameters that has been proposed by Sosulski et al. (2022). Ultimately, this could lead to a more efficient interaction with the BCI systems. Yet, the presence of input-dependent noise, and the challenges that come with it, are not limited to the domain of BCI. Other fields where heteroskedastic noise can be encountered include geostatistical modelling, econometrics, statistical finance, machine learning and robotics (Kersting et al., 2007; Lázaro-Gredilla & Titsias, 2011; Smith et al., 2008). If the data from these fields are combined with black-box optimization, e.g. for the optimization of machine learning hyperparameters as done in automated machine learning (AutoML) (Kurian et al., 2021), then optimization algorithms that can model heteroskedastic noise are of interest. Thus, the research question is phrased in a general fashion to emphasise that the application of the outcomes of this thesis could go beyond the BCI domain.

1.1.1 Research question

> How can optimization algorithms be made resilient against the high level of potentially heteroskedastic noise?

As the research question is considered in a BCI context, this thesis is aimed at improving the classification accuracy of a BCI experiment. To that end, various optimization strategies are tested on simulated BCI data. The simulations are based on existing datasets. Because the data have already been recorded, it is impossible to change the characteristics of the stimuli. Instead, the objective functions that are simulated are modelled as a function of the parameters that describe the decoding and classification process of the recorded data.

1.1.2 Hypotheses

The approach that has been proposed by Sosulski et al. (2022) could possibly benefit from the use of a Bayesian statistical model that is better able to handle the non-stationary nature of the recorded signals. To that end, Bayesian statistical models such as heteroskedastic Gaussian processes or random forests that are equipped with a posterior distribution could be used (Goldberg et al., 1997; Hutter et al., 2014; Kersting et al., 2007; Lázaro-Gredilla & Titsias, 2011; Le et al., 2005). In this context, heteroskedastic modelling has got two advantages over homoskedastic modelling. Firstly, the heteroskedastic model is expected to capture the noise more appropriately, resulting in a more accurate posterior mean. Secondly, the acquisition function of the Bayesian optimization algorithm could be enhanced by adding information about the input-specific (uncertainty of the) noise. Additionally, the quality of the noise-estimate of the statistical model could be improved by replicating samples, where the repeated evaluation of a location is expected to yield insights in the local noise structure. If the aforementioned improvements allow the Bayesian optimization process to better represent heteroskedastic non-Gaussian noise, then the improved optimization algorithm is more likely to find the optimal stimulus parameters, which leads to a better task-discriminability and classification performance.

To put the performance of the optimization algorithm into perspective, it will be compared to the performance of a random search method. Given this baseline, it is hypothesized that the Bayesian optimization approach will outperform the random search method when multiple parameters are simultaneously optimized. The increased volume of the higher dimensional parameter space is likely to hamper the performance of the uninformed algorithm, which is forced to explore the problem space without any heuristics.

In summary, the different hypotheses are:

H_1: Bayesian optimization with heteroskedastic surrogate models will perform better on a simulated BCI experiment than random search.

H_2: Bayesian optimization with heteroskedastic surrogate models will perform better on a simulated BCI experiment than Bayesian optimization with homoskedastic surrogate models when the noise around the objective function is heteroskedastic.

H_3: The performance difference between Bayesian optimization and random search increases if the dimensionality of the simulated objective function increases.

H_4: Bayesian optimization strategies that replicate existing designs have a better performance than Bayesian optimization strategies that do not make replications.

Given the different hypotheses, the answer to the research question is multi-faceted. I am not merely interested in the performance of the Bayesian optimization algorithm down the line; I also wish to access the algorithm's proficiency in representing the unknown objective function and associated noise. To this end, different optimization problems with varying degrees of noise are simulated. The details of this process are outlined in the next section.

1.2 Thesis outline

Throughout the thesis, I will assume that the reader is familiar with the basics of brain-computer interfaces, linear algebra and (statistical) machine learning. However, an in-depth knowledge of the aforementioned topics is not required. All concepts that are used will be introduced in Chapter 2, which discusses the preliminaries that form the basis for this thesis. Next, Chapter 3 provides an analysis of the problem domain by highlighting the existing work that relates to this thesis. Additionally, this chapter is aimed at identifying the areas where the existing methodology could be improved. An explanation of the newly proposed methods is presented

in Chapter 4, whereas the results of said methods are discussed in Chapter 5. Lastly, the conclusions and suggestions for further work are considered respectively in Chapters 6 and 7.

Code

The code that has been written to implement the methods of this thesis can be found on GitHub:

https://github.com/Racemuis/OnlineOptimization

This repository also includes the scripts that have been used to generate the figures that are included in this thesis.

Notational conventions

In the mathematical formulae that are used in this work, I follow the conventions described in the table below.

Mathematical element	*Typeset*
Scalar values	lowercase letters
Vectors	lowercase, bold letters
Matrices	uppercase, bold letters
Set sizes	uppercase Latin alphabet

Table 1.1: The mathematical conventions used throughout this thesis.

Typically, additional information regarding a parameter is added in the subscript.

Throughout this thesis, I aim to use similar notations for reoccurring concepts. The concepts that often appear within this thesis are presented in the table below for reference.

Concept	*General notation*	*Evaluated at x*
The objective function	f	$f(x)$
The noisy measurements of the objective function	y	$y(x)$
The homoskedastic objective noise	σ^2_{noise}	-
The heteroskedastic objective noise	r	$r(x)$
The posterior mean of a Bayesian model	m	$m(x)$
The posterior variance of a Bayesian model	σ^2	$\sigma^2(x)$
The modelling uncertainty	$\breve{\sigma}^2$	$\breve{\sigma}^2(x)$
The location that maximizes the acquisition function	x^*_{acqf}	-
The parameter regulating the degree of shrinkage regularization	γ	-

Table 1.2: Concepts that often occur in this thesis. Some concepts represent functions that can be evaluated at any location x, as presented in the last column of the table.

Furthermore, there are some concepts that relate directly to the optimization algorithm. These parameters are presented in Table 1.3.

Concept	*Notation*
The total number of iterations	N
The number of random/initialization samples	N_{random}
The number of informed samples	$N_{informed}$
The parameter regulating the exploration-exploitation tradeoff	β
The parameter regulating the convergence measure	τ

Table 1.3: The parameters of the Bayesian optimization algorithm.

Chapter 2

Preliminaries

Purpose of the chapter

This chapter aims to provide the reader with the theoretical basis that is relevant for the remainder of the thesis. To this end, Section 2.1 describes the neurophysiological background of the thesis with the discussion of event-related potentials. A more technical background is presented in Section 2.2, which comprises of an introduction to Bayesian optimization.

2.1 Event-related potentials

EEG is a recording technique that measures the sum of the action potentials of thousands of neurons (Kolb et al., 2016, p. 223). While the recorded neuronal activity tends to fluctuate with a change in behaviour, it is also possible to elicit complex electroencephalographic waveforms with the presentation of sensory stimuli (Kolb et al., 2016, p. 224). These waveforms are also known as ERPs. Even though ERPs are time-locked to the stimulus, they are not easy to detect due to the presence of other electrical signals in the brain that dilute the waves of interest. As a consequence, ERPs are often made visible by averaging over multiple responses to the same stimulus, an operation that tends to cancel out the irregular waveforms that are not associated with the stimulus (Kolb et al., 2016, p. 224). The ERP nomenclature is based on the sign of recorded waves' amplitude and their latency with respect to the stimulus onset (Kolb et al., 2016, p. 224). Figure 2.1 shows an example of the so-called N200 and P300 responses that have been recorded with the channels Cz (solid) and FC1 (dashed) during an auditory ERP paradigm (Denzer, 2016). The details of the recorded data are discussed in more detail in Section 4.1.

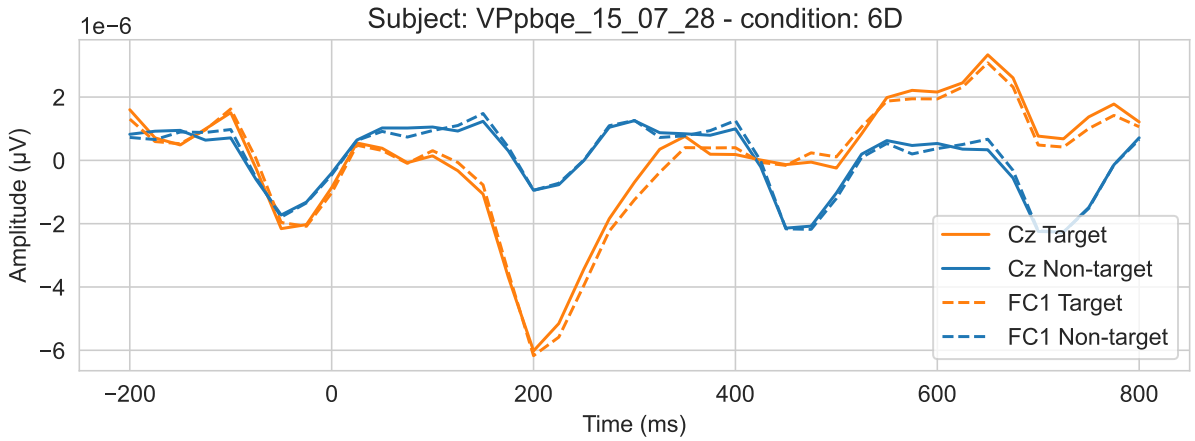


Figure 2.1: The target (orange) and non-target (blue) responses that have been recorded during an auditory ERP paradigm (Denzer, 2016). The visualized channels are Cz (solid) and FC1 (dashed). The recorded target signals demonstrate a N200 response at 210 ms with an amplitude around $-6\mu V$. Furthermore, a P300 response with an amplitude of around $3\mu V$ can be discerned around 660 ms (Denzer, 2016). The waveforms have been visualized by averaging over 540 target, and 2700 non-target responses.

The responses represent a negative (N) and a positive (P) peak, respectively approximately 200 and 300ms after the stimulus onset (0ms) (Denzer, 2016). Therefore, the presence of these responses is indicative of the stimulus that has been presented. The ERPs only occur when the target stimulus is presented. In the figure, the presentation of a target stimulus is indicated with "target" (orange), whereas the control condition is referred to as "non-target" (blue). The larger the amplitude of the recorded ERPs, the easier it becomes to tell the two conditions apart.

When a single epoch is considered, it is not assumed that the noise that comes with the recorded signal is specific to a time point. Thus, the noise that has been averaged out to create Figure 2.1 can be seen as homoskedastic. Yet, the latter assumption is invalid when an entire experiment, which consists of multiple epochs, is regarded. Due to non-stationarities such as habituation or fatigue that can occur during an experiment, the noise that can be found at later epochs could differ from the noise that is recorded early on. Though, it is important to note that the heteroskedastic noise introduced by non-stationarities is specific to the duration of the experiment. As a result, the non-stationary noise is only input-specific to the objective function if the duration is optimized.

2.2 Bayesian optimization

Bayesian optimization is a class of optimization algorithms that can be used to find the global optimum of black-box functions: functions f, for which there is no analytical solution for their optimum or any information available about their gradient (Frazier, 2018). Without this knowledge, the algorithm is constrained to querying informed samples from the objective f. As the evaluations are often considered expensive, it is key to do so in the most effective and efficient manner. Note that the outcomes $f(x)$ that are associated with queried locations x they are not necessarily noiseless. Therefore, a notational distinction is made between noiseless, $f(x)$, and noisy, $y(x)$, outcomes.

(The precursor of) Bayesian optimization has been introduced in 1964 by Kushner to replace quasi-Newton methods for objective functions for which it is impractical to approximate the derivative (Kushner, 1964). Since then, Bayesian optimization has been described in various works (e.g. (Booker et al., 1999; J. B. Močkus & Močkus, 1991; J. Močkus, 1975; Zilinskas, 1978)) before receiving a surge in popularity in response to the work by Jones et al. (Jones et al., 1998). Later, Snoek et al. provided the algorithm with even more recognition by demonstrating its applicability to the optimization of hyperparameters for deep learning models (Snoek et al., 2012). In all applications described above, Bayesian optimization has been employed due to its performance when the sampling budget is limited to a few hundreds of samples (Frazier, 2018).

Bayesian optimization aims to deal with this limited budget by sampling locations x such that the observed outcome, $y(x)$, provides as much information as possible about the shape of f. To achieve this goal, Bayesian optimization methods construct a surrogate of the objective function that helps the optimization method to decide where to evaluate the true objective function (Booker et al., 1999; Kushner, 1964). For that purpose, the algorithm is typically equipped with two components: a statistical model and an acquisition function. Together, these components operate in an iterative manner to strategically sample from the objective function. This process, which is limited by a budget of N evaluations, is described in more detail in Algorithm 1.

In Algorithm 1, the statistical regression model h can be evaluated at any location x to provide an estimate of $f(x)$. Yet, before the statistical can be queried, the model needs to be initialized with n_{random} initialization samples. Next, the algorithm makes $n_{informed}$ iterations where h is used to make an assessment of f at any input value x. Whilst this estimation of the *value* of $f(x)$ can be used by the acquisition function to decide where to sample next, it is also infor-

mative to know the *uncertainty* around that estimation. To facilitate the latter, the statistical model is required to describe the posterior probability distribution around the estimate of $f(x)$ (Kushner, 1964). In the next section, two different models that come with an uncertainty distribution around the modelled estimate are discussed: Gaussian process regression and random forest regression.

Algorithm 1 A basic Bayesian optimization algorithm

Require: $N > 0$, $n_{random} + n_{informed} = N$
1: Initialize the statistical model, h, as a prior on f
2: Evaluate $f(x)$ for $n_{random} \geq 0$ initial locations $x_i \in \boldsymbol{x}_0 : \{x_0,\ ...,\ x_n\}$, yielding the set $\boldsymbol{y}_0 : \{y(x_0),\ ...,\ y(x_n)\}$
3: $n \leftarrow 0$
4: $y \leftarrow \boldsymbol{y}_0$
5: $x \leftarrow \boldsymbol{x}_0$
6: **while** $n \leq n_{informed}$ **do**
7: Update h given all existing sample locations $\boldsymbol{x}$ and associated evaluations $\boldsymbol{y}$
8: Find x^*_{acqf} that maximizes the acquisition function given the updated h
9: Update the set of existing locations: $\boldsymbol{x} \leftarrow \boldsymbol{x} \cup \{x^*_{acqf}\}$
10: Update the set of existing evaluations: $\boldsymbol{y} \leftarrow \boldsymbol{y} \cup \{y(x^*_{acqf})\}$
11: Increment n
12: **end while**
13: Select $\hat{x}^*$ that is presumed to maximize f

2.2.1 (Bayesian) statistical models

Gaussian process regression

Gaussian processes can be used to describe a distribution over functions (Williams & Rasmussen, 2006, p. 13). To consider the Gaussian process in more detail, let $\boldsymbol{x} \in \mathbb{R}^n$ be the n one-dimensional training data points $\{x_1,\ ...,\ x_n\}$ with the associated, noise-free outcomes $\mathbf{f} \in \mathbb{R}^n$. Then, a Gaussian process can more formally be described as a finite collection of normally distributed random variables that is defined by a mean function

$$m(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})] \tag{2.1}$$

and covariance $\mathbf{K}$ over the training input $\boldsymbol{x}$. Here, $\mathbf{K} \in \mathbb{R}^{n \times n}$ is a Gram matrix whereof the elements i and j are defined as:

$$\mathbf{K}_{ij} = k(x_i,\ x_j) \tag{2.2}$$

for each pair $(x_i,\ x_j) \in \{\boldsymbol{x} \times \boldsymbol{x}\}$ where "$\times$" denotes the Cartesian product. In the remainder of this work, the notation $K(\boldsymbol{x}, \boldsymbol{x'})$ is used to indicate the construction of a covariance matrix between the vectors $\boldsymbol{x}$ and $\boldsymbol{x'}$.

In Equation 2.2, $k(x,\ x')$ is the kernel or covariance function that specifies the similarities between the elements x and x' (Bishop & Nasrabadi, 2006, p. 305). To be a valid kernel, the function needs to be symmetric, i.e. $k(x,\ x') = k(x',\ x)$ (Williams & Rasmussen, 2006, p. 80). Furthermore, the kernel should guarantee that the resulting covariance matrix is positive semi-definite (Frazier, 2018). Below, an example is provided to illustrate the workings of a Gaussian process. In the example, the Matérn kernel (Equation 2.3) is used to construct $\mathbf{K}$.

$$k(x, x') = \frac{2^{\nu-1}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}}{\ell} d(x, x') \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}}{\ell} d(x, x') \right) \tag{2.3}$$

In the equation, $\Gamma(\cdot)$ is the Gamma function and $K_\nu(\cdot)$ is the modified Bessel function (Williams & Rasmussen, 2006, p. 83). The degree of covariance between two elements x and x' depends on the distance $d(\cdot, \cdot)$ between them (Bishop & Nasrabadi, 2006, p. 299). In particular, the covariance of x and x' increases when $d(x, x')$ decreases. Furthermore, the covariance is regulated by the positive parameters ℓ, which denotes the length-scale of the kernel, and ν. Both parameters can be considered regulators of the smoothness of the kernel: the larger these parameter values, the smoother the kernel. ν is typically set to a half-integer value as this makes the function easier to differentiate (Williams & Rasmussen, 2006, p. 85). In the figure below, the covariance matrices for a Matérn $5/2$ kernel are shown (i.e. $\nu = 5/2$). The covariance is maximal on the main diagonal of the covariance matrices. It can be seen that, with larger length scales, the covariance drops less sharply for an increasing distance.

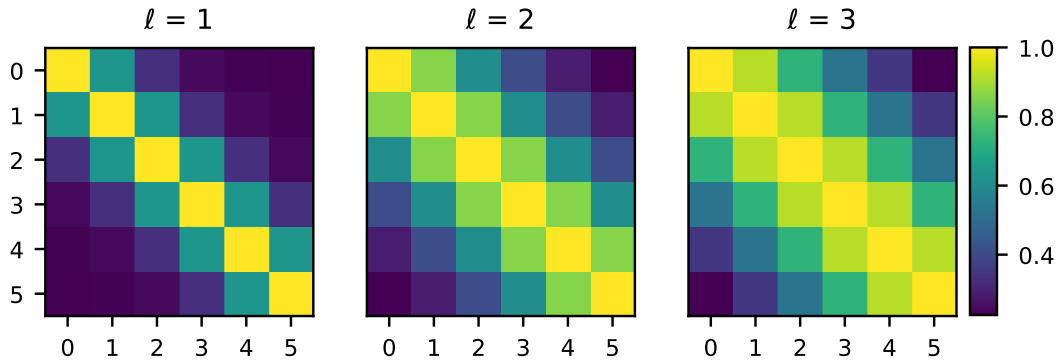


Figure 2.2: Covariance matrices $K(\boldsymbol{x}, \boldsymbol{x})$ for $\boldsymbol{x} = \{1, 2, 3, 4, 5\}$ as calculated with the Matérn $5/2$ kernel with different lengthscales $\ell \in \{1, 2, 3\}$. When comparing the three panes from left to right, one can verify that the covariance drops less sharply with an increasing distance if the value of ℓ is large. This is because, larger length scales allow for a larger covariance between samples that are distant from each other.

In the majority of the literature, the mean of the Gaussian process is set to $\mathbf{0}$ for notational reasons, as the process can entirely be governed by the covariance matrix $\mathbf{K} = K(\boldsymbol{x}, \boldsymbol{x})$ (Williams & Rasmussen, 2006, p. 14). Using this specification, vectors $\mathbf{f}_{prior} \in \mathbb{R}^n$ can be drawn from the distribution that is described by the Gaussian process (Williams & Rasmussen, 2006, p. 14),

$$\mathbf{f}_{prior} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}) \tag{2.4}$$

If the outputted values $\mathbf{f}_{prior}$ are plotted as a function of the inputs $\boldsymbol{x}$, then the results would, depending on the length scale of the kernel, look similar to the curves shown in Figure 2.3. In each of the three panes in the figure, ten samples from the distribution described in Equation 2.4 are shown. As this distribution in this equation is merely based on the input values $\boldsymbol{x}$ of the training data, the generated distribution can be considered the prior of the Gaussian process. When comparing prior samples within the three panes from Figure 2.4, we can see that the differences between the covariance matrices from Figure 2.2 are reflected in the shape of the prior samples: whereas the samples in the leftmost pane ($\ell = 1$) can be considered relatively "jittery", the samples become more smooth when the length scale increases.

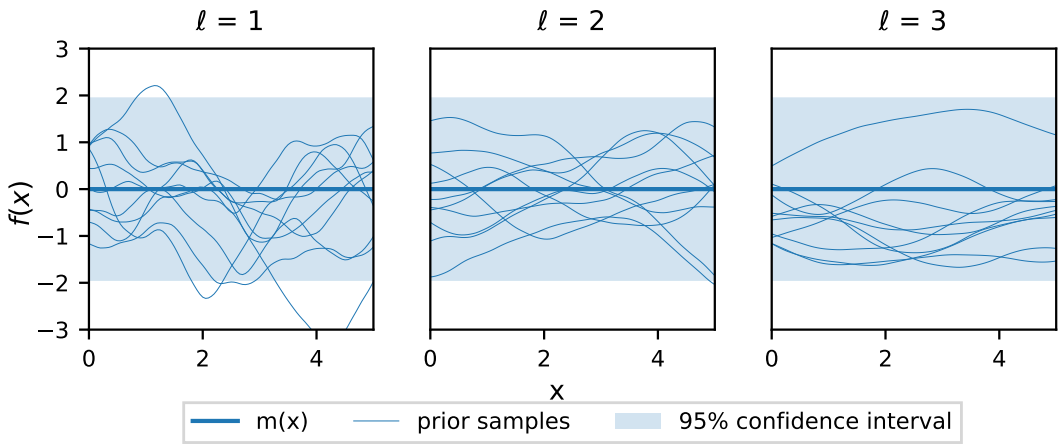


Figure 2.3: Ten random samples from $\mathcal{N}(\mathbf{0}, K(\boldsymbol{x}, \boldsymbol{x}))$ for $\boldsymbol{x} = \{0.05, 0.1, ..., 5.00\}$ where $K(\boldsymbol{x}, \boldsymbol{x})$ is parameterized with the different length scales $\ell \in \{1, 2, 3\}$.

Since the prior distribution of the Gaussian process does not include the information that is provided by the training outcomes $\mathbf{f}$, it is not particularly interesting to sample from, or make predictions with, this distribution. Therefore, the joint distribution of the training outcomes $\mathbf{f}$ and test outcomes $\mathbf{f}_*$ is constructed as shown in Equation 2.5(Williams & Rasmussen, 2006, p. 15).

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_* & \mathbf{K}_{**} \end{bmatrix}\right) \tag{2.5}$$

The covariance matrix of the joint distribution is composed out of Gram matrices whose inner product spaces are defined by the Cartesian product of the training inputs ($\mathbf{K} = K(\boldsymbol{x}, \boldsymbol{x})$), the training inputs and merely the test inputs ($\mathbf{K}_* = K(\boldsymbol{x}, \boldsymbol{x_*})$) and the test inputs ($\mathbf{K}_{**} = K(\boldsymbol{x_*}, \boldsymbol{x_*})$). Note that the training outcomes $\mathbf{f}$ are still not involved in the construction of the covariance matrix of the joint prior distribution. Instead, they are contained in the matrix on the left-hand side of the equation. While the distribution defined above can be used to obtain the posterior distribution by means of rejection sampling[1], this process is not efficient (Williams & Rasmussen, 2006, p. 16). Rather, the joint distribution is conditioned on $\mathbf{f}$ by exploiting the identities of the (partitioned) multivariate Gaussian distribution (Bishop & Nasrabadi, 2006, p. 89) to yield

$$\mathbf{f}_* | \boldsymbol{x},\ \boldsymbol{x}_*,\ \mathbf{f} \sim \mathcal{N}\left(\mathbf{k}_* \mathbf{K}^{-1} \mathbf{f}, \mathbf{K}_{**} - \mathbf{K}_* \mathbf{K}^{-1} \mathbf{K}_*\right) \tag{2.6}$$

where $\boldsymbol{x}_*$ represents the input values of the test data (Bishop & Nasrabadi, 2006, p. 308).

To illustrate the distribution described above, the prior distribution from Figure 2.3 with $\ell = 1.0$ has been conditioned on five data points that have been uniformly sampled from $f(x) = \sin(x)$. The mean and the 95% confidence interval of resulting posterior are shown in Figure 2.4. As the training data are assumed to be noise-free, the variance of the posterior distribution merely depends on the spacing of the input data, not on the outcomes (Binois et al., 2019). More specifically, the variance of the posterior distribution at the observed data points is 0, whereas the posterior variance is large within regions that do not contain any observed data points.

[1]This could be done by sampling values from the joint prior distribution while discarding all samples that do not agree with the training outcomes $\mathbf{f}$.

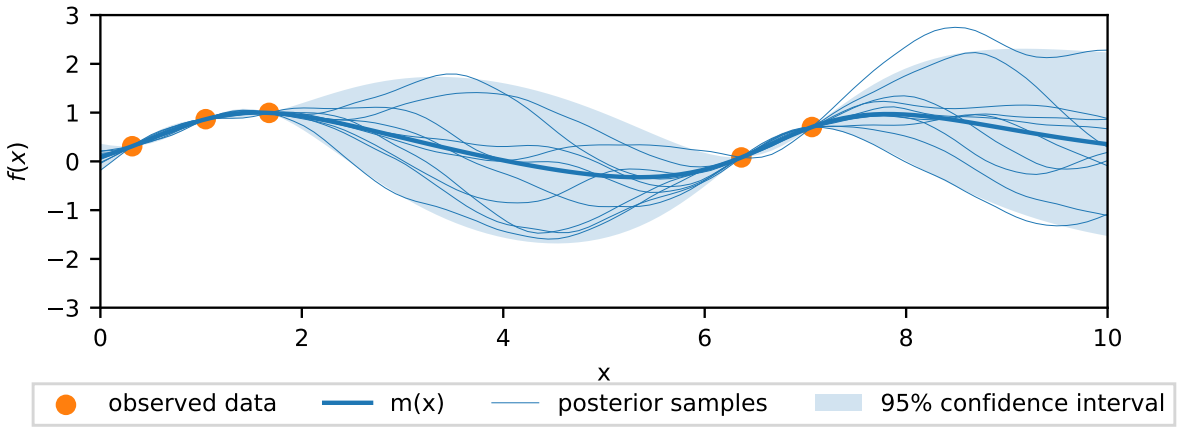


Figure 2.4: Ten samples drawn from the posterior distribution of the Gaussian process, which is the joint prior distribution that has been conditioned on the observed data points (orange). The shaded area indicates the 95% confidence interval of the posterior distribution.

While the posterior distribution shown above provides a proof of concept, it is not realistic to assume that the noise-free objective $f(x)$ is available in all scenarios. Fortunately, observational noise could be accounted for by refining the definition of the covariance matrix $\mathbf{K}$. More specifically, if the observational noise ε is assumed to be iid according to a zero-mean Gaussian distribution with variance σ^2_{noise}, then the observational noise can be incorporated in the Gaussian process prior by adding σ^2_{noise} to each entry on the main diagonal of $\mathbf{K}$ (Williams & Rasmussen, 2006, p. 16):

$$\mathbf{K}_y = \mathbf{K} + I\sigma^2_{noise} \tag{2.7}$$

As the entries on the main diagonal of the covariance matrix represent the variance per data point, $\mathbf{K}_y$ describes a covariance matrix for which the noise for all data points is assumed to be equal and independent, as is implied by the iid assumption. The mean and confidence interval from the posterior distribution of a Gaussian process that assumes observational noise is shown in Figure 2.5. Due to the presence of $\sigma^2_{noise} = 0.1$ in the covariance, the variance of the posterior distribution at the observed data points is no longer equal to 0.

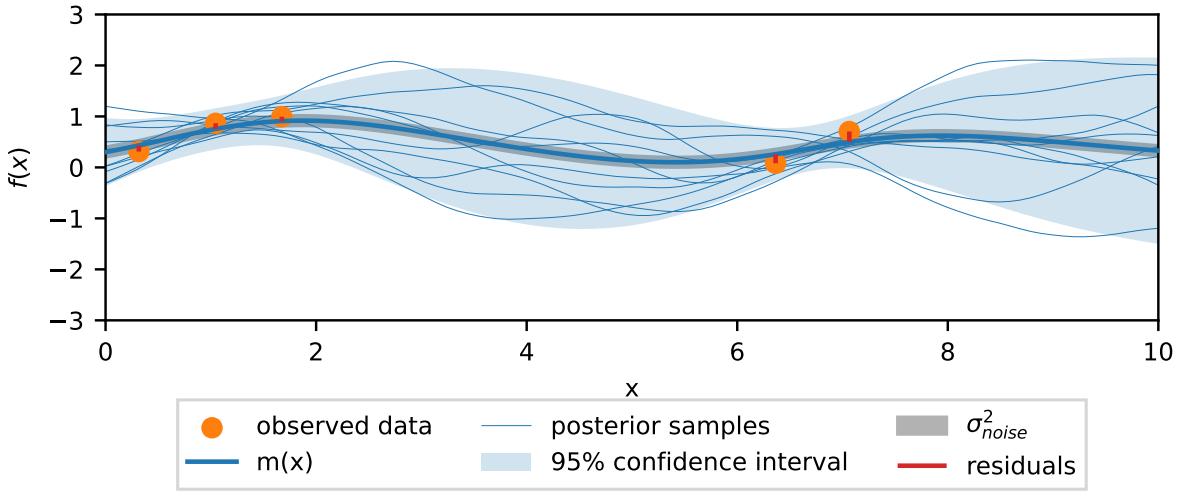


Figure 2.5: Ten samples drawn from the posterior distribution of the Gaussian process that has been fit with $\sigma^2_{noise} = 0.1$. The vertical red lines highlight the distance between the observed data points and the posterior mean of the Gaussian process. In addition to the 95% confidence interval that has been plotted in blue, the assumed variance of the noise process σ^2_{noise} is indicated with the shaded grey area.

Still, the solution proposed in Equation 2.7 would be an oversimplification if the observational noise is heteroskedastic, meaning that variance of the noise around f depends on the input location x. In that case, it would be more sensible to tailor the estimation of the observational noise to each input by making the noise term σ^2_{noise} a function of the input data. Accordingly, each element of the covariance matrix $\mathbf{K_y}$ is defined as follows (Goldberg et al., 1997).

$$\mathbf{K}_y = \mathbf{K} + I\boldsymbol{r}(\boldsymbol{x}) \tag{2.8}$$

In the equation above, $\boldsymbol{r}(\boldsymbol{x}) = (r(x_1), \ldots, r(x_n))$ is a vector containing the variance of the noise at each input value $x_i \in \boldsymbol{x}$.

To estimate the variance of the noise process and the kernel hyperparameters (such as the length scale in Equation 2.2), one can use Bayes' rule to calculate the posterior distribution over these parameters, as demonstrated by Williams and Rasmussen (Williams & Rasmussen, 2006, p. 109, Equation 5.5). As the utilization of Bayesian inference for this purpose involves the evaluation of (analytically intractable) integrals, Markov Chain Monte Carlo (MCMC) methods are often used to approximate the posterior (Williams & Rasmussen, 2006, p. 109). Note that these methods are inherently very slow when large data sets are used (Lázaro-Gredilla & Titsias, 2011). To decrease the runtime of the algorithm, the Gaussian process can be fitted to the training data by maximizing the marginal (log-)likelihood with respect to the hyperparameters (Williams & Rasmussen, 2006, p. 109). While being less computationally heavy than the Bayesian approaches that involve MCMC, this approximation is more prone to overfitting (Williams & Rasmussen, 2006, p. 109).

Both approaches described above are applicable if the noise is assumed to be iid. However, the marginal likelihood is not analytically tractable when heteroskedastic noise (Equation 2.8) is considered (Lázaro-Gredilla & Titsias, 2011). For that reason, Lazaro et al. have proposed a variational inference method to approximate the marginal log-likelihood (Lázaro-Gredilla & Titsias, 2011). Alternatively, the variance of the heteroskedastic noise $\boldsymbol{r}(\boldsymbol{x})$ can be estimated by training a second, homoskedastic Gaussian process, as proposed by Goldberg et al. in (Goldberg et al., 1997). Whereas Goldberg et al. sample the posterior of the noise distribution using MCMC, the MAP can also be used to approximate the hyperparameters of the second Gaussian process, as demonstrated by Kersting et al. for ordinary regression and Quadrianto et al. for quantile regression (Kersting et al., 2007; Quadrianto et al., 2009).

The MAP-based approaches are faster than the approaches that are based on MCMC. To make the generation of proposals as fast as possible, which would be favourable if the sampling budget is limited by a specified time, Gaussian process framework as proposed by Kersting et al. has been used in this work. The work by Kersting et al. will be discussed in more detail below to provide the reader with a sufficient background before diving into the methods.

In their work, Kersting et al. propose an expectation-maximization-like algorithm to learn an estimate of the input-specific noise $\boldsymbol{r}(\boldsymbol{x})$ (Kersting et al., 2007). To do so, they train (at least) three distinct Gaussian processes. The first Gaussian process, G_1, is a homoskedastic one that is trained on the data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$. As shown in Figure 2.5, the posterior mean of the Gaussian process does not necessarily have to traverse all data points. Instead, there might be a distance between the data points and the posterior mean: $|\mu(x_i) - y_i|$, as indicated in red in Figure 2.5. With this difference, Kersting et al. create an estimate of the noise levels, z_i, which is set to the variance within the distances between the observed labels y_i and s samples from the posterior distribution, y_i^j, where $j \in \{1, \ldots, s\}$ (Kersting et al., 2007):

$$z_i = var\left(y_i, G_1(x_i, \mathcal{D})\right) \approx s^{-1} \sum_{j=1}^{s} 0.5 \cdot (y_i - y_i^j)^2 \tag{2.9}$$

The estimates z_i are used as input data $\mathcal{D} = \{(x_i, z_i))\}_{i=1}^{n}$ to fit the second Gaussian process, G_2. On its turn, G_2 is used to predict levels $\boldsymbol{r(x)}$, that, in combination with the original data $\mathcal{D}$, are used to train a third Gaussian process, G_3 (Kersting et al., 2007). An example of a posterior distribution that can be obtained using this algorithm is shown in Figure 2.6. Whilst the distribution is comparable to the one shown in Figure 2.5, the shaded gray area, that now indicates the noise levels $\boldsymbol{r(x)}$, is now an estimate of heteroskedastic noise.

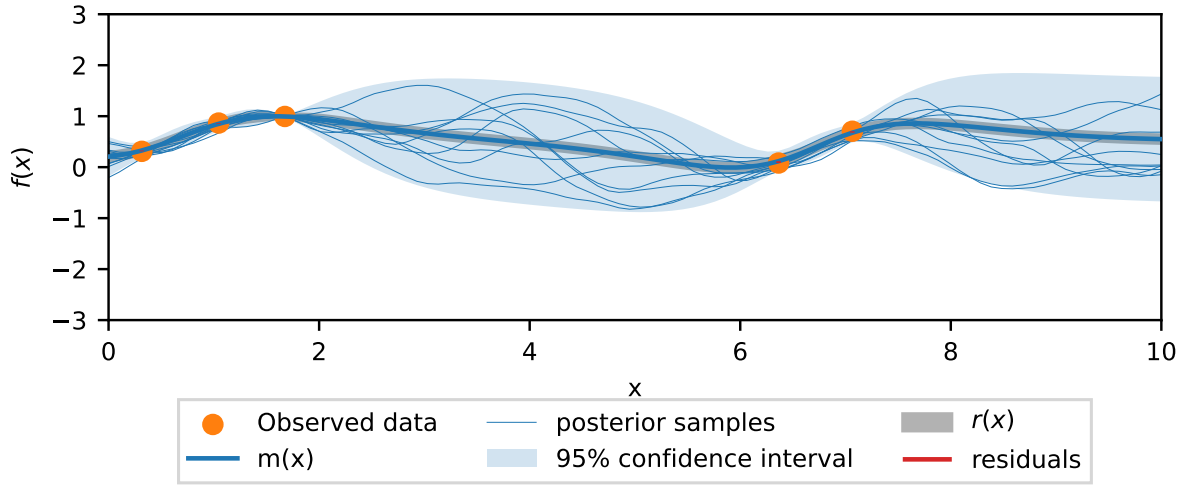


Figure 2.6: Ten samples drawn from the posterior distribution of the *Most Likely Heteroskedastic Gaussian Process Regression* algorithm that has been proposed by Kersting et al. (2007). The vertical red lines highlight the distance between the observed data points and the posterior mean of the Gaussian process. In addition to the 95% confidence interval that has been plotted in blue, the assumed variance of the heteroskedastic noise process $\boldsymbol{r(x)}$ is indicated with the shaded grey area.

According to the authors, the process described above can be repeated until convergence where after the last step, G_1 is set to G_3. Though, the authors also state that repeating the process is not guaranteed to improve the likelihood of G_3 given the data (Kersting et al., 2007).

Random forest regression

A random forest is an ensemble method that can solve machine learning tasks by constructing a collection of decision trees (Breiman, 2001). To ensure that the forest does not consist out of identical trees, each individual decision tree is fit on a subset of the training data that has been sampled with replacement, a procedure that is also known as bootstrapping (Breiman, 1996). The decision trees are created in a top-down fashion: starting from the root node, the training data are repeatedly split into smaller subsets. At each split, the algorithm aims to repartition the data in such way that the resulting subsets are as homogeneous as possible (Breiman, 1984, p. 93). The data is not split to the point where each leaf node merely contains data points that belong to a single class, as this comes with a high risk of overfitting. Rather, the splitting procedure is ceased when either a specified depth has been reached, the size of each subgroup has been reduced to a predefined minimum or when making more splits does not improve the purity of the resulting partitions (Therneau & Atkinson, 1997). A decision tree can make a prediction by navigating a new input down the tree. Once the test input has reached a leaf node, the final prediction is made by calculating the average of the labels of all training outcomes that are associated with the same leaf node.

A simple example of a single decision tree is shown in Figure 2.7. A one-dimensional input x traverses through the tree based on the value of its (only) attribute. The training outcomes

are stored within each associated leaf to make the final prediction. For example, for an input $x = 10$, which ends up in leaf 3, the tree predicts an outcome of $y_{predicted} = \frac{1+3+8}{3} = 4$.

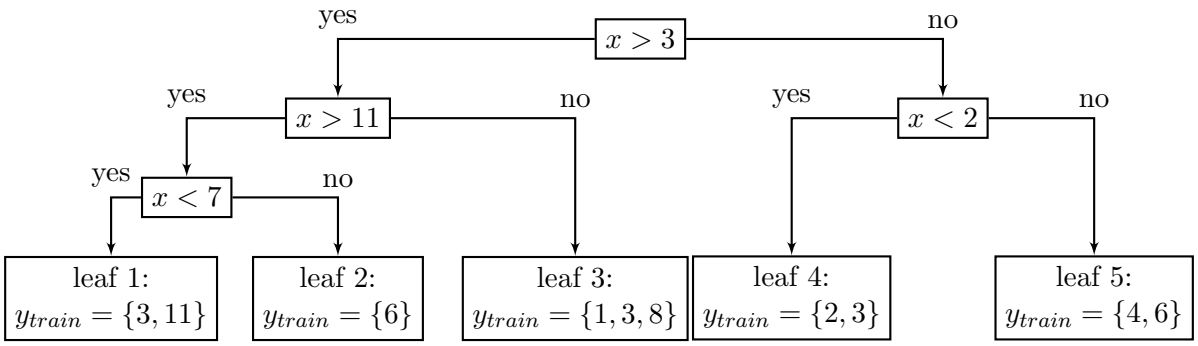


Figure 2.7: An example of a regression tree that is fit on a one-dimensional variable x. In each leaf of the tree, the associated training outcomes are stored. The tree's prediction for an input x is equal to the mean of the training outcomes of the leaf whereto x is allocated.

When grown to a sufficient depth, decision trees are able to capture complex patterns within the data, providing them with low bias at the cost of a high variance (Hutter et al., 2014). To reduce the latter, random forests aggregate over the predictions of all individual trees. To this end, the final output boils down to the average of all individual predictions (Hutter et al., 2014).

The standard random forest framework, as described above, does not provide a measure of uncertainty about its prediction. The absence of a predictive distribution makes the random forest as is a suboptimal choice for the surrogate function for the Bayesian optimization. Instead, the random forest could be modified to provide a quantification of the predictive uncertainty of the model. In (Hutter et al., 2014), the authors propose to update the definition of a random forest so that it is able to provide both a predictive mean (as did the original random forest) and variance. To provide this predictive variance, one needs to combine the two different types of empirical variances that can be found within the forest:

1) the variance that is associated with the training data within each leaf of each regression tree; and
2) the variance of the outputs of all estimators within the random forest.

More formally, consider a random forest regressor that consists out of N trees. For a new test input, all trees in the forest yield an individual predictive mean and variance, respectively μ_i and σ_i^2 for $i \in \{1, \ldots, N\}$, which are calculated over all n training data points within the leaf that is associated with the test input. To illustrate, consider the tree that is described in Figure 2.7 the j^{th} tree in the random forest. If the test input is associated with leaf 1, then:

$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{2}(3 + 11) = 7$$

$$\sigma_j^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_j) = \frac{1}{2}((3-7)^2 + (11 - 7^2)) = 16$$

Next, using the law of total variance, Hutter et al. define the predicted mean and variance of the random forest as follows (for a derivation see (Hutter et al., 2014, p. 17)).

$$\mu = \frac{1}{n}\sum_{i=1}^{N} \mu_i \tag{2.10a}$$

$$\sigma^2 = \left(\frac{1}{N}\sum_{i=1}^{N} \sigma_i^2 + \mu_i^2\right) - \mu^2 \tag{2.10b}$$

The variance described in Equation 2.10b becomes larger if the variation between the verdicts of the different trees, μ_i, increases. Furthermore, σ^2 increases if σ_i^2 increases. That is, the total variance increases if the predictions that are made by the individual trees are uncertain.

2.2.2 Acquisition functions

Given the predictive posterior distribution that is provided by the surrogate model of the Bayesian optimization process, the acquisition function is used to decide where to sample (see Algorithm 1, line 8). The acquisition function regulates the trade-off between the exploration and exploitation of the surrogate model by combining its posterior mean and variance. In this thesis, a variant of the upper confidence bound (UCB) will be used, an acquisition function that balances between exploration and exploitation using a parameter β (Srinivas et al., 2010).

$$a(x) = \mu(x) + \beta\sigma(x) \tag{2.11}$$

For large values of β, the acquisition function is large for regions where the standard deviation of the posterior distribution, $\sigma(x)$, is large. For noiseless objective functions, a large $\sigma(x)$ implies that the confidence of the surrogate function is low due to a small number of samples in that particular region (for example, consider the relatively large 95% confidence interval of the posterior in Figures 2.5 and 2.6 for $x \in [8, 10]$). Consequently, making a new sample in this region could be considered an exploratory act (Frazier, 2018).

For small values of β, the acquisition function is more likely to be large for regions where the posterior mean of the surrogate model is large. That is, the acquisition function focuses on the regions where, according to the surrogate, one is very likely to observe a large value (Frazier, 2018). In other words, larger values for β encourage the Bayesian optimization process to exploit the beliefs sketched by the surrogate function.

For noisy objective functions however, the effect of β on the exploration-exploitation trade-off is less sharply defined. This is because the posterior variance can be a composite of the noise around the objective function[2] and the modelling uncertainty that comes with a small number of samples, as was the case for the noiseless objective function. Therefore, higher values of β are not necessarily associated with exploratory behaviour, as a high posterior variance could also reflect a very noisy objective function. Additionally, the evaluation of samples from a region that is governed by a high level of objective noise is not guaranteed to lower the posterior variance. This is because the magnitude of the objective noise $r(x)$ is, in contrast to $\check{\sigma}(x)$, independent of the location of the evaluations. As such, the algorithm could get stuck when "exploring" a region that is associated with a high posterior variance that is rooted in a high objective noise, as the magnitude of the posterior variance only slightly decreases with the evaluation of new samples.

Consequentially, it would be worthwhile to obtain a reliable estimate of the noise around the objective function, so that the acquisition function can be better informed. A better estimate of the objective noise at a certain location could be acquired by making replications, as explained in the next section.

2.2.3 Replication strategies

The repeated evaluation of a sample at the same location, also known as replication, provides an insight into the observed variance of a function (Binois & Gramacy, 2021). Replications are performed in lieu of the evaluation of a new, unseen sample. Therefore, the pseudo code describing a basic Bayesian optimization algorithm can be extended as shown in Algorithm

[2]This is the noise that has been estimated with σ^2_{noise} in Figure 2.5 and $\boldsymbol{r(x)}$ in Figure 2.6.

2. Compared to the basic Bayesian optimization process that is described in Algorithm 1, a replication scheme has been added in line 9, and line 10 and 11 have been updated accordingly.

Algorithm 2 A Bayesian optimization algorithm with replications

Require: $N > 0$, $n_{random} + n_{informed} = N$
1: Initialize the statistical model, h, as a prior on f
2: Evaluate $f(x)$ for $n_{random} \geq 0$ initial locations $x_i \in \boldsymbol{x}_0 : \{x_0, \ ..., \ x_n\}$, yielding the set $\boldsymbol{y}_0 : \{y(x_0), \ ..., \ y(x_n)\}$
3: $n \leftarrow 0$
4: $y \leftarrow \boldsymbol{y}_0$
5: $x \leftarrow \boldsymbol{x}_0$
6: **while** $n \leq n_{informed}$ **do**
7: Update h given all existing sample locations $\boldsymbol{x}$ and associated evaluations $\boldsymbol{y}$
8: Find x^*_{acqf} that maximizes the acquisition function given the updated h
9: Use a replication scheme to decide whether to evaluate x^*_{acqf} or make a replication $x_{repl} \in \boldsymbol{x}$ such that $x_{chosen} = x^*_{acqf} \vee x_{repl}$
10: Update the set of existing locations based on the chosen sample: $\boldsymbol{x} \leftarrow \boldsymbol{x} \cup \{x_{chosen}\}$
11: Update the set of existing evaluations based on the chosen sample: $\boldsymbol{y} \leftarrow \boldsymbol{y} \cup \{y(x_{chosen})\}$
12: Increment n
13: **end while**
14: Select $\hat{x}^*$ that is presumed to maximize f

Yet, given the added pseudo code, it remains unclear how to decide whether to evaluate a new sample or make a replication; and what existing sample should be replicated if a replication is made. Different strategies have been proposed to approach this problem. In general, one could decide to re-evaluate an existing sample $x_{repl} \in \boldsymbol{x}$ instead of querying the new sample x^*_{acqf}, if x_{repl} and x^*_{acqf} are in close proximity. However, what should be considered "close" is highly dependent on the replication budget and the dimensionality of the optimization problem.

Instead, the incentive for replicating samples could be based on the influence of the replication on the global predictive accuracy of the surrogate model. For Gaussian process regression, the integrated mean squared prediction error (IMSPE), which is the integrated "de-noised" posterior variance, can be considered a measure of the global predictive accuracy (Ankenman et al., 2008; Binois et al., 2019). More specifically, recall from the previous section that the posterior variance of a noisy objective function, $\sigma^2(x)$, is a combination from the modelling uncertainty, $\check{\sigma}^2(x)$, and $r(x)$, the noise that is associated with the objective function itself. Therefore, the IMSPE can be defined as in Equation 2.12 by subtracting the observational noise from the posterior variance (Binois et al., 2019). The integral is taken over the data $\boldsymbol{x}$.

$$\text{IMSPE}(\boldsymbol{x}) = \int_{x \in \boldsymbol{x}} \check{\sigma}^2(x) \ dx = \int_{x \in \boldsymbol{x}} \sigma^2(x) - r(x) \ dx \tag{2.12}$$

In the previous section, it was concluded that the modelling uncertainty $\check{\sigma}^2(x)$ merely arises from the spacing of the modelled input values $\boldsymbol{x}$ (see Figure 2.4). Because the IMSPE is a measure of $\check{\sigma}^2(x)$, it is independent of the outcomes that are associated with the modelled data. Therefore, the IMSPE can not only be used to make an assessment of the predictive accuracy of the surrogate model in its current state ($\boldsymbol{x} = \{x_0, ..., x_n\}$), it is also possible to calculate the IMSPE for a next sample ($\boldsymbol{x} = \{x_0, ..., x_{n+1}\}$) *without* the need of evaluating x_{n+1} (Binois et al., 2019).

The latter observation allows one to "look ahead" by providing a heuristic for what sample, either the new sample or an existing one, should be chosen to evaluate. Interestingly, Binois et al. point out that this lookahead strategy can be elaborated by recursively adding the best

replicate or the new sample (Binois et al., 2019). That is, given a lookahead horizon h, Binois et al. recursively create h unique sequences that contain h replications and one evaluation of a new sample. Here, the authors exploited the fact that the surrogate model does not need to be updated to propose a new "best replication" after the IMSPE has been evaluated. Since the surrogate model needs to be updated to provide a best new sample (x^*_{acqf}), each sequence is limited to containing a single new sample. To illustrate the process described above, the 3 sequences that correspond to a lookahead horizon of $h = 2$ are shown in Table 2.1.

Sequence index	*Sample type*		
1	New sample	Replication	Replication
2	Replication	New sample	Replication
3	Replication	Replication	New sample

Table 2.1: An example of the recursive lookahead scheme as proposed by Binois et al. (2019) with a horizon of $h = 2$.

For each sequence, the authors compare the IMSPE of the surrogate model that is calculated using the existing samples $\boldsymbol{x}$ and all samples from that particular sequence. Next, the first step of the sequence that minimizes the IMSPE is chosen as the final strategy (Binois et al., 2019).

Chapter 3

Related work

Purpose of the chapter

The idea of adaptive methodologies within experimentation is not new to neuroscience: several studies have been conducted to demonstrate the feasibility of the real-time exploration of experimental conditions. This chapter first presents an overview of the existing adaptive methodologies in neuroscience (Section 3.1). Subsequently, the focus is shifted towards the work that regards adaptive experimental designs in BCI (Section 3.2). Lastly, Section 3.3 presents the possible improvements on, and additions to, the existing methodologies that have been discussed.

3.1 Adaptive methodologies in neuroscience

The performance of a (neuroscientific) experiment can be time-consuming and costly (Lewi et al., 2009). Therefore, experiments are set up to be as efficient or informative as possible. In particular when a sequential experiment design is considered, where the quantity or composition of the samples depend on prior observations (Robbins, 1952), the optimization of such design can be supported with machine learning. The use cases of adaptive experiments can roughly be divided into three categories. Each of the categories will be discussed in the sections below.

3.1.1 Modelling neural activity

Firstly, the information that is gathered during an experiment can be used to probe and model the activity of neurons. To this end, J.W. Pillow and M. Park use Bayesian active learning to assess how the firing rate of a simulated neuron changes given a change in stimulus parameters such as orientation, spatial frequency and position (Pillow & Park, 2016). Furthermore, J.W. Pillow, M. Park and Lewi et al. simulate the estimation the receptive fields of a neuron to demonstrate applicability of Bayesian active learning to high-dimensional optimization problems (Lewi et al., 2009; Pillow & Park, 2016). Alternatively, Shababo et al. aim to maximize the mutual information between the model parameters and the data over trials to map the connectivity between neurons.

3.1.2 Discriminating between statistical models

In the work from the second category, design optimization has been used to discriminate between competing statistical models. As such, Caravagarno et al. use adaptive design optimization to select what model has the highest expected utility in a memory research setting (Cavagnaro et al., 2010). Similarly, Myung et al. have conducted simulation-based experiments to find the optimal design that helps identifying the most-likely data generating model from a set of competing models. To do so, Myung et al. try to iteratively optimize the expected utility of each model, to decide what model displays the best performance (Myung et al., 2009).

3.1.3 Evoking desired brain states

Lastly, experiments can be optimized in real-time to select stimuli that evoke a desired brain-state. The work that falls within this last category opposes the previously discussed work in the sense that the experiments have been conducted in the real world instead of in a simulated environment. Examples include the automatic optimization of stimuli to elicit a pre-defined target brain state in the context of functional magnetic resonance imaging (fMRI) (Lorenz et al., 2016). Similar work has been done in the field of BCI, where Sosulski et al. investigated the feasibility of the online adaptation of BCI stimuli (Sosulski et al., 2022). As the work by Sosulski et al. forms the basis for this thesis, it is discussed in more detail in the next section.

3.2 Online optimization of brain-computer interface stimulus characteristics

To act upon the brain's electrical activity, BCI protocols have been designed to effectively record and decode brain signals. The recorded brain signals and associated decoding performance are influenced by the parameters of the protocol. As the parameter values that lead to the optimal decoding performance are subject-specific (Sugi et al., 2018), research has been conducted towards an automatic optimization method to tailor the parameters to each subject.

In (Sosulski et al., 2022), the authors apply the idea described above to optimize the stimulus onset asynchrony (SOA) within an auditory oddball task. During this auditory paradigm, a subject is presented with a sequence of high (target) and low-pitched (non-target) tones. As discussed in Section 2.1, the ERP responses that are elicited with the presentation of these tones differ for the target and non-target variants. To make the classification of the tones more facile, Sosulski et al. aim to find the SOA that maximizes the class discriminability of the evoked target and non-target ERPs. To this end, the problem at hand is modelled as a black-box function: a continuous objective function is introduced, where the SOA is used as a regressor for an estimate of the ERP discriminability, such as the amplitude of the P300 response or the classification performance on a single-trial, which is reflected by the area under the receiver operating characteristic curve (ROC AUC) (Sosulski et al., 2022).

More formally, let $f(x) : \mathbb{R} \to \mathbb{R}$ be the objective function that describes the ERP discriminability for different SOAs. Given this setting, Sosulski et al. aim to find the value for $x \in \mathbb{R}$ that maximizes f, i.e.:

$$\max_x f(x) \tag{3.1}$$

Yet, the calculated ROC AUC scores or the measured P300 amplitude y are assumed to be noisy measurements of the objective function $f(x)$, that is

$$y(x) = f(x) + \varepsilon \tag{3.2}$$

where ε denotes the noise factor around the measurement (Williams & Rasmussen, 2006, p. 8).

Now, recall that there is no information regarding the gradient of the objective function. Therefore, information about the shape and global optimum of the objective function can only be obtained by making queries to f. However, consider that for each query that is made to f, a new trial should be recorded. This makes the objective function expensive to evaluate. The expensiveness of the objective function limits the total number of queries that can be made due to time or budgetary constraints (Frazier, 2018; Jones et al., 1998; Lam et al., 2016). As a consequence, Sosulski et al. resort to Bayesian optimization, a popular parameter tuning framework that can be used in the context of expensive black-box functions, as demonstrated by Jones et al. (1998), J. Močkus (1975), and Snoek et al. (2012) and explained in more detail in Section 2.2. Unfortunately, due to the high noise level and a limited number of participants, Sosulski et al. were not able to prove that their optimization methods significantly outperform the traditional random search method (Sosulski et al., 2022).

3.3 Directions for improving the existing methodologies

As already hinted at in the Introduction, the optimization algorithm that has been used by Sosulski et al. is presumed to perform suboptimally due to the violation of some of the assumptions that have been made. In particular, the presumed presence of heteroskedastic observational noise could have led to a suboptimal fit of the statistical model that was employed. Sosulski et al. modelled $y(x)$ using homoskedastic Gaussian process regression, which is able to model the observational noise by learning the noise parameter σ^2_{noise} (Equation 2.7). Yet, due to the utilization of a single noise parameter, the model does not have the capacity to capture the hetereskedasticity of the noise. This could, in combination with the relatively high amount of noise around the objective function, lead to a suboptimal fit (Sosulski et al., 2022). To this end, Sosulski et al. relate the expected performance of the optimization algorithm to the objective to noise ratio (ONR), a measure of the noise around the objective function (Sosulski et al., 2022). To alleviate this problem, one could use the more flexible heteroskedastic Gaussian process that is able to learn the input specific noise estimate, $r(x)$ (Equation 2.8).

Other aspects of adaptive methodologies for BCI that have not been explored include the optimization of multiple parameters at the same time, as also touched upon by Pillow and Park (2016). More formally, $f(\boldsymbol{x}) : \mathbb{R}^d \rightarrow \mathbb{R}$ now represents a d-dimensional optimization problem. A multi-dimensional optimization problem could for example arise from a visual BCI paradigm where, next to the SOA, the colour or intensity of the presented stimuli could be optimized. The efficiency of Bayesian optimization is proven to be more impactful when higher-dimensional optimization problems are considered (Rai et al., 2019). At the same time, a baseline that is based on random sampling could become less efficient, for example if the optima are concentrated within a small subspace. Interestingly, the class of multidimensional problems comes with the possibility of interacting parameters, where the value of one parameter influences the perception of another. For instance, if the colour of a stimulus is represented with the hue, saturation, value (HSV) colour model, then there is no use tweaking hue and saturation parameters if the value is close to 0. The interaction of stimuli could not be alleviated for certain by altering the design of the paradigm as even the effects of stimulus characteristics that do not seem directly related might still interact on a neural level. Therefore, it is interesting to learn how the performance of the optimization algorithm changes if the dimensionality of the objective function increases.

Chapter 4

Methods

Purpose of the chapter

This chapter describes the methods that have been used to simulate a closed-loop BCI experiment, perform Bayesian optimization and evaluate the results. First, the dataset that forms the backbone of the simulation is described in Section 4.1. Next, the details pertaining to the implementation of the simulation are laid out in Section 4.2. Subsequently, Section 4.3 formulates the Bayesian optimization algorithm in more detail. To evaluate hypotheses H_1: and H_3:, the just-defined optimization algorithms are tested on various simulated objective functions. These experiments are presented in more detail in Section 4.4. Lastly, Section 4.5 presents the metrics that have been used to evaluate the different experimental settings. To get an idea of a suitable parameterization of the Bayesian optimization algorithms, the effectiveness of different optimization setups are explored. This process, which also evaluates hypothesis H_4:, is described at length in Appendix A to improve the flow of this chapter.

4.1 Data

The data that have been used in this work have been recorded with an auditory ERP paradigm by S. Denzer. During this paradigm, the participants have been presented with six auditory word stimuli (Denzer, 2016). The paradigm comprises three conditions that differ in the spatial location of the source of the stimuli. In the first condition, 6D, the stimuli were presented from six speakers that are placed in a ring, surrounding the participant who sits in the centre. In the 1D condition, a single, non-spatial speaker is used to present the stimuli. Lastly, in the HP condition, the stimuli were presented using non-spatial headphones (Denzer, 2016).

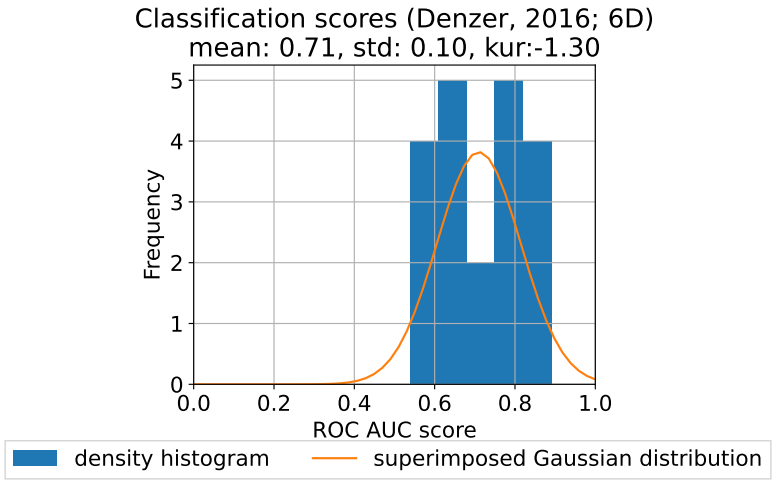


Figure 4.1: The distribution of the ROC AUC scores that are associated with the 6D condition of the word ERP dataset.

In total, 20 participants participated in the study. Each participant was presented with 540 target and 2700 non-target stimuli per condition. The data have been recorded using 63 EEG channels from 250ms before the stimulus onset until 1200ms after the stimulus onset. The elicited ERP responses are characterized with a negative and positive peak in the amplitude of the signal (as shown in Figure 2.1). Across the participants, these negative and positive responses can broadly be found within respectively [150, 400]ms and [350, 750]ms from the stimulus onset. The classification scores differ between the three conditions, where condition 6D was found to yield the best results (Denzer, 2016). The distribution of the ROC AUC scores that are associated with the condition 6D is shown in Figure 4.1. Throughout this thesis, the dataset that has been discussed in this section will be referred to as the "word ERP" dataset.

4.2 Simulations

4.2.1 Defining an objective function

The dataset that is the described previous section is used to simulate the online optimization of BCI parameters. Naturally, it is impossible to change the parameters that have been used to record the data. However, it is possible to tweak the parameters that are used to process and classify the data. To identify what parameters are suitable for this purpose, the classification process is described in more detail.

The classification of event-related potentials

As mentioned before, the used data have been collected during an ERP paradigm, where, for a number of epochs, a continuous EEG signal is recorded at 63 positions across the participant's scalp (Sosulski & Tangermann, 2022). As a result, the recorded data have both spatial and temporal features and are stored in a two-dimensional matrix of shape $N_c \times N_t$ per epoch. Here, N_c denotes the number of EEG channels (the spatial component) and N_t denotes the number of time points per epoch that the data have been recorded over (the temporal component).

The recorded epochs are band-pass filtered to the range [0.5-16]Hz, resampled to 40Hz and classified into target and non-target ERPs using linear discriminant analysis (LDA), as done by Sosulski and Tangermann (2022). Furthermore, a baseline correction was applied to the signals that have been recorded within the 200 ms that preceded the stimulus[1]. While LDA can be used to classify D-dimensional input vectors, it is impossible to use the algorithm for the classification of (spatio-temporal) matrices (Bishop & Nasrabadi, 2006, p. 187). Therefore, the temporal and spatial information are integrated in a single feature vector per epoch. To this end, the two-dimensional matrices in $\mathbb{R}^{N_c \times N_t}$ are flattened by concatenating the rows to create a vector in $\mathbb{R}^{N_c N_t}$, as shown in Equation 4.1 (Sosulski & Tangermann, 2022). The notation x_c^t is used to indicate that x has been recorded with channel c at time point t.

$$\begin{bmatrix} x_{c_1}^{t_1} & x_{c_2}^{t_1} & \dots & x_{c_{N_c}}^{t_1} \\ x_{c_1}^{t_2} & x_{c_2}^{t_2} & \dots & x_{c_{N_c}}^{t_2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{c_1}^{t_{N_t}} & x_{c_2}^{t_{N_t}} & \dots & x_{c_{N_c}}^{t_{N_t}} \end{bmatrix} \in \mathbb{R}^{N_c \times N_t} \xrightarrow[\text{flattening}]{\text{row-wise}} \begin{bmatrix} x_{c_1}^{t_1} & x_{c_2}^{t_1} & \dots & x_{c_{N_c}}^{t_1} & x_{c_1}^{t_2} & \dots & x_{c_{N_c}}^{t_{N_t}} \end{bmatrix} \in \mathbb{R}^{N_c N_t} \tag{4.1}$$

Under the hood of LDA, the feature vectors for all N_e epochs within the training data can be stacked to create the matrix $\boldsymbol{X} \in \mathbb{R}^{N_c N_t \times N_e}$ for each class. Next, the algorithm bases its verdict

[1]The application of a baseline correction introduces non-stationarities to the data, which violates assumptions that are implemented by more elaborate versions of linear discriminant analysis such as Toeplitz-LDA (Sosulski & Tangermann, 2022). However, baseline correction is still applied as LDA based on merely shrinkage regularization is not hampered by this operation.

on, among others, the estimated covariance matrix of the mean-free data, $\tilde{\boldsymbol{X}}$, $\hat{\boldsymbol{\Sigma}} = \frac{1}{N_e - 1}\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^T$ (Sosulski & Tangermann, 2022), (Bishop & Nasrabadi, 2006, p. 188). It has been found that the estimated covariance matrix can be a sub-optimal estimator of the true covariance matrix (see (Blankertz et al., 2011) for an in-depth discussion). Therefore, a balance is struck between $\hat{\boldsymbol{\Sigma}}$ and a unit sphere νI to alleviate the bias within $\hat{\boldsymbol{\Sigma}}$. This type of regularization, also known as shrinkage, is shown in Equation 4.2 (Blankertz et al., 2011).

$$\tilde{\boldsymbol{\Sigma}} = (1 - \gamma)\hat{\boldsymbol{\Sigma}} + \gamma\nu I \tag{4.2}$$

In Equation 4.2, ν is the average eigenvalue trace of $\hat{\boldsymbol{\Sigma}}$ and $\gamma \in [0, 1]$ regulates the degree of shrinkage that is applied (Blankertz et al., 2011). $\tilde{\boldsymbol{\Sigma}}$ is the regularized covariance matrix that is used as the basis for LDA.

Typically, not all N_t time points are used for the classification of the epochs. Instead, the dimensionality of the feature vector is reduced by averaging the recorded signals over N_i temporal intervals (Blankertz et al., 2011). The offset (t_0) and size (Δt) of these intervals are specific to the data that is being analysed (Sosulski & Tangermann, 2022). The number of intervals that is averaged over has been set to five, as done by Sosulski and Tangermann (2022), to limit the dimensionality of the optimization objective.

To obtain a ROC AUC score, the 3240 epochs that are recorded per participant are split in 2430 training epochs and 810 test epochs (corresponding to a 0.75/0.25 split). The training data are used to fit the LDA classifier, while the test data are used to evaluate the performance of said classifier. For each evaluation of the objective function, the same train-test split is made to guarantee identical outcomes for identical preprocessing classification strategies. The ROC AUC score that is associated with this split of 2430 training epochs and 810 test epochs will be considered the true value of the objective function, $f(x)$.

Optimizing the classification pipeline

The parameters that have been introduced in the previous section can be translated into the objectives that could be optimized. That is, the optimization algorithm can be tasked with finding the parameter values that lead to the best classification performance (ROC AUC): the outcome of the objective function. The parameters that have been optimized are shown in Table 4.1. The domains of the parameters that regulate the temporal averaging are chosen in such a way that the total time that can be averaged over is equal to $100 + 5 \cdot 140 = 800$ms. This was done because this time period is assumed to contain the information that is useful for the decoding process (Denzer, 2016).

Parameter	*Meaning*	*Domain*
γ	degree of shrinkage regularization	[0, 1]
t_0	offset of the temporal intervals	$[0, 100]$ms
$\Delta t_1, \Delta t_2, \Delta t_3, \Delta t_4, \Delta t_5$	size of the temporal intervals	$[30, 140]$ms

Table 4.1: The parameters from the classification pipeline that have been selected for optimization.

The dimensionality of the objective function depends on the number of parameters that is optimized. To test hypothesis H_3: and assess how the performance of the Bayesian optimization algorithm scales to multi-dimensional optimization problems, (combinations of) parameters have been tuned. The parameter combinations that have been assessed are presented in Table 4.2. In order to model a one-dimensional optimization problem, merely the degree of shrinkage regularization within the application of LDA is tuned. For a two-dimensional optimization problem, the degree of shrinkage and the offset of the temporal intervals are estimated. Lastly, the sizes

of the five intervals can also be estimated separately. In combination with the optimization of the shrinkage parameter and the offset of the intervals, the dimensionality of the objective can total to seven.

Optimized parameters	*Dimensionality of optimization problem*
γ	1
γ, t_0	2
γ, t_0, $\Delta t_1, \Delta t_2, \Delta t_3, \Delta t_4, \Delta t_5$	7

Table 4.2: The parameter combinations that have been optimized to model optimization objectives with different dimensionalities.

With the seven-dimensional optimization problem, we aim to discover how the optimization algorithm performs with higher-dimensional optimization problems. Furthermore, we are interested in how the optimization algorithm handles parameters that interact, as discussed in Section 3.3. In the current problem definition, the size Δt_i of each time window t_i shifts the start of subsequent windows t_j where $i < j$ further away from the stimulus onset. Thus, the information that is contained in each window depends on the sizes of all prior windows, which could make the optimization problem more complex.

The parameters that are not tuned during the optimization process are taken from the literature. In particular, γ is set to the Ledoit-Wolf estimate (Ledoit & Wolf, 2004), t_0 to 100ms and the five interval sizes $\Delta t_1, \ldots, \Delta t_5$ to 70, 60, 70, 110 and 90ms as done by Sosulski and Tangermann (2022). Furthermore, the parameterization that is fully based on the initialization presented by Sosulski and Tangermann (2022) (i.e. no optimization is applied) will be used as a reference. This particular parameterization will be referred to as the *paper parameters* in the remainder of the thesis. Note that this parameterization is not guaranteed to give the best score with the ordinary LDA algorithm that is used in this thesis, as in the work of Sosulski et al. Toeplitz-LDA is used (Sosulski & Tangermann, 2022).

(Enhancing) the objective landscape

As the optimal values for the parameters that constitute the objective function can be subject-specific, the shape of the objective landscape could differ heavily between subjects. To provide an idea of the shape of the objective landscape, the relation between the value of the shrinkage parameter γ, the offset of the temporal intervals t_0; a fixed interval size Δt_i for all averaging windows i; and the ROC AUC score are plotted. The three-dimensional plots have been generated by fixing the values of the three parameters in turn to $\gamma = 0.05$, $t_0 = 0.1s$ and $\Delta t = 0.08s$. In Figure 4.2, the objective landscapes from the participants VPpblz_15_08_14 (Figure 4.2a-c) and VPpboa_15_08_11 (Figure 4.2d-f), condition 6D from the auditory aphasia dataset are shown. The colour of the plotted surfaces can be interpreted as an indicator of the ROC AUC score, where blue represents an ROC AUC close to 0.5 whilst red represents an ROC AUC close to 0.8, as described with the colourbar in Figure 4.2j. From the Figures 4.2(a-c), we can learn that the objective landscapes of participant VPpblz_15_08_14 shows a clear optimum when the value of the shrinkage parameter goes to 0 while the size of the interval goes to $0.14s$. In contrast, there is not such a clear trend when the Figures 4.2(d-f) are regarded. For participant VPpboa_15_08_11, the ROC AUC score of the classification oscillates around 0.5, regardless of the parameter values. The latter observation could be rooted in the lack of informativeness of the recorded data. In these cases, the classification ROC AUC is close to the probability of random guessing, which is 0.5 for a binary problem.

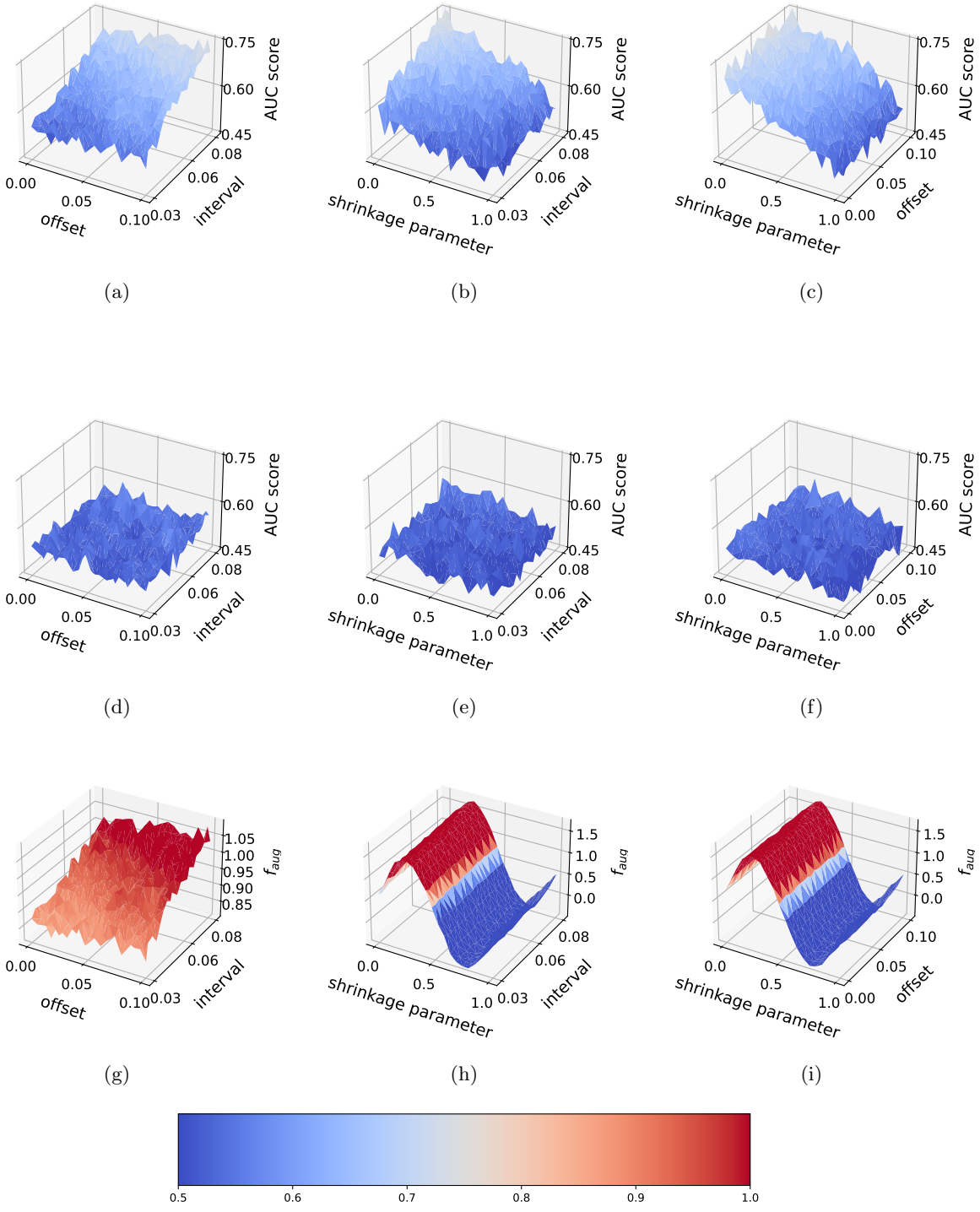


Figure 4.2: The parameter space of participants VPpblz_15_08_14 (a-c, g-i) and VPpboa_15_08_11 (d-f) from the word ERP dataset. The condition that is associated with the shown data is 6D. The figures have been generated by sampling from the three-dimensional objective function, while setting the three parameters in turn to $\gamma = 0.05$ (a, d), $t_0 = 0.1s$ (b, e) and $\Delta t_i = 0.08s$ (c, f). The offset and interval parameters are presented in seconds. Figures g-i describe the parameter space of participant VPpblz_15_08_14 that has been enhanced according to Equation 4.3.

To avoid flat landscapes, or to make sure that the optimization process is able to find the optima that are not located on the edge of the domain, the objective landscape could be enhanced with

artificial optima. The latter can be done by combining the classification outcome with a sine wave. To do so, the input domain of the parameter γ, which runs from 0 to 1 is mapped to the domain [0, 2π] by multiplying γ by 2π. Next, the augmented objective function is created by adding the sine wave to the original outcome, as shown in Equation 4.3 for the one-dimensional objective function.

$$f_{aug}(\gamma) = \sin(\gamma * 2\pi) + f(\gamma) \tag{4.3}$$

Due to the addition of the sine wave, the value of the augmented outcome is not limited to [0, 1], the domain of the ROC AUC. Even though this makes the simulations less realistic, I decided against clipping $f_{aug}(\gamma)$. The latter operation would yield closely spaced input values with the same maximal value, which is not realistic either.

4.2.2 Introducing noise to the objective function

Now that the objective function has been specified, we move on to the introduction of noise to model the noisy circumstances that can be found within a BCI experiment. Recall that the true value of the objective function, $f(x)$ is calculated using the 2430 training epochs and 810 test epochs. The noisy objective function $y(x)$ can be modelled in two ways, which will be discussed below.

4.2.3 Superimposed noise

Firstly, noise can be superimposed on the objective function $f(x)$ by sampling the observed outcome $y(x)$ from a distribution that is centered at the true evaluated ROC ROC AUC score, $f(x)$. For this purpose, a Gaussian distribution with mean $f(x)$ and variance $r(x)$ could be used, such that

$$y(x) \sim \mathcal{N}(f(x), r(x)) \tag{4.4}$$

As the variance $r(x)$ is a function of x, this method can be used to model heteroskedastic noise. As a consequence, this method will be used to assess hypothesis H_2:. An advantage of this method is that the amount of noise that has been added to the objective function is known. Therefore, this method is used to model the noise when the optimization algorithm's capability of reconstructing the noise around the objective function is assessed.

4.2.4 Sampling noise

Secondly, the noise can be modelled by calculating the ROC AUC over a subset of the training data, rather than using all 3240 epochs to train and evaluate the LDA classifier. This process directly introduces a sampling noise to the objective outcome, which removes the necessity to sample $y(x)$ from a probability distribution centered at $f(x)$. This second method is hypothesized to yield a noise structure that, in some aspects, is more similar to the noise that can be found within the field of BCI than the method described by Equation 4.4. The is because the recorded epochs could be considered merely a sample from a larger dataset too, such that they come with their own sampling noise. Yet, the generated sampling noise is independent of the location of the input samples x. Therefore, this method can be considered homoskedastic and does not account for generating the heteroskedastic noise that is presumed to pollute the objective function. Moreover, it is harder to control the quantity of the noise that is introduced.

To introduce the sampling noise, random subsets of 450 epochs are sampled without replacement or class stratification, which are distributed into 338 training epochs and 112 evaluation epochs. Due to the randomness in the sampling, the ROC AUC scores that correspond to this training and evaluation process will be different for identical inputs. Therefore, these scores are considered the observed outcomes $y(x)$, whereas the ROC AUC score that has been obtained with all 3240 epochs is considered the true objective $f(x)$.

4.3 Implementation of the Bayesian optimization algorithm

There are several ways to adapt the "standard" Bayesian optimization algorithm presented in Algorithm 1. In this section, different components are introduced that are expected to improve the performance of the algorithm. The implementations of the different architectures and components are discussed in the sections below. The algorithm is formalized in Python 3.8. Furthermore, it is designed in the modular fashion that is described in Figure 4.3. The modular abstraction makes it easy to swap different components while keeping the general structure of the algorithm intact. In the remainder of this section, the implementation of each individual component will be described in detail.

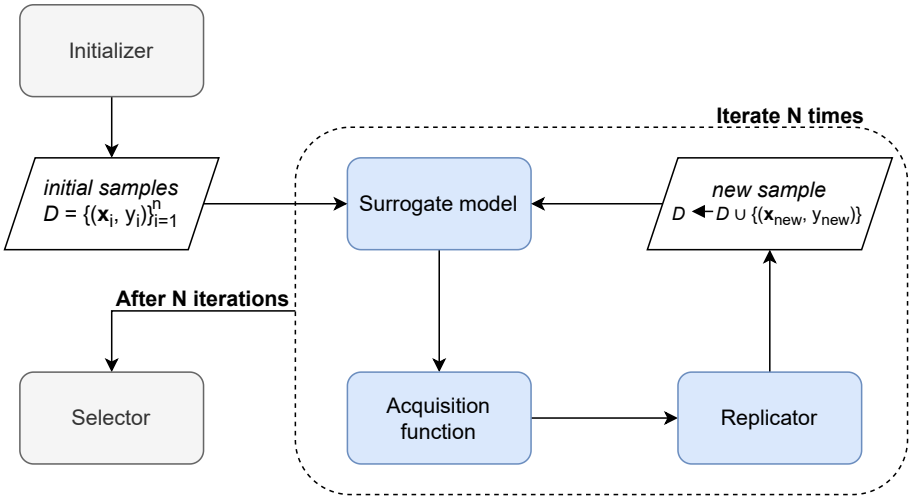


Figure 4.3: The modular architecture of the Bayesian optimization algorithm. The different modules are indicated with the rounded boxes. The gray-coloured modules are only used once, whereas the blue modules are updated during every iteration of the algorithm. The parallelograms indicate changes to the state of the dataset containing the evaluated samples, $\mathcal{D}$.

4.3.1 Initializer

As its name suggests, the initializer is used to select the N_{random} samples that are used to initialize the surrogate model of the Bayesian optimization algorithm. The choice of input locations is based on a (quasi-)random sequence. More specifically, the initializer uses Sobol sequences (SciPy, version 1.10.1, (Virtanen et al., 2020)) to ensure that the input space is covered more evenly than ordinary, uniform sampling ("Constrained Bayesian Optimization with Noisy Experiments, author=Benjamin Letham and Brian Karrer and Guilherme Ottoni and Eytan Bakshy", 2018; Sobol', 1967).

4.3.2 Surrogate models

In this work, three surrogate models are assessed. The first model, homoskedastic Gaussian process regression, is a popular choice within the literature (Frazier, 2018). The second, heteroskedastic Gaussian process is more complex than the homoskedastic variant. As such, the heteroskedastic model is expected to be better able to handle complex noise structures. The third model, random forest regression, is also able to capture heteroskedastic noise. Furthermore, the random forest is expected to require less training time than the heteroskedastic Gaussian process as the training of the forest can be done in parallel; a great benefit in time-constrained settings.

Homoskedastic Gaussian process regression

To model a homoskedastic Gaussian process, the implementation from the BoTorch library (version 0.8.1, (Balandat et al., 2020)) is used. This library allows for an automatic optimization of the Gaussian process parameters using type II MLE (Gardner et al., 2018). The Gaussian process is parameterized with a Matérn $5/2$ kernel. The prior distribution on the length scale ℓ is set to a Gamma distribution with $\alpha = 4.0$ and $\beta = 1.0$ to prevent the prior from being too strong.

Heteroskedastic Gaussian process regression

In an attempt to model the presumably heteroskedastic noise that pollutes the BCI data, the Most likely heteroskedastic Gaussian process has been implemented, as proposed by Kersting et al. (Kersting et al., 2007). The three individual Gaussian processes that form this model are implemented using BoTorch. The hyperparameters that have been used to initialize the Gaussian processes are shown in Table 4.3.

Gaussian process	*Kernel*	*Length scale prior*
GP1	Matérn $5/2$	Gamma(4.0, 1.0)
GP2	Matérn $5/2$	Gamma(1.5, 4.0)
GP3	Matérn $5/2$	Gamma(4.0, 1.0)

Table 4.3: The Gaussian process hyperparameters. Gamma(α, β) is used to indicate a Gamma distribution with shape α and rate β.

The Matérn kernel is chosen after the example by Sosulski et al. (2022), where its use is advised on the basis of its flexibility by Stein (1999) as well. The length scale prior of the GP2, the Gaussian process that is used to learn the noise $r(x)$ has a higher density for the smaller length scales than the length scale priors of the Gaussian process that are trained on the input data itself. While this prior belief is not very strong, the data is expected to be more smoothly distributed than the noise. Furthermore, if the length scale of GP1 would be too small, GP1 would be able to cross every data point, which would not leave any residual distance for the estimation of $\boldsymbol{z}$ for the training of G2. While Kersting et al. propose to train Most likely process for a number of iterations until convergence (Kersting et al., 2007), all processes have only been trained once as the increase of training iterations did not seem to have a large influence on the correctness of the fit. The input data are normalized to the unit cube prior to fitting the Gaussian process.

Random forest regression

The random forest is a simpler alternative to the combination of Gaussian processes described above. To implement a random forest that provides a posterior distribution, the random forest from the scikit-learn library (version 1.2.1, (Pedregosa et al., 2011)) has been adapted according to the proposal of Hutter et al. (Hutter et al., 2014). The number of trees in the forest has been set to 10, while the minimum samples per split was set to 5, as done by Hutter et al. (2014). In order to make the random forest regressor compatible with the rest of the Bayesian optimization pipeline, which is tailored to the Gaussian posteriors that come with Gaussian processes, the posterior mean and variance of the forest are used as basis for a multivariate normal distribution. More specifically, using the vectors describing the posterior mean $\boldsymbol{\mu(x)}$ and variance $\boldsymbol{\sigma^2(x)}$, the distribution of the forest is defined as $\mathcal{N}(\boldsymbol{\mu(x)}, \boldsymbol{\sigma^2(x)}I)$. Here, I denotes the identity matrix.

4.3.3 Acquisition function

Given the distribution over the input data that is provided by the surrogate model, the acquisition function is optimized to decide where to make the next sample. The acquisition function that is used in this project is a version of the upper confidence bound (Equation 2.11) that is adapted to limit the exploration parameter β to the interval $[0, 1]$. This is done to be able to optimize β more easily later on due to the bounded input domain. The definition of the bounded upper confidence bound is provided in Equation 4.5. Just like the Gaussian process regression model, the acquisition function is optimized using BoTorch.

$$a(x) = (1 - \beta)\mu(x) + \beta\sigma(x) \tag{4.5}$$

The limitation of β to the domain between 0 and 1 requires the posterior means and variances to be z-scored over the input space. This operation prevents either component from dominating the acquisition function. That is, without normalization, either component from Equation 4.5 could dominate the acquisition function. For example when an objective function that is modelled with a very high posterior mean, e.g. in the hundreds, is modelled with a variance that is not in the same ballpark, say $\sigma(x) = 1$, then the algorithm is very unlikely to explore since β is bounded between 0 and 1.

In the previous chapter, it was discussed that a high β does not necessarily result in exploratory behaviour due to the composite nature of the posterior variance ($\sigma^2(x) = r(x) + \check{\sigma}^2(x)$) of the surrogate model (Binois et al., 2019). When the objective function is very noisy, which is not uncommon within a BCI setting, the acquisition function could prefer regions that are associated with a high $r(x)$ instead of a high $\check{\sigma}^2(x)$. In theory, the exploratory behaviour could be recovered by merely considering the modelling uncertainty $\check{\sigma}^2(x)$ in the acquisition function as shown in Equation 4.6.

$$a(x) = (1 - \beta)\mu(x) + \beta\check{\sigma}(x) = (1 - \beta)\mu(x) + \beta\sqrt{\sigma^2(x) - r(x)} \tag{4.6}$$

In practice however, this approach could lead to undesired side effects. Even if the noise around the objective function, $r(x)$, can be estimated from the data, which can be done using Gaussian process regression, this estimate can be erroneous. A flawed estimate of the objective noise could have problematic consequences for the acquisition function as proposed in Equation 4.6. The latter holds in particular when magnitude of the noise around the objective function is overestimated. Namely, if the estimated modelling uncertainty is larger than the posterior variance, then the remaining "modelling uncertainty" will be negative (or even undefined if the standard deviation is considered). As a consequence, there is a lower incentive to explore the input domain where $r(x) > \sigma^2(x)$, which prevents the surrogate model to improve the estimate of $r(x)$ within said domain. Therefore, the definition that is presented in Equation 4.5 is used for all optimization algorithms.

4.3.4 Replicator

The repeated evaluation of a sample at the same location could lead to a better estimate of the noise at that location. To this end, two different replication strategies have been implemented. Both implemented strategies will be specified in more detail below.

Sequential replicator

The sequential replicator implements the IMSPE-based replication strategy discussed in Section 2.2.3. The horizon of the replicator has been set to two to limit the computation time.

Variance replicator

Because the IMSPE-based strategy from the sequential replicator is only compatible with a surrogate model that can make an estimate of the noise around the objective function, this strategy cannot be used in combination with random forest regression. Therefore, an additional, simpler replication scheme has been implemented that is based on the location and the posterior variance of the evaluated samples. That is to say, whether or not a replication is made, depends on two criteria:

1) the Euclidean distance between the the existing sample with the highest associated evaluated value, $x^*_{existing}$, and the proposed new sample x^*_{acqf} needs to be smaller than all the distances between the proposed sample and the other existing locations $x_i \in \mathcal{D} \backslash \{x^*_{existing}\}$; and
2) the posterior variance that is associated with $x^*_{existing}$ has to be larger than the posterior variance at the x^*_{acqf}.

If both requirements are met, then a replication is made, where the input location of the existing sample with the highest associated evaluated value is evaluated again. This strategy aims to "verify" the highest evaluated value by replicating the associated input location. However, the recommendation by the acquisition function is also taken into account as the replication is only made if $x^*_{existing}$ is the closest sample in $\mathcal{D}$ to x^*_{acqf}.

4.3.5 Selector

Once the Bayesian optimization algorithm has finished N iterations of evaluating a sample, updating the surrogate model and informing the acquisition function and replicator to make a new evaluation, the selector is used to decide what sample is the most likely to maximize the objective function (Figure 4.3). The selector makes a decision given the posterior mean $m(x)$ and variance $\sigma^2(x)$ of the surrogate model. During the final selection, a high posterior variance makes a location unfavourable, as this indicates that the estimated $m(x)$ comes with a high uncertainty. Therefore, the selector evaluates all data points $x_i \in \mathcal{D}$ according to the following definition:

$$\text{fitness}(x) = (1 - \beta)m(x) - \beta\sigma(x) \tag{4.7}$$

Just like in the acquisition function, the results y and the posterior variances $\sigma(x)$ are z-scored. Next, the x with the highest fitness is chosen as the final output of the Bayesian optimization algorithm. In the definition from Equation 4.7, the fitness of an input stands or falls with the posterior variance $\sigma(x)$. Therefore, it is vital that the posterior variance is an accurate estimation of the variance of the objective function. In an attempt to monitor the quality of the surrogate model (and thus the posterior variance), different measures have been implemented.

Convergence measures

The convergence measures are based on the change of the posterior mean of the surrogate model when an increasing number of samples is considered. For each iteration of the Bayesian optimization algorithm, the surrogate model is fit on a dataset that has been extended with the sample (and associated evaluation) that has been selected during the previous iteration. Due to the inclusion of this new data point, the shape of the fitted model could differ a lot from the shape of the model during the previous iteration. If the shape of the fitted model is stable over the iterations of the Bayesian optimization process, then the model is assumed to be more reliable than a model whereof the shape is subject to a lot of change. To quantify the change in the shape of the surrogate model, two measures are proposed.

Magnitude of mean squared error The first measure is applicable to any surrogate model that provides a posterior mean. The measure attempts to model the change in the shape of

the posterior mean by calculating the mean squared error (MSE) between the posterior mean of the current and the previous iteration. For an optimization process of N iterations, this results in a collection of $N-1$ MSEs. Then, the MSE that describes the difference in the posterior mean of the last and penultimate iteration is normalized to a value between 0 and 1 relative to all other MSEs, as demonstrated in Equation 4.8. In the equation, $\mathcal{C}$ indicates the collection of convergence measures, which in this case refers to the collection of MSEs. c_N is used to denote the convergence measure that has been recorded within the last iteration, which is the MSE that describes the difference in the posterior mean of the last and penultimate iteration.

$$\text{convergence score} = \frac{c_N - \min(\mathcal{C})}{\max(\mathcal{C}) - \min(\mathcal{C})} \tag{4.8}$$

Change in length scale The second measure works in a similar fashion but it is merely applicable to Gaussian processes as it monitors the change in the length scale of the kernel over the N iterations[2]. That is, for each iteration i, the length scale of the Gaussian process is stored. Furthermore, the deviation of the current length scale from the mean of the length scales that have been observed thus far is calculated. These deviations are normalized to a value between 0 and 1 as well, in the same fashion as the previously described MSEs.

Fitness The resulting convergence scores are incorporated into the fitness measure that is shown in Equation 4.7 to evaluate their influence on the Bayesian optimization process. For both measures, a value close to 0 suggests that the shape of the surrogate model has been stable over the iterations, whereas a value close to 1 indicates that the shape of surrogate model has been subject to a lot of change. If the surrogate model is deemed to be an unstable estimator, then the estimated posterior variance of the model should not be held in a regard that is similar to that of a surrogate model that seems to have converged. Therefore, the convergence score is inverted prior to being multiplied to the second part of Equation 4.7. The updated definition of the fitness is shown below, where $\tau = (1 - \text{convergence score})$, the inverted convergence score.

$$\text{fitness}(x) = \tau(1-\beta)y(x) - (1-\tau)((1-\beta)m(x) - \beta\sigma(x)) \tag{4.9}$$

In this definition, τ is the convergence score that regulates the trade off between the observed outcomes $y(x)$ and the modelled outcomes $m(x)$ and $\sigma(x)$, which respectively indicate the posterior mean and standard deviation of the surrogate model. If the surrogate model is not deemed trustworthy, then the convergence measure τ is close to 1. In that case, the fitness is largely determined by the observed outcomes. However, if the model is deemed stable over iterations, then a value of τ close to 0 gives extra weight to the modelled estimates of the objective function.

4.4 Analyses

4.4.1 Optimization strategies

We have just seen how the Bayesian optimization process can be initialized with different modules and strategies. Naturally, the performance of the algorithm may be subject to the number of (initialization) samples that are evaluated, the chosen balance between exploration and exploitation and the different modules that are used. The choice of these parameters is not straightforward and could very well depend on the problem domain at hand. Therefore, the different options within the Bayesian optimization process are assessed with a subset of the data to select the building blocks that constitute the algorithms that are evaluated on the remainder of the data. The subset that has been used comprises the subjects VPpblz_15_08_14,

[2]Even though the length scale of a Gaussian process kernel merely describes the smoothness of the resulting curve and not necessarily the shape, it is hypothesized that a large change in the length scale over iterations could indicate that the Bayesian optimization process is not fully converged.

VPpboa_15_08_11 and VPpbob_15_08_13, the first three subjects in the dataset. Based on this process, which is described in Appendix A, the following components and parameter settings are chosen: the value for β, which parameterizes both the acquisition function and the selector is set to 0.187. With this choice of β, an exploitative strategy seems to be preferred. Next, the optimization algorithms follow the *variance* replication strategy that is based on the location of, and the posterior variance around, the evaluated samples. Lastly, the proposed convergence measures did not seem to have a positive effect on the optimization process. Therefore, no such measure is used. The absence of a convergence measure does not imply that there is no selector used. Rather, the value of τ in Equation 4.9 is set to 0.

The surrogate model has not been preselected as the performance of all three surrogate models, the homoskedastic Gaussian process (BO_{homGP}), the Most likely heteroskedastic Gaussian process (BO_{hetGP}) and the random forest (BO_{RF}), is of interest. The three Bayesian optimization algorithms are compared against the traditional random search (RS) method. The latter method was also chosen as a baseline by Sosulski et al. (2022). More specifically, this method is initialized with a Sobol sequence. The locations of all subsequent samples are drawn uniformly from the input domain. The final verdict of the traditional random search method is the location x that is associated with the highest recorded outcome $y(x)$. With this choice of baseline, I hope to be able to prove that there can be a significant difference in performance between the random search method and the more informed Bayesian optimization strategies. An overview of the four optimization methods that are evaluated is presented in Table 4.4.

Name	Icon	Surrogate model	Initialization	Replication strategy	Convergence measure	β
BO_{homGP}		Homoskedastic GP	Sobol	Variance	-	0.187
BO_{hetGP}		Heteroskedastic GP	Sobol	Variance	-	0.187
BO_{RF}		Random forest	Sobol	Variance	-	0.187
RS		-	Sobol	-	-	-

Table 4.4: An overview of the three optimization strategies.

4.4.2 Optimization performance

The performance of the Bayesian optimization algorithms described above is evaluated on the remaining 17 subjects within the dataset to avoid a circular analysis. During the optimization process, the algorithm is tasked to find the optimal classification parameters with 8 random and 50 informed samples, totalling to 58 iterations. The main analysis of this thesis regards the optimization performance of the optimization algorithms in question. Furthermore, the capability of the optimization algorithms to model the objective function and noise is analyzed as well. All analyses have been run on the High Performance Computing (HPC) cluster that is hosted by the Donders Centre for Cognitive Neuroimaging (DCCN).

Optimizing the classification score

For each participant in the Word ERP dataset, a objective function is simulated for different dimensionalities as described in Section 4.2.1. The simulation of the objective function does not include any of the enhancements that are described in Section 4.2.1. The true objective is represented by the ROC AUC score that is calculated over the 3240 epochs of each participant (see Section 4.2.1 for more details). Additionally, two noise forms are evaluated. Firstly, the objective function is made noisy by calculating the ROC AUC score that corresponds to an observation over a subset of 450 epochs out of the total 3240 epochs, introducing a form of homoskedastic noise, as discussed in Section 4.2.2. Secondly, the noise is calculated as a function of γ to model heteroskedastic noise. In particular, this superimposed noise is modelled as $r(\gamma) = |sin(2\pi\gamma)|$, where $|\cdot|$ indicate the absolute value.

Modelling the objective function and associated noise

While the analyses provide an insight in the eventual performance of the optimization algorithms, one cannot know what happens under the hood of each algorithm when merely the optimization performances are presented. As, the research question specifically considers *potentially* heteroskedastic noise, the algorithms are tested on optimization problems that are troubled with both homo- and heteroskedastic noise. Furthermore, the optimization algorithms of interest are exposed to increasing noise levels to learn at what point the objective function is so noisy that performance of the algorithms start to break down. Additionally, the posterior distributions that are modelled by the surrogate models of BO_{homGP}, BO_{hetGP} and BO_{RF} are inspected as well. To this end, the models are applied to simulations with normal and enhanced objective functions where superimposed noise (Section 4.2.3) is applied to make sure that the distribution of the noise is known.

4.5 Evaluation

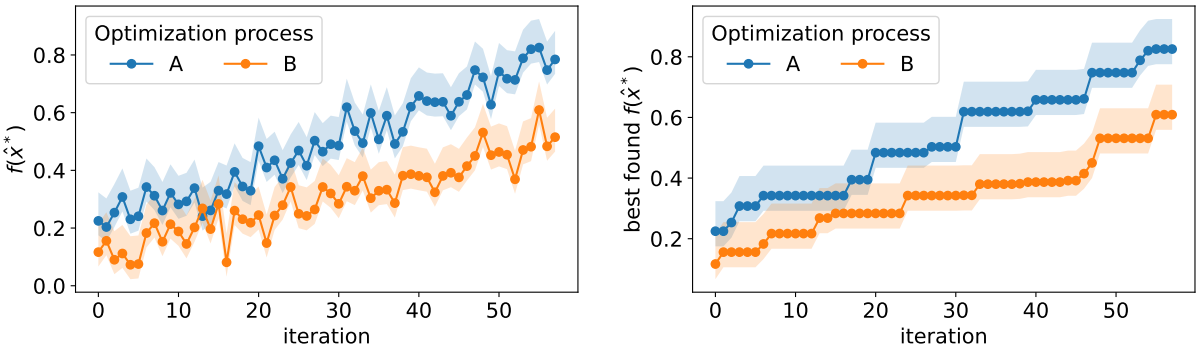
4.5.1 Metrics

To compare the performance of differently parameterized optimization processes, the evaluation metrics that have been proposed by Dewancker et al. are used (Dewancker et al., 2016). To illustrate these metrics, recall the pseudocode that has been presented in Algorithm 2. As highlighted in line 14 of the algorithm, the Bayesian optimization process is engineered to select a single location $\hat{x}^*$ that is presumed to maximize f. While the different algorithms could be ranked by comparing the noise-free outcomes that are associated with $\hat{x}^*$ after iteration N, one could argue that these comparisons are merely based on a snapshot of the algorithms' performances. That is to say, the quality of the found solutions might not generalize to a different number of iterations. Furthermore, the aforementioned comparison does not take into account the speed with which the solution is attained (Dewancker et al., 2016).

The metrics that are introduced by Dewancker et al. (2016) aim to counter both drawbacks by evaluating *traces* of optima that are proposed by the algorithm. To obtain a trace of optima, Dewancker et al. propose to query the selector of the Bayesian optimization algorithm at each iteration $i \in \{1, \ldots, N\}$, instead of making a single query when all N iterations have been completed. Next, all noise-free outcomes that are associated with the selected locations are calculated. To illustrate this principle, consider Figure 4.4a, where the optima traces of two hypothetical optimization processes are plotted. As the results are stochastic, Dewancker et al. suggest to refrain from basing the evaluation of the optimization processes on a single trace. Rather, each optimization method is ran repeatedly with different initializing samples to acquire a better assessment of the method's performance. As such, the traces that are shown in Figure 4.4a can be interpreted as the mean over all runs, where the shaded area represents a measure of the variance between the runs. In this work, the number of runs for each algorithm is set to 10. Given the different runs, Dewancker et al. calculate the "best found" metric: for each iteration i, this metric stores the location that has been associated with the highest score thus far[3], as shown in Figure 4.4b. According to Dewancker et al. the "best found" metric provides an assessment of the effectiveness of the optimization algorithms (Dewancker et al., 2016). Note that the metric is ever increasing. Eventually, the methods are evaluated by comparing the "best found" scores at iteration N.

In case the scores that have been calculated according to the "best found" metric at iteration N are inconclusive, the authors suggest to consider the speed with which the optima are attained as a tie-breaker. To assess the optimization speed of an algorithm, the area under the "best

[3]That is, the highest of all scores that are associated with the locations $\hat{x}_j^*$ for the iterations j for which holds $j \leq i$.



(a) (b)

Figure 4.4: The bases for the evaluation metrics proposed by Dewancker et al. (2016). In the left panel, an illustration of the optima traces of the hypothetical optimization processes A and B is shown. The right panel describes the "best found" trace that corresponds to the outcomes of panel a. Both algorithms have been ran for 58 iterations. All traces that are visualized are averaged over a number of optimization processes. The shaded area of each trace indicates the variance between the outcomes.

found"-curve that is presented in Figure 4.4b is calculated. This AUC score is not to be confused with the ROC AUC score that is used to model the objective function. The faster the algorithm has converged, the higher the AUC score. Therefore, methods that are associated with a high AUC score are favoured.

As the spacing of initializing samples could influence the performance of the optimization algorithms, the initializing samples of the different algorithms should be kept consistent over the runs. To illustrate, the i^{th} and j^{th} run of algorithm A are initialized with different samples if $i \neq j$ (because they are different runs). However, the i^{th} run of algorithm A is initialized with the same set of random samples as the i^{th} run of algorithm B to provide both algorithms with the same basis.

Chapter 5

Results

Purpose of the chapter

In this chapter, the performances of the algorithms BO_{homGP}, BO_{hetGP}, BO_{RF} and RS will be presented. The chapter is threefold: In Section 5.1, the results that are obtained from the comparison between the four optimization methods are shown. Next, statistical tests are used to verify whether the differences that are found between the models are significant. This process is described in Section 5.2. Lastly, Section 5.3 lays out the modelling qualities of BO_{homGP}, BO_{hetGP} and BO_{RF} by visualising the algorithms' capability of modelling the objective function and the associated noise. Furthermore, the effect of different noise levels on the performance of the algorithms is presented.

5.1 Optimization performance

This section presents the optimization traces from the four optimization algorithms BO_{hetGP}, BO_{homGP}, BO_{RF} and RS for the 20 participants from the Word ERP dataset. The participants VPpblz_15_08_14, VPpboa_15_08_11 and VPpbob_15_08_13 have been included out of curiosity: the scores that have been obtained with the data from these participants will not be considered in the statistical tests as these data have already been used to fine-tune the optimization algorithm in Appendix A. Recall from Section 4.4.2 that the algorithms have been tested with under two different noise types. The results that have been obtained from the optimization with homoskedastic sampling noise are considered first.

5.1.1 Optimization with homoskedastic noise

The optimization processes of the one, two and seven-dimensional optimization problems with homoskedastic noise are presented respectively in Figure 5.1,5.2 and 5.3. The traces that are shown in each pane of the figures represent the mean ROC AUC score per optimization algorithm. The mean is calculated over the 10 runs algorithm at each iteration index $i \in \{1, \ldots, 58\}$. The shaded area around the mean totals to 2 standard errors of the mean (SEM). Note that the 8 initializing samples are the same for each algorithm, as explained in Section 4.5.1. Therefore, the traces are identical for the iteration indices $0 < i \leq 8$.

From the results for the one-dimensional objective function, presented in Figure 5.1, BO_{hetGP} seems to outperform BO_{RF}, BO_{homGP} and RS for all participants but VPpboa_15_08_11, VPpbqb_15_08_06 and VPpboe_15_08_21. Additionally, BO_{hetGP} has a higher score than the paper parameters at iteration $N = 58$ for 9 out of the 20 participants. Furthermore, it is striking that aforementioned difference in performance becomes evident after a relatively small (about 25) number of iterations. The performance of BO_{RF} is for most participants comparable to the performance of the baseline, RS. Lastly, BO_{homGP} tends to perform worse than the baseline for all participants but VP_boa_15_08_11.

The results of the two-dimensional objective function, presented in Figure 5.2, are comparable

to the one-dimensional case. Yet, the differences between BO_{hetGP}, BO_{RF} and RS have become smaller. Despite the higher dimensionality of the optimization problem, the scores that are obtained by the four algorithms are comparable to the one-dimensional case. Furthermore, BO_{hetGP} outperforms the paper parameters at iteration $N = 58$ for 12 out of the 20 participants.

For the seven-dimensional objective function, for which the results are shown in Figure 5.3, the differences between the traces of the four algorithms are even less clear compared to the two preceding figures. Still, the baseline RS is outperformed by BO_{hetGP} and BO_{RF} for the majority of the participants. However, from presented traces, there seems to be no clear difference between the algorithms BO_{hetGP} and BO_{RF}. A drop in the general performance of the four algorithms can be observed as well, where BO_{hetGP} is only able to perform better than the paper scores for 4 out of the 20 participants. Still, the best found optimization score are typically found by BO_{hetGP}.

Using the optimization traces, the traces that describe the best found scores at each iteration, as presented in Figure 4.4b, can be calculated. To improve the flow of the chapter, these plots have been moved to Appendix B. As introduced in Section 4.5.1, the values of the "best found" traces at the last iteration are used to compare the different strategies. The "best found" scores that are averaged over the 10 runs are presented per participant in Appendix C. The average over all participants is described in Table 5.1, together with the SEM.

Optimization problem	BO_{homGP}		BO_{hetGP}		BO_{RF}		RS	
	μ	SEM	μ	SEM	μ	SEM	μ	SEM
One-dimensional	0.6743	0.0045	**0.7153**	0.0048	0.6919	0.0046	0.6884	0.0046
Two-dimensional	0.6695	0.0044	**0.7149**	0.0048	0.6999	0.0048	0.6877	0.0045
Seven-dimensional	0.6768	0.0042	0.7093	0.0047	**0.7095**	0.0046	0.6916	0.0046

Table 5.1: The average "best found" scores at the final iteration for the four tested optimization problems with homoskedastic noise.

From the table, it seems that the strategy BO_{hetGP} outperforms the strategies BO_{homGP}, BO_{RF} and RS for the one and two-dimensional optimization problems. Yet, the performance of the algorithm BO_{hetGP} tends to decline when the dimensionality of the objective function increases where BO_{RF} and RS display a increase in performance. As a consequence, the performance of the algorithms BO_{hetGP} and RS seems to be on par for the seven-dimensional optimization problem. The last-mentioned observation is also reflected in Figure 5.3. The algorithm BO_{homGP} has, regardless of the dimensonality of the objective function, the worst performance of the four optimization algorithms. The variance between participants is not affected by the increase in dimensionality.

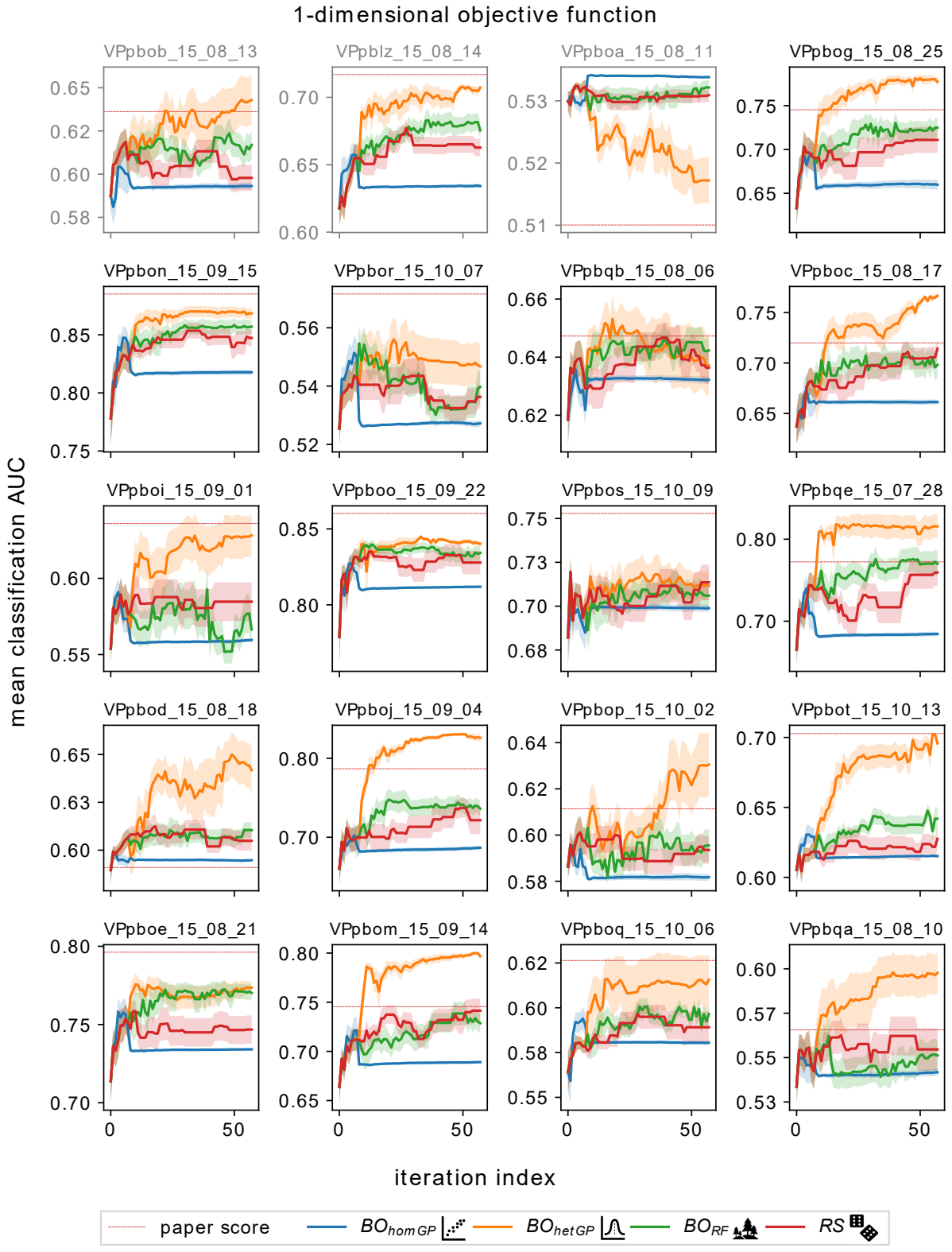


Figure 5.1: The optimization traces that have been found for the one-dimensional objective function for all participants of the word ERP dataset. The first three panes on the first row are outlined in gray for the data of these participants have been used to tune the Bayesian optimization algorithm in Appendix A. In each subplot, the traces of the optimization algorithms that are informed by homoskedastic Gaussian process regression (blue), heteroskedastic Gaussian process (orange) random forest regression (green) are plotted in combination with the random sampling baseline (red). The shaded area totals to two SEMs. The horizontal red bar indicates the paper score that is associated with each participant.

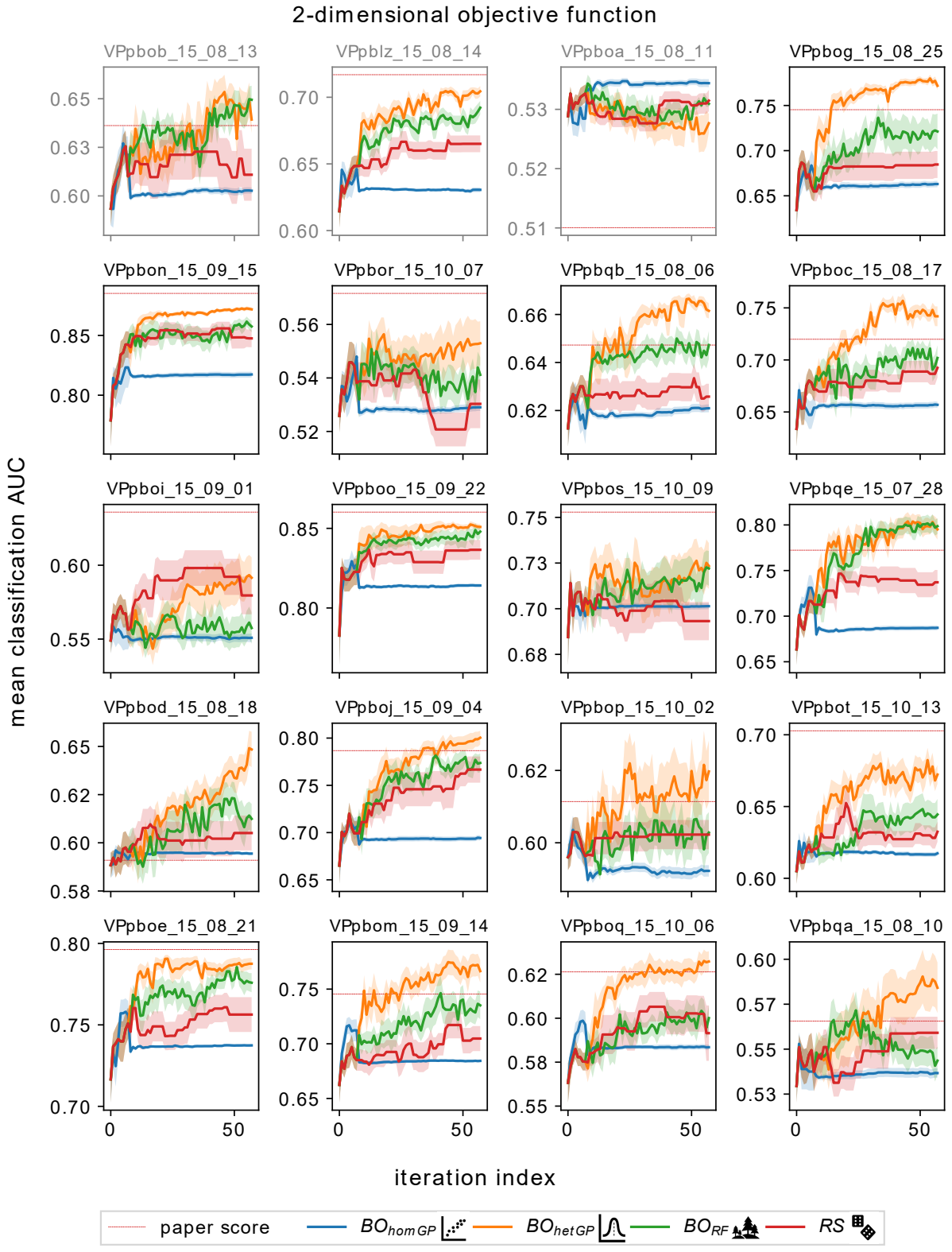


Figure 5.2: The optimization traces that have been found for the two-dimensional objective function for all participants of the word ERP dataset. The results are presented as in Figure 5.1.

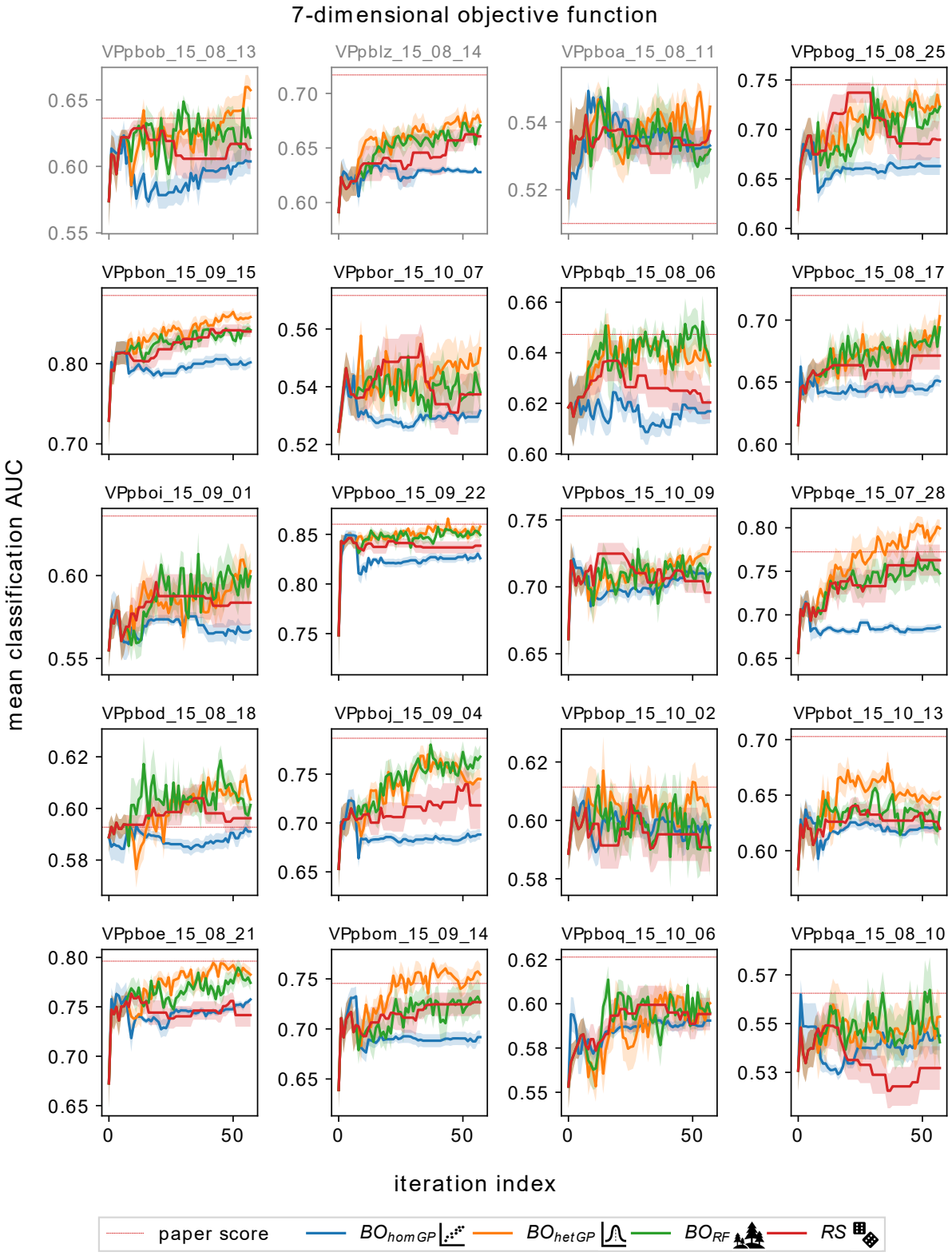


Figure 5.3: The optimization traces that have been found for the seven-dimensional objective function with heteroskedastic noise for all participants of the word ERP dataset. The results are presented as in Figure 5.1.

5.1.2 Optimization with heteroskedastic noise

The results that have been found for the optimization of the objective function that is modelled with heteroskedastic noise are broadly similar to the results found in the previous section. The results will be briefly discussed below. Just like in the previous chapter, the "bset found" traces that correspond to the obtained scores can be found in Appendix B.

For the one-dimensional objective function, shown in Figure 5.4, the difference between the performances of BO_{hetGP} and the other three models is larger than for the homoskedastic setting presented in Figure 5.1. This is because the performance of BO_{homGP}, BO_{RF} and RS seems to have worsened compared to Figure 5.1. The performance of BO_{hetGP} does not seem to have improved a lot: just like for the optimization problem with homoskedastic noise, the model performs better than the paper parameters for 9 out of the 20 participants. For most participants, the increase in the found classification ROC AUC starts to level after approximately 30 iterations.

For the two-dimensional objective function, which is shown in Figure 5.5, the performance of all models has dropped. While BO_{hetGP} still outperforms the three other models, the performances of BO_{homGP}, BO_{RF} and RS have become very comparable.

Lastly, when considering the seven-dimensional objective function, which is shown in Figure 5.6, it becomes clear that the performance of BO_{hetGP} has worsened to the point where it does not outperform the other models anymore. Rather, the models perform equally well, where the performance for all models for all participants but VPpboa_15_08_11 has dropped below the scores obtained with the paper parameters.

The mean best found scores over the 10 traces are presented in Table 5.2. The individual scores per participant can be found in Appendix C. When considering Table 5.2, one can find trends that are similar to the ones that are shown in Table 5.1: the performance of BO_{hetGP} worsens while the dimensionality of the objective function increases. While the performance of BO_{RF} increases with the dimensionality of the objective function, the performance of this method is still generally lower than BO_{hetGP}. In contrast to Table 5.1, a positive trend can now also be found for the models BO_{homGP} and RS.

Optimization problem	BO_{homGP}		BO_{hetGP}		BO_{RF}		RS	
	μ	SEM	μ	SEM	μ	SEM	μ	SEM
One-dimensional	0.6720	0.0043	**0.7137**	0.0046	0.6810	0.0043	0.6767	0.0044
Two-dimensional	0.6763	0.0043	**0.7080**	0.0047	0.6888	0.0046	0.6799	0.0044
Seven-dimensional	0.6800	0.0040	**0.7011**	0.0043	0.6907	0.0043	0.6854	0.0043

Table 5.2: The average "best found" scores at the final iteration for the four tested optimization problems with heteroskedastic noise.

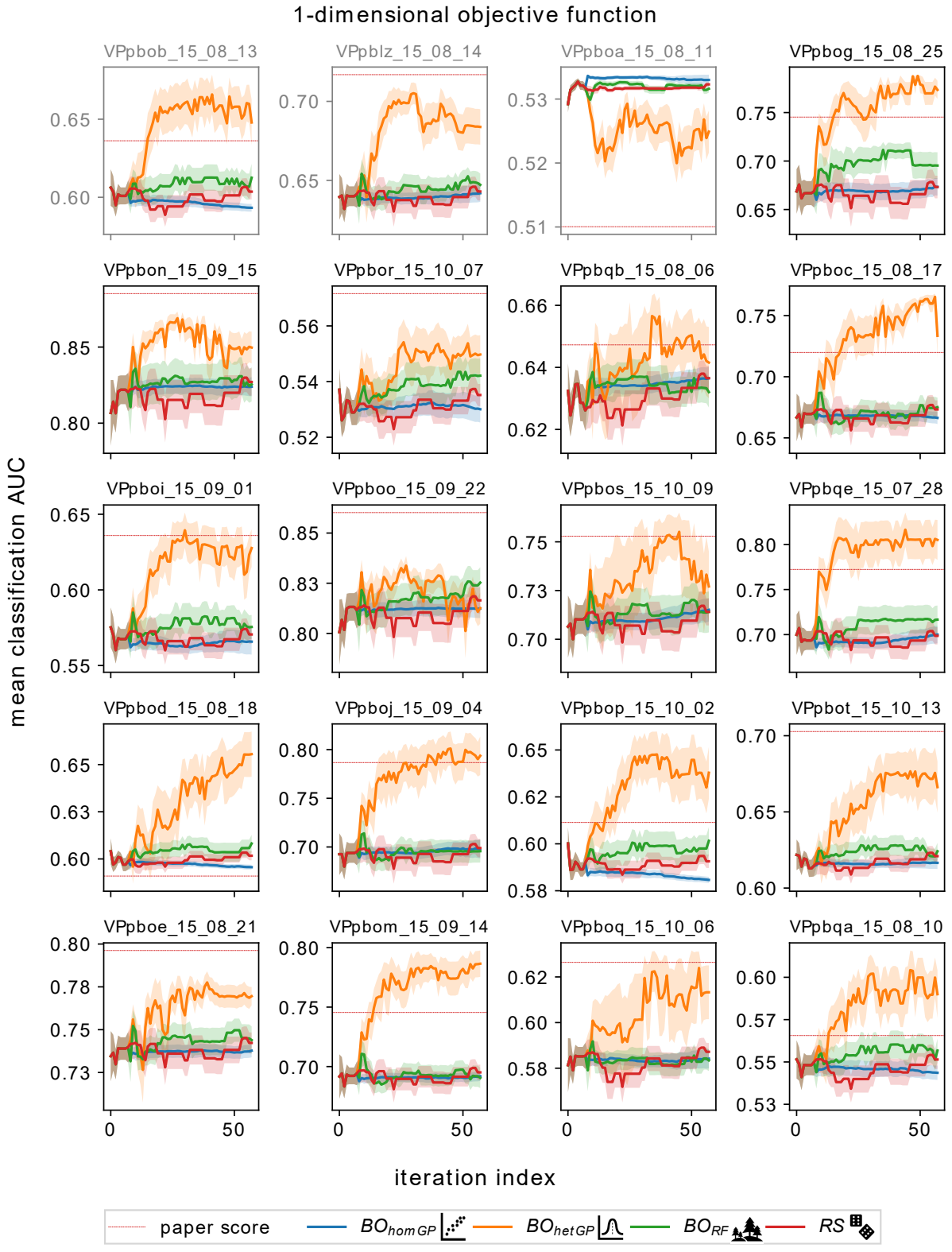


Figure 5.4: The optimization traces that have been found for the one-dimensional objective function with heteroskedastic noise for all participants of the word ERP dataset. The results are presented as in Figure 5.1.

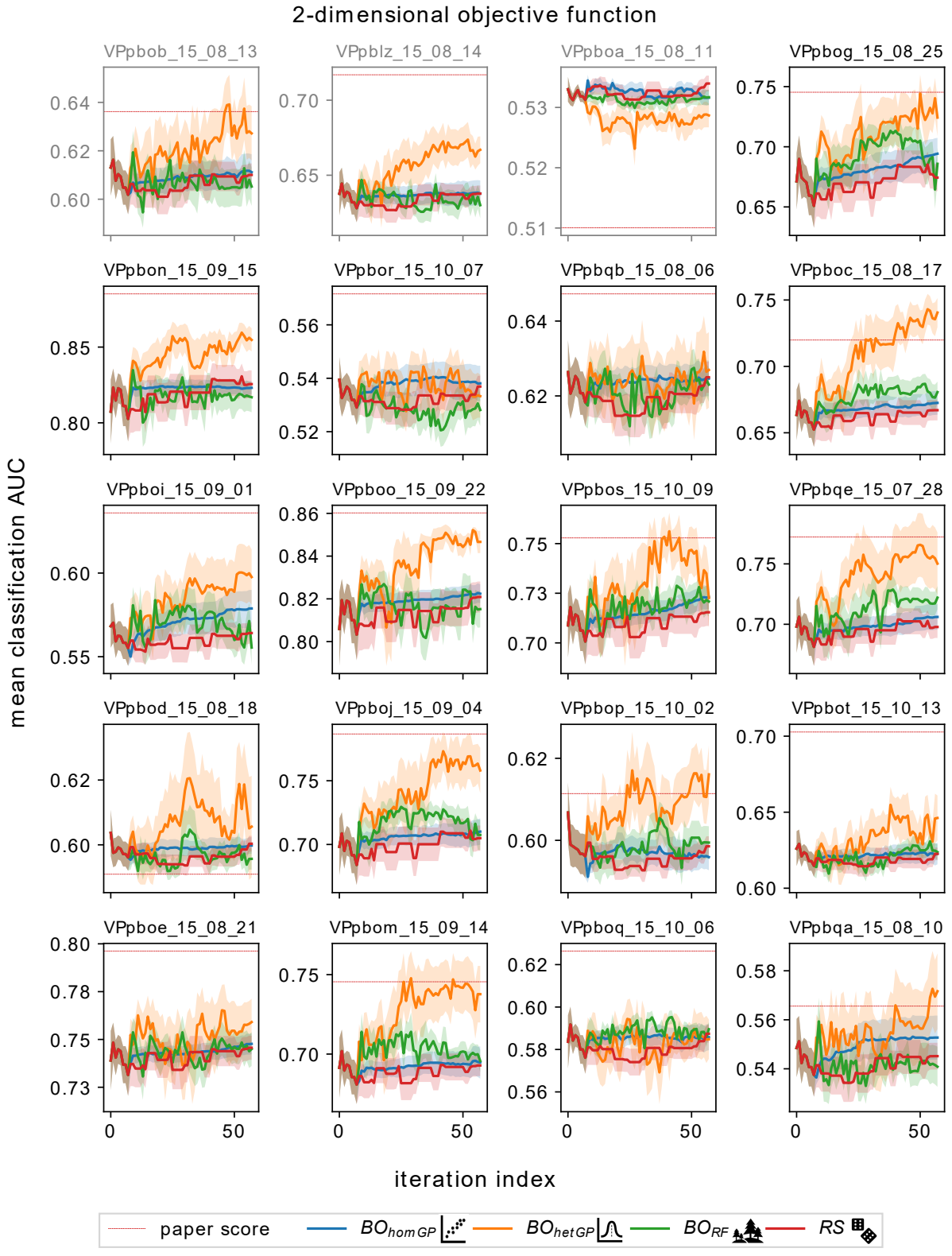


Figure 5.5: The optimization traces that have been found for the two-dimensional objective function with heteroskedastic noise for all participants of the word ERP dataset. The results are presented as in Figure 5.1.

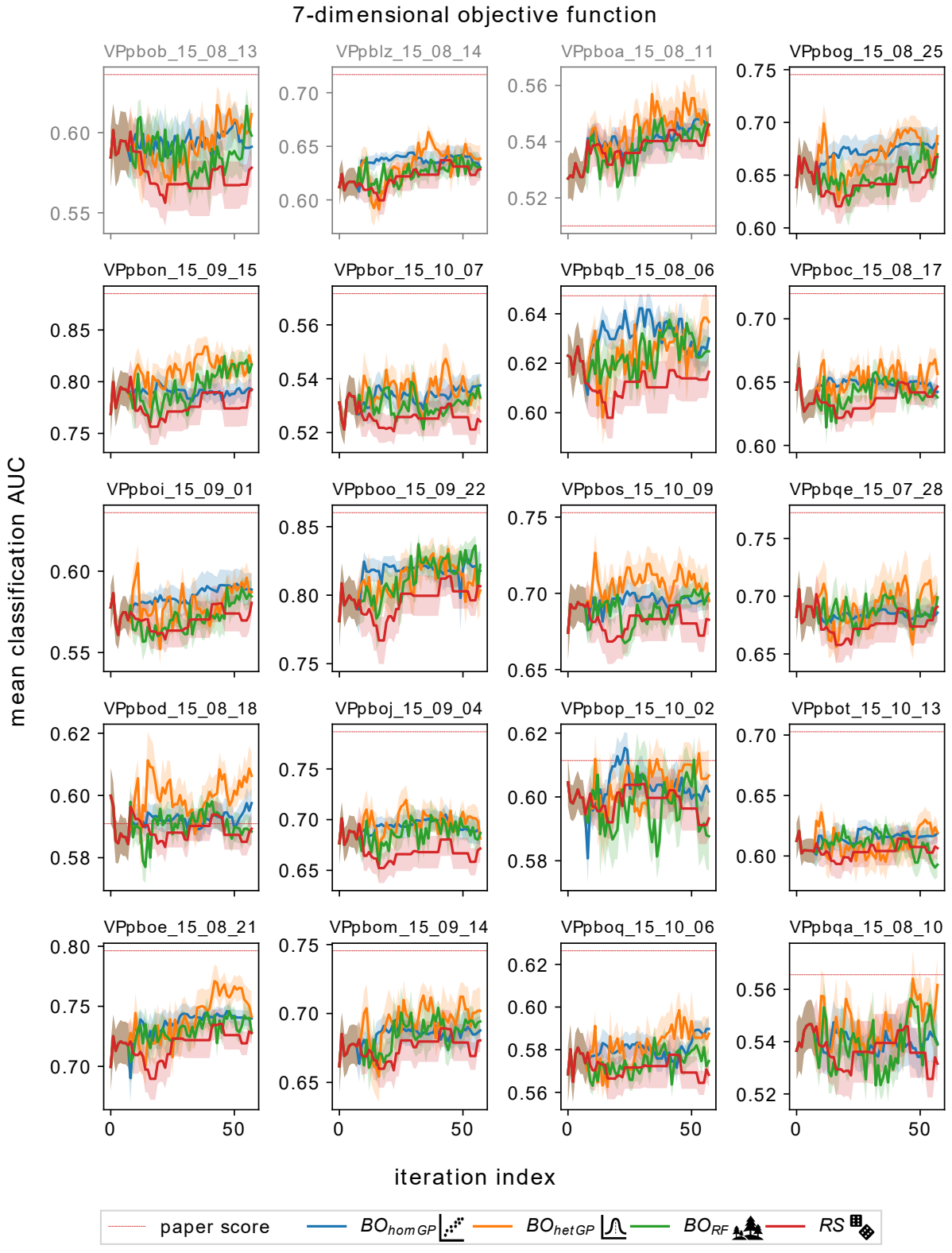


Figure 5.6: The optimization traces that have been found for the seven-dimensional objective function with heteroskedastic noise for all participants of the word ERP dataset. The results are presented as in Figure 5.1.

5.1.3 Optimization speed

The runtimes of the used algorithms are presented in the table below. RS is the fastest algorithm, followed in turn by BO_{homGP}, BO_{RF} and BO_{hetGP}. Furthermore, the runtimes of RS do not change when the dimensionality of the optimization problem changes (this is due to the lack of a surrogate model). From the strategies that have a surrogate model, the runtimes from BO_{homGP} and BO_{hetGP} seem to scale better than the ones from BO_{RF} when the dimensionality increases.

Optimization problem	BO_{homGP}	BO_{hetGP}	BO_{RF}	RS
One-dimensional	00:50:11	01:17:18	01:01:09	**00:45:52**
Two-dimensional	00:51:03	01:27:45	01:08:23	**00:46:41**
Seven-dimensional	00:55:08	01:38:22	01:36:18	**00:46:47**

Table 5.3: The wall clock runtimes of the four optimization strategies. A single execution of an algorithm includes 10 optimization runs of 58 iterations for each of the 20 participants. The presented durations are in the hh:mm:ss format.

5.2 Statistical tests

In the previous section, it was found that the algorithms BO_{hetGP} and BO_{RF} seem to outperform the baseline RS. To find out whether the differences between the observed results that were found in the previous section are statistically significant, statistical tests have been applied. Due to the setup of the analyses, where the stochastic nature of the optimization process is countered by running the algorithms 10 times per participant, there are 10 "best found" scores for each participant. To assess the statistical significance that is based on the differences between participants, the 10 "best found" scores for each participant are smoothed by taking the average. The average scores per participant are presented in the Tables C.1-C.3 in Appendix C. The participants VPpblz_15_08_14, VPpboa_15_08_11 and VPpbob_15_08_13 are not considered in the statistical tests, as the data from these participants have been used to tune the architecture of the optimization algorithms.

When one has to decide what statistical test to use, there are two characteristics of the data of interest: the relation between the samples and the distribution of the samples. Since we compare the differences between the performances within each participant, the samples of interest are paired. To discover whether the distribution of the samples allows for the use of a parametric test, the mean "best found" scores are visualized in a histogram. The distributions that correspond to the one-dimensional optimization problem are presented in Figure 5.7. The distributions that are found for the remaining optimization problems can be found in Appendix C.3.

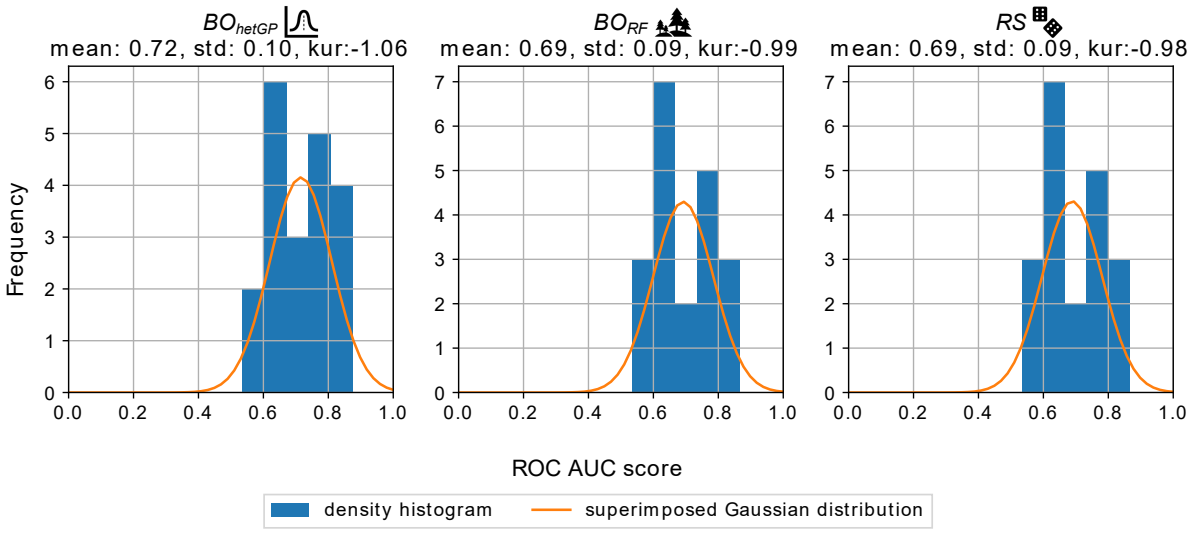


Figure 5.7: The histograms of the mean "best found" ROC AUC scores that have been found for the participants of the Word ERP dataset, given the one-dimensional objective function with homoskedastic noise. The bins are superimposed with a Gaussian distribution that is based on the mean and standard deviation of the samples. On top of each pane, the kurtosis, abbreviated with *kur*, of the histograms is shown. Fishers definition of the kurtosis is used, implying that a Gaussian distribution has a kurtosis of 0.0 (Virtanen et al., 2020).

Given the relatively low frequency of samples around the mean of the distributions, the shape of the histograms does not resemble a normal distribution. Rather, the scores seem to be distributed in a bimodal fashion. Recall that this distribution was also found by Denzer (2016), as shown in Figure 4.1. The bimodality of the distribution is also reflected in the kurtosis scores, which are far from 0.0. Therefore, the Wilcoxon signed-rank test, a non-parametric, statistical test for paired samples, is used (SciPy, version 1.10.1, (Virtanen et al., 2020)). The two-sided variant of the test is used as the sign of the difference between the groups can swing both ways. The p-values that have been calculated are presented in Table 5.4, where the performances of BO_{hetGP} and BO_{RF} are tested against the baseline, RS. For all the found p-values, the test statistic is in favour of either BO_{hetGP} or BO_{RF}.

The p-values per optimization problem

Optimization problem	BO_{hetGP} vs RS	BO_{RF} vs RS
One-dimensional (hom)	3.052×10^{-5}	7.968×10^{-2}
One-dimensional (het)	1.907×10^{-6}	3.814×10^{-6}
Two-dimensional (hom)	1.526×10^{-5}	8.392×10^{-4}
Two-dimensional (het)	3.815×10^{-6}	9.536×10^{-6}
Seven-dimensional (hom)	3.052×10^{-5}	3.052×10^{-5}
Seven-dimensional (het)	1.907×10^{-6}	1.986×10^{-3}

Table 5.4: The results of the two-sided Wilcoxon signed rank test. The scores of RS have been compared to the baseline. The mean scores per participant served as samples for each test. The abbreviations *hom* and *het* are used to indicate the use of respectively homoskedastic and heteroskedastic noise.

The verification of multiple hypotheses leads to the problem of multiplicity, where there is an increased risk of Type I errors (false positives) (Holm, 1979). To counter this phenomenon, the Holm-Bonferroni correction is applied. During this procedure, the found p-values are first sorted in an ascending order. Next, each p-value is compared to a significance level that depends on the index of the p-value in the collection of n sorted values. More specifically, for a p-value

at index $i \in \{1, 2, \dots n\}$, the associated significance level is

$$\text{significance level} = \frac{\alpha}{n + 1 - i} \tag{5.1}$$

In this definition, $0 < \alpha < 1$ is fixed for all tests (Holm, 1979). The Holm-Bonferroni correction considers the sorted p-values sequentially, starting by comparing the smallest value with the significance level α/n until a null hypothesis is encountered that is accepted. The latter is the case when the p-value is larger than or equal to the associated significance level, indicating that there is no significance difference between the compared groups (Holm, 1979). The outcomes of the Holm-Bonferroni method with $\alpha = 0.05$ are described in the table below. Given Table 5.5, the differences between the scores that are obtained by BO_{RF} and RS for the one-dimensional objective function with homoskedastic noise is the only non-significant result.

As BO_{hetGP} performs significantly better than both BO_{homGP} and RS for all noise functions and dimensionalities, the results found are in line with the hypotheses H_1: and H_2:. In contrast, hypothesis H_4: does not hold for BO_{hetGP}: the performance of the model tends to worsen when the dimensionality of the objective function increases, whereas the performance of RS improves (see Table 5.1 and 5.2). Even though the performance of BO_{RF} increases with the dimensionality of the objective function and BO_{RF} significantly outperforms RS in the higher dimensions, one cannot state that the performance diffence between the models is clearly growing.

Comparison	*p-value*	*Significance level*	*Status of null hypothesis*
BO_{hetGP} vs RS (1D, het)	1.907×10^{-6}	$0.05/12 = 4.167 \times 10^{-3}$	Rejected
BO_{hetGP} vs RS (7D, het)	1.907×10^{-6}	$0.05/11 = 4.545 \times 10^{-3}$	Rejected
BO_{RF} vs RS (1D, het)	3.814×10^{-6}	$0.05/10 = 5.000 \times 10^{-3}$	Rejected
BO_{hetGP} vs RS (2D, het)	3.815×10^{-6}	$0.05/9 = 5.556 \times 10^{-3}$	Rejected
BO_{RF} vs RS (2D, het)	9.536×10^{-6}	$0.05/8 = 6.250 \times 10^{-3}$	Rejected
BO_{hetGP} vs RS (2D, hom)	1.526×10^{-5}	$0.05/7 = 7.143 \times 10^{-3}$	Rejected
BO_{hetGP} vs RS (7D, hom)	3.052×10^{-5}	$0.05/6 = 8.333 \times 10^{-3}$	Rejected
BO_{RF} vs RS (7D, hom)	3.052×10^{-5}	$0.05/5 = 1.000 \times 10^{-2}$	Rejected
BO_{hetGP} vs RS (1D, hom)	3.052×10^{-5}	$0.05/4 = 1.250 \times 10^{-2}$	Rejected
BO_{RF} vs RS (2D, hom)	8.392×10^{-4}	$0.05/3 = 1.667 \times 10^{-2}$	Rejected
BO_{RF} vs RS (7D, het)	1.986×10^{-3}	$0.05/2 = 2.500 \times 10^{-2}$	Rejected
BO_{RF} vs RS (1D, hom)	7.968×10^{-2}	$0.05/1 = 5.000 \times 10^{-2}$	Accepted

Table 5.5: The results of the Holm-Bonferroni correction. The dimensionality of the objective function has been abbreviated with xD. The abbreviations *hom* and *het* are used to indicate the use of respectively homoskedastic and heteroskedastic noise.

5.3 Under the hood of the optimization process

5.3.1 The validity of the chosen parameters

Provided that the optimization processes BO_{hetGP} and BO_{RF} outperform the baseline RS for the higher dimensional optimization problems, it is important to verify the strategies that have been used by the algorithms. Just the fact that there is a high ROC AUC score associated with the chosen parameters does not necessarily imply that the found parametrization is sound. In the figures below, the distributions of the "optimal" parameters of the seven-dimensional optimization problem according to BO_{hetGP} and BO_{RF} are shown. The simulated data has been recorded from participant VPpboo_15_09_22, a participant with whom the algorithms achieved a relatively high score, to verify whether the scores indeed come from sensible parameter choices.

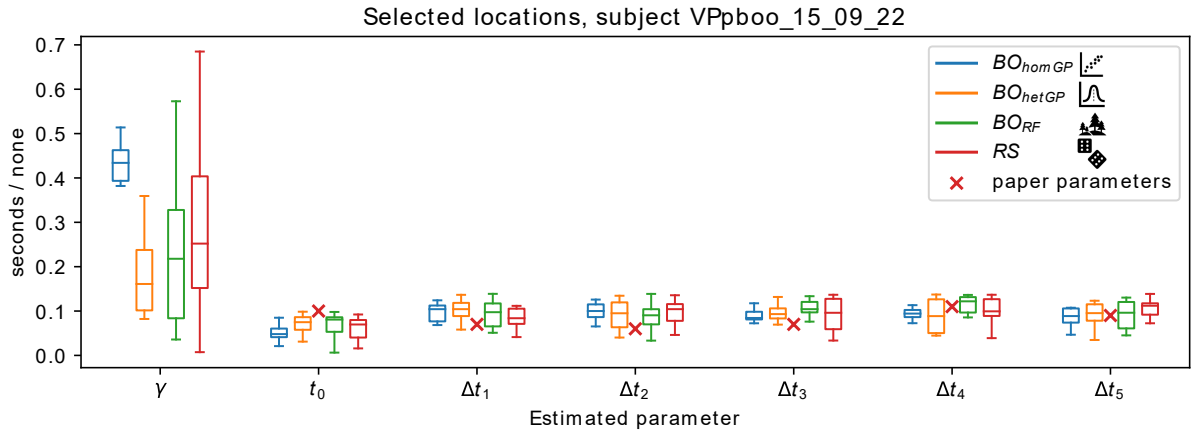


Figure 5.8: The distributions of chosen parameter values for the 10 different runs of BO_{homGP}, BO_{hetGP}, BO_{RF} and RS on the seven-dimensional optimization problem based on participant VPpboo_15_09_22. The paper parameters refer to the parameterization that is used by Sosulski and Tangermann (2022).

As shown in Figure 5.8, the distributions of the chosen parameters are in the same league as the parameters that have been used in Sosulski and Tangermann (2022). Furthermore, it is remarkable that the selected values for γ vary a lot more than the other parameters.

5.3.2 The posterior distribution of the surrogate model

Next to the validity of the submitted locations, the posterior distribution that is modelled by the surrogate models is of interest. As mentioned in Section 4.3.2, the Most likely heteroskedastic Gaussian process regressor is presumed to be able to make an adequate capture of the objective function and the associated objective noise $r(x)$. This hypothesis is taken to the test by visualising the posterior mean, variance and estimated $r(x)$ of the model. The results that have been obtained for the optimization of γ with the data from participant VPpboo_15_09_22 are presented in Figure 5.9. In the simulations that have been used to create the figure, the super-

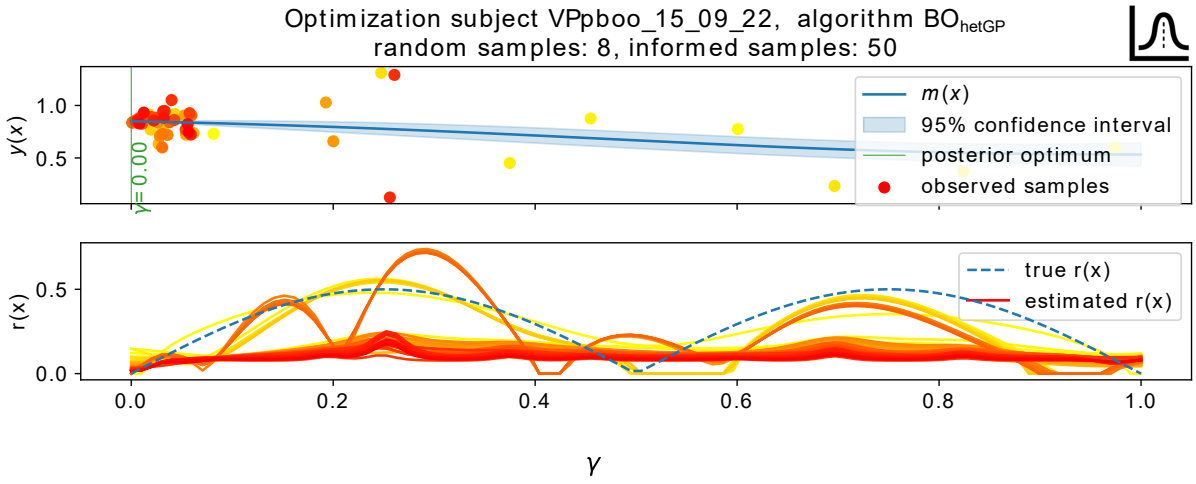


Figure 5.9: The posterior distribution that has been obtained from optimizing the parameter γ for the participant VPpboo_15_09_22. The top pane presents the posterior distribution of the Most likely heteroskedastic Gaussian process regressor. The distribution is accompanied by the coordinates of the observed data. The observed samples are coloured after the iteration wherein they have been evaluated. More specifically, the samples that have been evaluated during early iterations are yellow, after which the colour slowly turns red for further iterations. In the bottom pane, the true and estimated $r(x)$ are presented. The colour scheme for the estimated noise is the same as for the observed samples.

imposed version of the noise is used to ensure that $r(x)$ is known. Just like the other analyses of the objective function with heteroskedastic noise, the noise is modelled as $r(\gamma) = |sin(2\pi\gamma)|$, where $|\cdot|$ indicates the absolute value. In the upper pane of the figure, we can see that BO_{hetGP} tends to sample values of γ that are close to 0. Only in the initial runs, whereof the associated samples are coloured yellow, the algorithm tends to explore. Naturally, the exploratory behaviour is inherited from the initialization samples that are sampled according to a Sobol sequence. Yet, the majority of the later, more red, samples are concentrated at the left of the domain. The model posterior, which is based on all 58 samples, is relatively flat.

In the lower pane of Figure 5.9, the true noise around the objective function $r(x)$ is described by the dashed blue line. Furthermore, the noise that is estimated by the Gaussian process at each of the 50 iterations is plotted as well. The noise estimates are coloured according to the iteration index at which they have been obtained. The colour gradient flows from yellow (for low indices) to red (for high indices). Here, it is striking that the eventual noise estimates of the Gaussian process, described in red is much worse than the initial estimates, described in yellow. More specifically, it seems that the estimate of $r(x)$ is the most accurate when the model has just been initialized with the low-discrepancy Sobol sequences. Possibly, the high concentration of samples in a specific place seems to hamper the capacity of the model to estimate $r(x)$ in the remainder of the domain.

To put this hypothesis to the test, the simulation is run again with an augmented objective surface as introduced in Section 4.2.1, Equation 4.3 to shift the posterior optimum. The results of this experiment are presented in Figure 5.10. From the upper pane of Figure 5.10 it becomes clear that the algorithm now expresses a high interest in the region around $\gamma = 0.25$. This was to be expected as the optimum of the augmentation lies at $\frac{\pi/2}{2\pi} = 0.25$. Still, the value of the optimum is not exactly equal to 0.25, as the optimum might have shifted to the left compared to a sin wave due to the slightly downward slope that has been found for the original objective function $f(x)$.

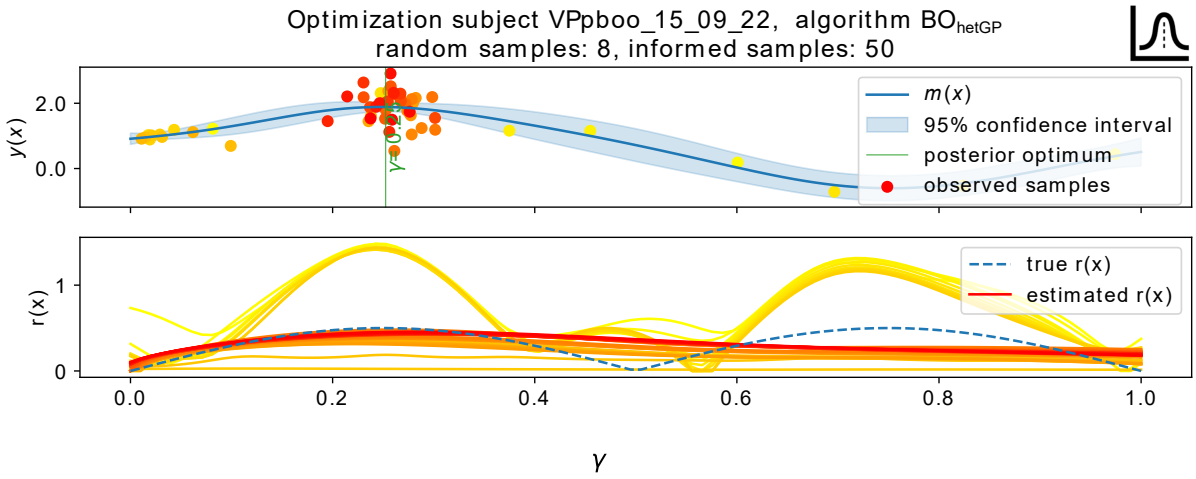


Figure 5.10: The posterior distribution and the estimated variance around the objective function from a simulated, augmented objective function. The results are presented as in Figure 5.9.

Furthermore, it is interesting to see that, in the lower pane of Figure 5.10, the $r(x)$ estimates from later iterations (in red) are more accurate than the estimates of earlier iterations (in yellow). Even though the latter estimates have approximately the right shape, they wildly overestimate the magnitude of the variance. In contrast, the magnitude of the estimation of $r(x)$ is more similar to the true $r(x)$ for later iterations, particularly for the region around $\gamma = 0.25$ where a lot of evaluations are made.

To assess the qualities of the posterior means of the three surrogate models, the optimization algorithms are tasked to optimize the function $f(\gamma) = sin(2\pi\gamma)$, whereof the optimum is known to lie exactly around $\gamma = 0.25$. The posterior distribution of the three models are presented below.

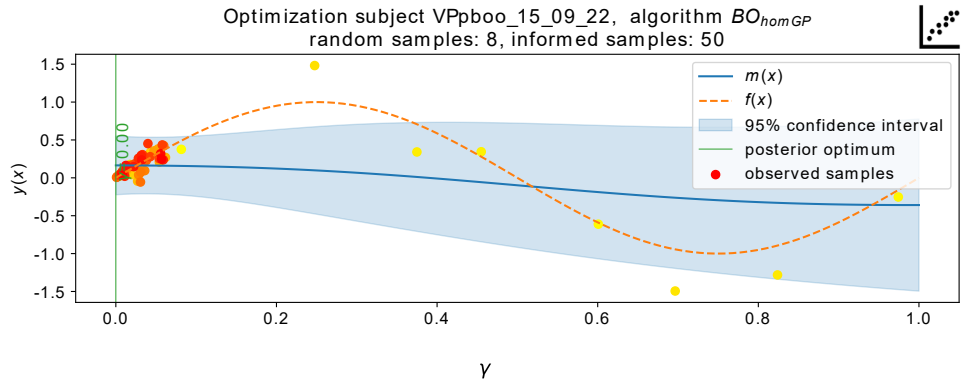


Figure 5.11: The posterior distribution of BO_{homGP}. The objective function $f(\gamma) = \mathcal{N}(sin(2\pi\gamma)$, is plotted with the dashed orange line. The optimum of $f(\gamma)$ lies at $\gamma = 0.25$. The noise around the objective function is modelled as $r(\gamma) = |0.5sin(2\pi\gamma)|$.

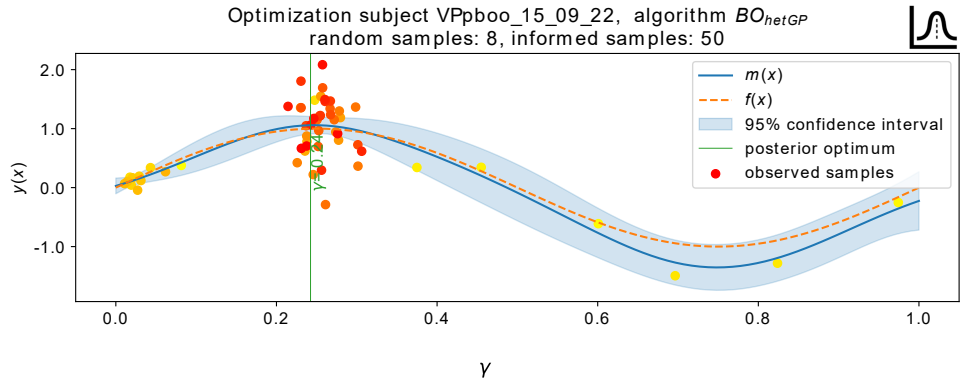


Figure 5.12: The posterior distribution of BO_{hetGP}. The objective function $f(\gamma) = \mathcal{N}(sin(2\pi\gamma)$, is plotted with the dashed orange line. The optimum of $f(\gamma)$ lies at $\gamma = 0.25$. The noise around the objective function is modelled as $r(\gamma) = |0.5sin(2\pi\gamma)|$.

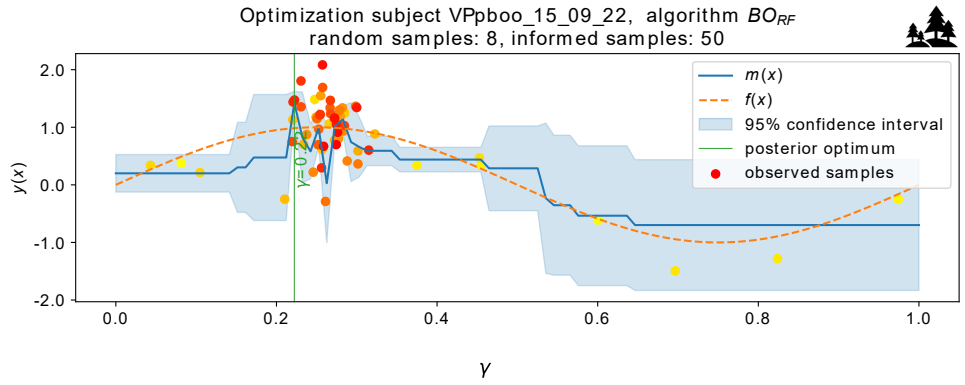


Figure 5.13: The posterior distribution of BO_{RF}. The objective function $f(\gamma) = \mathcal{N}(sin(2\pi\gamma)$, is plotted with the dashed orange line. The optimum of $f(\gamma)$ lies at $\gamma = 0.25$. The noise around the objective function is modelled as $r(\gamma) = |0.5sin(2\pi\gamma)|$.

From the figures, it becomes clear that the posterior distribution of the homoskedastic Gaussian process is the best estimate of the objective function. The optimum that has been modelled by BO_{hetGP} is with $\gamma = 0.25$ the closest to the true optimum. The random forest is the runner-up, with an optimum of $\gamma = 0.22$. Yet, the posterior of the random forest is harder to recognize as a sine wave as the estimated variance of the objective function seems to be too small in the region around the maximum, which results in a jittery posterior mean. Lastly, the homoskedastic Gaussian process has the least accurate fit. According to this model, the optimum lies around $\gamma = 0.0$. Furthermore, the shape of the posterior mean is too flat to be an accurate resemblance of $f(x)$.

5.3.3 Noise resilience

Now that the modelling qualities of the surrogate models have been discussed, I continue with the assessment of the noise resilience of the different models. Recall that for the generation of Figures 5.1-5.3 the objective noise was based on the sampling noise that comes with evaluating merely a subset of the available epochs. Initially, the size of this subset was set to 450 epochs.

In Figure 5.14 the performances of the four different algorithms are shown for subsets with 300, 150 and 50 epochs. The objective function that has been optimized is seven-dimensional, corresponding to the data from participant VPpboo_15_09_22. When comparing Figure 5.14 with the corresponding pane in Figure 5.3, it becomes clear that mostly the performances of BO_{RF} and RS are affected when the subset size shrinks from 450 epochs to 300 epochs. With a subset size of 150 epochs, the performance of all algorithms has worsened. Still, BO_{hetGP} and BO_{RF} seem to outperform RS. Lastly, with an even smaller subset size of 50 epochs, the performance of the informed strategies has decreased even further to the point where BO_{RF} is outperformed by the baseline, RS. Interestingly, the performance of BO_{hetGP} is rather low in the first few iterations of Figure 5.14c, whereafter the algorithm seems to be able to recover from the high level of noise that is associated with the objective function. The performance of BO_{homGP} stays stable under the different noise conditions.

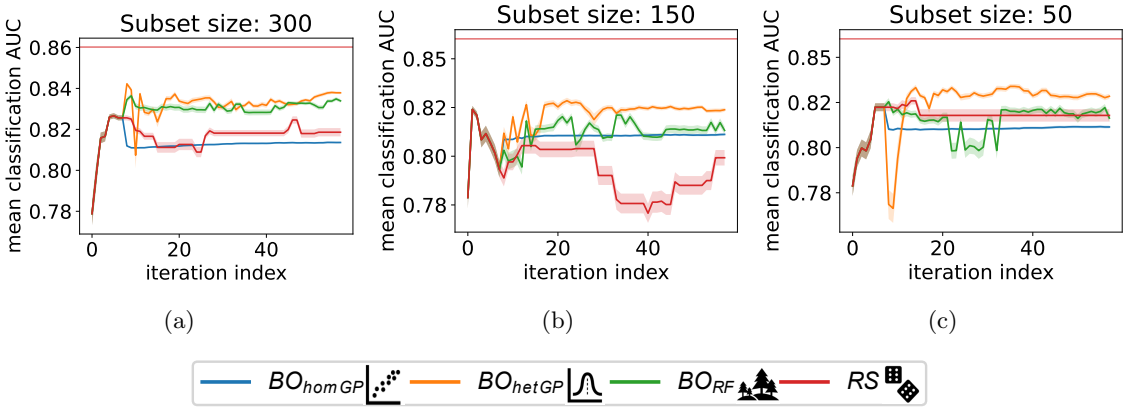
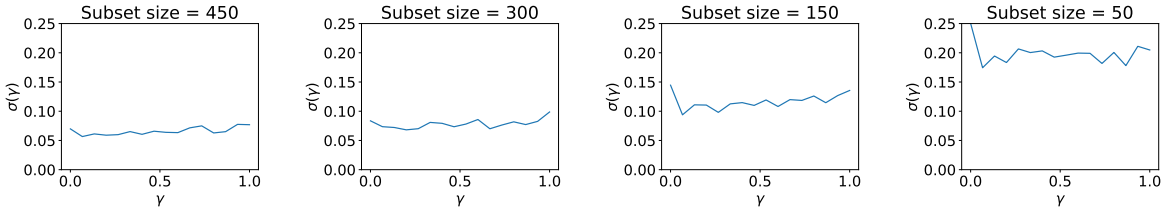


Figure 5.14: The performance of the algorithms BO_{hetGP}, BO_{RF} and RS under different noise conditions. The performance has been measured with the seven-dimensional objective function of participant VPpboo_15_09_22. The noise conditions are simulated by varying the size of the subset that is used to calculate the noisy ROC AUC scores. The exact subset size is annotated in the title of each pane.

Even though the size of the subset can provide an indication of the quantity of the noise, the exact magnitude of the noise is unknown. To still gain an insight in the quantity of the noise, 100 evaluations are made at 16 evenly spaced locations within the domain of the one-dimensional objective function. Next, the standard deviations of these distributions of 100 samples are calculated. The results are presented in Figure 5.15. The standard deviation that is associated with subset containing 450 epochs (Figure 5.3) lies around 0.06. Furthermore, the standard

deviation of the noise tends to increase when the size of the subset decreases. As a consequence, the standard deviations that correspond to distributions that have been created using the subset sizes 300, 150 and 50 lie respectively around 0.9, 0.11 and 0.17.



(a) (b) (c) (d)

Figure 5.15: The approximated standard deviations that are associated with the simulated objective noise. As the objective noise is introduced by evaluating subsets of the data, the true objective noise is unknown. Due to the limited number of 100 samples, the variance is not exactly the same for all values of γ. However, the variance is expected to converge to a single value if the number of samples increases. The objective noise has been estimated given a one-dimensional objective function. Yet, the magnitude of the noise is assumed to be independent of the dimensionality of the objective function.

Chapter 6

Discussion

Purpose of the chapter

In the previous chapter, the optimization performances of the four algorithms have been presented. The performance of BO_{hetGP} significantly outperforms the baseline RS for all objective functions that have been considered. Furthermore, the performance of BO_{RF} significantly outperforms RS for all but the one-dimensional objective function. Yet, the found results do not generalize to situations with a higher objective noise. On top of that, BO_{homGP} tends to perform worse than the baseline in almost all cases. Additionally, the superiority of the two other informed models cannot be confirmed for all participants from the Word ERP dataset. In the current chapter, the possible causes of the aforementioned observations are discussed. Moreover, the limitations to the approaches that have been used are presented.

6.1 Explaining the differences in performance between participants

The found optimization performances differ heavily between participants. However, for most participants that have been modelled in the one and two-dimensional optimization problem, there seems to be a general trend: BO_{hetGP} has the best performance, followed by BO_{RF}. Both models tend to outperform RS in most cases. However, this tendency is not observed with the participants VPpboa_15_08_11, VPpbos_15_10_09 and VPpboi_15_09_01 (the two-dimensional case). These observations could possibly be explained by considering the information that is contained in the data that is associated with these participants.

Firstly, the ROC AUC performance that has been calculated with the data from the first-mentioned participant is very close to 0.5. This indicates that the performance of LDA is very close to the classification performance of making a random guess. Perhaps the recorded data is not informative of the class label. If this is the case, then there is little use tuning the parameters of the classification algorithm. As a consequence, the performances of the three algorithms are comparable, despite the presumed advantages of the informed algorithms BO_{hetGP} and BO_{RF} over RS.

Secondly, when trying to explain the results for the one and two-dimensional objective function of participant VPpbos_15_10_09, it is important to note that the value of γ is assumed to influence the classification performance of the classification algorithm. Naturally, if γ is equal to 1, then $\tilde{\Sigma} = \nu I$ (see Equation 4.2), which implies that this part of the training procedure of LDA is not based on the data. Thus, a choice of $\gamma = 1$ is unlikely to yield a high optimization score. However, there might be a lot of values for γ that yield an equally adequate or good classification performance. The latter hypothesis is reflected in the spread of the distribution of the values chosen by BO_{hetGP}, BO_{RF} and RS for γ that is shown in Figure 6.1. Note that the spread of the chosen values of both $\gamma \in [0, 1]$ and $t_0 \in [0.03, 0.1]$s cover the full domains of these parameters. In the latter scenario, it does not matter much what value of γ the optimization algorithm proposes, as long as the proposals are not too extreme.

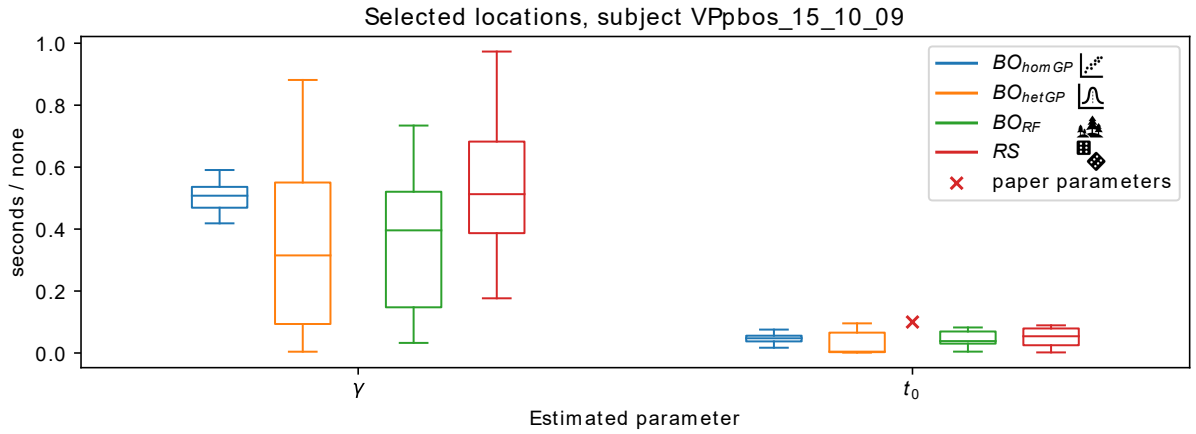


Figure 6.1: The distributions of chosen parameter values for the 10 different runs of BO_{hetGP} and BO_{RF} on the seven-dimensional optimization problem based on participant VPpbos_15_10_09. The paper parameters refer to the parameterization that is used by Sosulski and Tangermann (2022).

Lastly, the observed scores for the two-dimensional objective function for VPpboi_15_09_01 could be caused by a combination of the above-mentioned reasons. However, mainly the performance of BO_{hetGP} has dropped when comparing the one-dimensional objective function to the two dimensional case. It is not clear why the Gaussian process regressor is so particularly affected by this change in dimensionality.

6.2 The effect of the objective noise on the model performance

In Section 5.3.3, it was demonstrated that the performance of BO_{hetGP} and BO_{RF} drops when the modelled noise (described by the standard deviation) increases. This observation has also been made by Sosluski et al; who observed that the optimization process was impeded by a low objective-to-noise ratio (Sosulski et al., 2022). However, this drop mainly seems to affect BO_{RF} for the subset with 50 elements, indicating that both BO_{hetGP} and BO_{RF} are rather resilient to noise levels below $r(x) = 0.10$. It is noteworthy that RS seems less affected by higher noise levels. Namely, the final verdict that is yielded by RS is the location that is associated with the highest evaluated outcome. As a consequence, the algorithm can be fooled into selecting relatively low true ROC AUC values that are inflated by a high variance. It is more clear why the introduction of more noise negatively affects the performance of the informed methods, as the fit of the surrogate models could be negatively affected by the large quantities of noise, leading to biased verdicts.

6.2.1 The quality of the noise estimation by BO_{hetGP}

The noise estimation of the Most likely heteroskedastic Gaussian process is an impressive feature. Unfortunately, this feature does not seem to work optimally in combination with Bayesian optimization, as observed in Figure 5.9. In particular when the optimization algorithm is mainly exploiting existing solutions ($\beta < 0.5$), the estimation of $r(x)$ is likely to be hampered. Namely, with a focus on exploitation, the optimization algorithm is expected to make a lot of samples in a relatively small (promising) region. As a consequence, there is a lot of information available for the estimation of the uncertainty in that particular region. However, the estimated variance in the remainder of the input space is less-informed, which makes the estimate suboptimal. To make this problem less severe, more initialization samples, that are evenly spaced throughout the input space can be queried.

6.2.2 Investigating the low performance of BO_{homGP}

It is remarkable that the informed method BO_{homGP} performs worse than the random sampling baseline. In theory, BO_{homGP} should be able to adequately model the objective function if the noise around the objective is homoskedastic. In practice however, the surrogate model seems to encourage the optimization algorithm to exclusively sample around a local optimum, as demonstrated in Figure 5.11. This could also be the reason why the optimization traces of BO_{homGP} tend to flatten after the initialization phase, as demonstrated in Figures 5.1-5.3. It is possible that the value of the exploration-exploitation tradeoff parameter β, which has been set in Appendix A, is suboptimal for the homoskedastic Gaussian process. As the estimate of the posterior variance of the homoskedastic Gaussian process is more crude than the estimate of the heteroskedastic variant[1], the acquisition function of BO_{homGP} is less well-informed than the acquisition of BO_{hetGP}. To assess whether BO_{homGP} would have benefited from a more exploratory acquisition strategy, Figure 5.11, where $\beta = 0.187$ is used, is recreated with $\beta = 0.8$. The results are shown in Figure 6.2 below. While the location of the posterior optimum is more adequate (recall that the true optimum lies at $\gamma = 0.25$), the fit of the model is still not as good as the posteriors from BO_{hetGP} and BO_{RF} shown in Figure 5.12 and 5.13. This could also suggest that the remarkably low performance of the single, homoskedastic Gaussian process is caused by the relatively low capacity of the model.

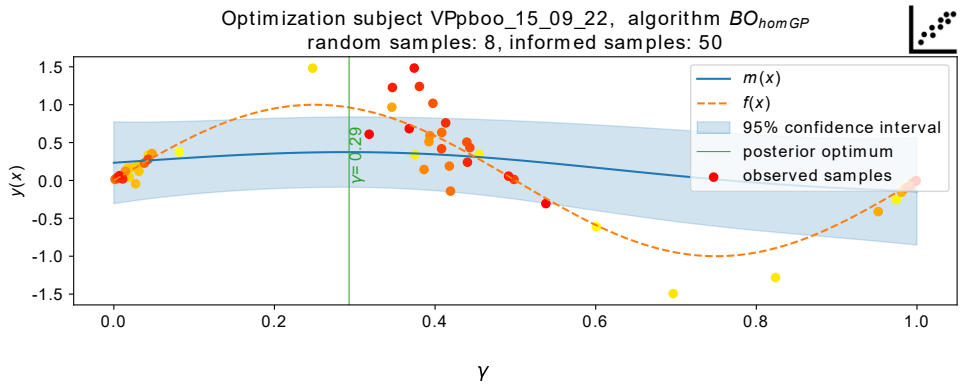


Figure 6.2: The posterior distribution of BO_{RF} as modelled in figure 5.11. The exploration-exploitation tradeoff parameter β has been set to 0.8 to favour exploration.

6.3 Limitations

Now that the unanticipated patterns that have been found in the results have been discussed, I move on to the review of the limitations of this work. Section 6.3.1 starts with the discussion of the discrepancy between the observed traces and the found significance scores. Furthermore, to emphasize that this is not the only class of optimization algorithms, I will present some alternative approaches in Section 6.3.2.

6.3.1 Validity of the evaluation metrics

The differences between the optimization traces of the seven-dimensional objective function (Figure 5.3) seem to be smaller than those of the one and two-dimensional objective function (Figures 5.2 and 5.2). Yet, the p-values that have been calculated for the differences between the optimization algorithms and the random search method are the smallest for the seven-dimensional objective function (Table 5.4). This discrepancy could originate from the metrics

[1]The estimate of the posterior variance of the homoskedastic Gaussian process is more crude than the estimate of the heteroskedastic Gaussian process as the latter model has more flexibility in estimating $r(x)$ for each input x as opposed to a single estimate of $\sigma_n^2 oise$.

that have been used to evaluate the optimization processes. That is, the metrics merely consider the highest scores that have been found by each optimization algorithm. As a consequence, an outlier of an optimization algorithm that is generally performing less well than other algorithms, could turn the favour towards the first-mentioned strategy. Recall that for the calculation of the p-values, an attempt has been made to address this problem by smoothing the "best-found" scores for each participant by taking the average over the 10 runs. Yet, the found results do not guarantee that the Bayesian optimization algorithms always outperform random search.

6.3.2 Alternative optimization approaches

Even though the solutions to hyperparameter optimization tasks have been dominated by Bayesian optimization methods, there are other algorithms that are worth investigating, such as hyperband (Li et al., 2017), evolutionary algorithms (Bäck & Schwefel, 1993) and BOHB, a combination of the Bayesian optimization (BO) and hyperband (HB) algorithms (Falkner et al., 2018). From these algorithms, in particular hyperband and BOHB are assumed to be more efficient than Bayesian optimization.

6.4 From a simulation to the real world

To discover how the results found roughly translate to a real-world experiment, there are three aspects to take into account: the definition of the evaluation of the objective function, the number of iterations that are used to optimize the algorithm and the runtime of a single iteration. Consider that the objective function is evaluated as done by Sosulski et al. (2022). That is, for each evaluation, the proposed stimulus characteristics are tested on full trials of 90 tones. Say that the duration of a single trial is equal to about 36 seconds[2]. Taking a 8s pause between each trial, similar to Sosulski et al. (2022), the duration of a single trial totals 44 seconds. The runtime of a single iteration of all algorithms is below one second. As a consequence, a new suggestion can be made during the inter-trial pause. Next, in particular BO_{hetGP} was found to have an optimization curve which starts with a steep hill, only to flatten after roughly 30 iterations for most participants, noise types and dimensionalities. Thus, if the optimization algorithm is run for 30 iterations, the total duration of the parameter optimization per participant is $30 \times 44 = 1320$ seconds, which is equal to 22 minutes[3].

The choice of optimization algorithm depends on the number of parameters that is optimized and the noise type that is encountered. To assess the latter, one could analyze the EEG resting state recordings. Given the results from Table 5.1 and 5.2, BO_{RF} is the preferred method for high-dimensional objective functions that are polluted with homoskedastic noise. However, as the performance of the BO_{RF} degrades when the magnitude of the noise increases, the preference shifts towards BO_{hetGP} if the standard deviation that is associated with the objective noise is above 0.11. Furthermore, BO_{hetGP} seems to be the preferred model if heteroskedastic noise is encountered or a small number (one or two) of parameters is optimized. The proposed strategy is summarized in Figure 6.3.

[2]As the SOA could be one of the parameters of interest, it is difficult to give an estimate of the duration of the trial. Therefore, this estimation is based on the 400ms SOA that has been used by Sugi et al. (2018), plus the duration of the stimulus, which is 40ms (Sosulski et al., 2022). This yields a trial duration of $\frac{90 \times 400}{1000} = 36$ seconds.

[3]The times presented here differ highly between paradigms. For example, the duration of a single epoch of a code-modulated visual evoked potential (c-VEP) paradigm is much shorter than the duration of an epoch of a ERP paradigm. Thus, during a c-VEP, a lot more samples can be evaluated.

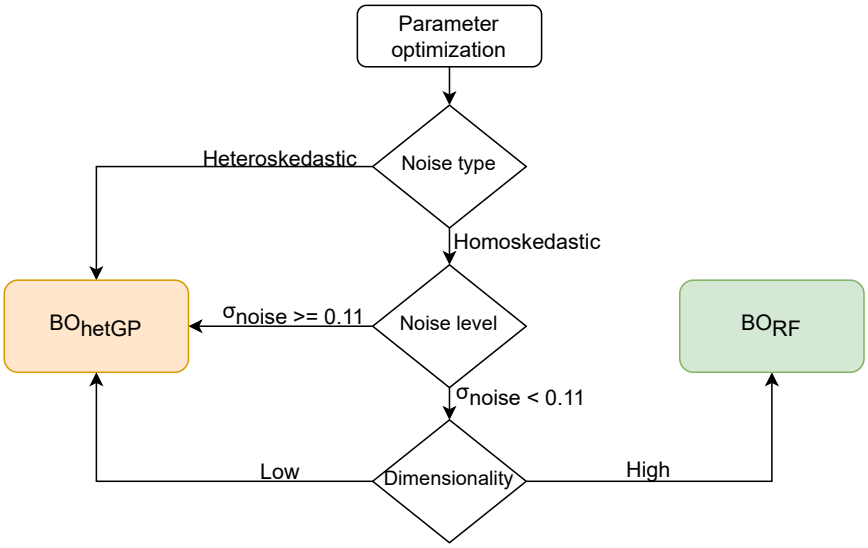


Figure 6.3: The proposed strategies given the different characteristics of the optimization problem.

Lastly, it is worthwhile to discuss whether the results that have been found with the simulated optimization problems are likely to generalize to the online optimization of a real experiment. That is, there are several limitations to the simulated objective functions. Firstly, it cannot be guaranteed that the modelled objective noise is comparable to the noise that can be found during an ERP paradigm. Secondly, the simulations to not account for the non-stationarities that can be encountered in an experimental setting. In particular the decline of attention or the habituation to the task could have a major impact on the classification performance of the decoding algorithm (Ravden & Polich, 1999). As a consequence, there can be a shift in the mean of the outcomes that have been obtained during a later state of the optimization. Hypothetically, one could regard the noise r as a two-dimensional function. Instead of modelling the heteroskedacity as a function of merely the location of the proposed sample, x, the point in time t at which it evaluation is made should also be considered, $r(x, t)$. Lastly, if the stimulus characteristics of an experimental paradigm are optimized, then the participants are presented with a lot of different stimuli. This change in stimulus characteristics could be very distracting, which makes it harder for the participant to concentrate or get into the flow of the task. This too, can have a negative effect on the performance of the classification algorithm, which in its turn (negatively) influences the optimization process.

6.5 Relation to automated machine learning

The application of the findings that have been presented in this thesis is not limited to the BCI domain. Bayesian optimization techniques are, among others, of interest within the more general field of AutoML, which focuses on the automation of the application of machine learning algorithms (Kurian et al., 2021). The optimization of machine learning hyperparameters, as done in this thesis, is an important step in the automated application of machine learning algorithms (Thornton et al., 2013). As a consequence, the results found could also be relevant to applications of AutoML.

Chapter 7

Conclusion

To improve the optimization process of objective functions that are subject to presumably non-iid noise, I propose to enrich Bayesian optimization with heteroskedastic modelling and the evaluation of replicated samples. In particular, complementing the optimization algorithm with heteroskedastic Gaussian process regression and the location-based replication of existing designs significantly outperforms the random sampling baseline for all objective functions that have been considered. Furthermore, the heteroskedastic Gaussian process can be used to estimate the noise that is associated with the objective function, which provides additional insights in the objective to noise ratio of each participant. Yet, there is still room for improvement as the exploitative nature of the optimization algorithm hinders the noise estimation of the model. Perhaps this problem could be made less severe in further work, by making more uniformly-spaced initialization samples to better inform the noise estimation.

The objective function can also be modelled adequately by random forest regression, a faster, less elaborate model. This model is simpler than the Gaussian process as it does not separately estimate the noise that is associated with the objective function. Nevertheless, the performance of the optimization algorithm that makes use of the random forest seems to scale better to higher-dimensional optimization problems. Regrettably, the performance of this algorithm seems to break down when the magnitude of the objective noise increases or when the noise is strongly non-iid.

This work does not present the only solution to optimizing objective functions in heteroskedastic settings as merely Bayesian optimization algorithms have been considered. Possibly, better results can be obtained with other optimization methods such as evolutionary computing and hyperband. Besides, the results found are not guaranteed to generalize to a real world BCI scenario due to the differences between the simulated and the real noise and the presence of non-stationarities that have not been modelled. Yet, the application of the results is not limited to the BCI domain. Naturally, the findings that have been presented in this thesis are applicable to other fields that suffer from the presence of heteroskedastic noise. More specifically, as the methods from this thesis are focused around the optimization of hyperparameters that regulate classification process, the work can be related to AutoML and black-box optimization within other fields that suffer from high noise levels.

Thus, returning to the research question: how can optimization algorithms be made resilient against the high level of potentially heteroskedastic noise?

It can be concluded that, in particular in the presence of heteroskedastic noise, heteroskedastic Gaussian process regression and location-based replications can be used within Bayesian optimization to better model the objective function and the associated noise.

In further work, more attention could be devoted to researching the effect of different noise types on the optimization performance. Furthermore, the line of research can be continued by implementing a real-world BCI paradigm that features the online optimization of stimulus characteristics to assess whether the findings generalize to the real world.

Bibliography

Aggarwal, C. C., Hinneburg, A., & Keim, D. A. (2001). On the Surprising Behavior of Distance Metrics in High Dimensional Space. In J. Van den Bussche & V. Vianu (Eds.), *Database theory — icdt 2001* (pp. 420–434). Springer Berlin Heidelberg.

Allison, B., & Pineda, J. (2006). Effects of SOA and flash pattern manipulations on ERPs, performance, and preference: Implications for a BCI system. *International journal of psychophysiology : official journal of the International Organization of Psychophysiology*, *59*, 127–40. https://doi.org/10.1016/j.ijpsycho.2005.02.007

Ankenman, B., Nelson, B. L., & Staum, J. (2008). Stochastic kriging for simulation metamodeling. *2008 Winter simulation conference*, 362–370.

Bäck, T., & Schwefel, H.-P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, *1*(1), 1–23.

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., & Bakshy, E. (2020). BOTORCH: A Framework for Efficient Monte-Carlo Bayesian Optimization. *Proceedings of the 34th International Conference on Neural Information Processing Systems*.

Binois, M., & Gramacy, R. (2021). hetGP : Heteroskedastic Gaussian Process Modeling and Sequential Design in R. *Journal of Statistical Software*, *98*. https://doi.org/10.18637/jss.v098.i13

Binois, M., Huang, J., Gramacy, R. B., & Ludkovski, M. (2019). Replication or exploration? Sequential design for stochastic simulation experiments. *Technometrics*, *61*(1), 7–23.

Birbaumer, N., Ghanayim, N., Hinterberger, T., Iversen, I., Kotchoubey, B., Kübler, A., Perelmouter, J., Taub, E., & Flor, H. (1999). A spelling device for the paralysed. *Nature*, *398*, 297–298. https://doi.org/10.1038/18581

Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning* (Vol. 4). Springer New York, NY.

Blankertz, B., Lemm, S., Treder, M., Haufe, S., & Müller, K.-R. (2011). Single-trial analysis and classification of ERP components — A tutorial [Multivariate Decoding and Brain Reading]. *NeuroImage*, *56*(2), 814–825. https://doi.org/https://doi.org/10.1016/j.neuroimage.2010.06.048

Booker, A. J., Dennis, J. E., Frank, P. D., Serafini, D. B., Torczon, V., & Trosset, M. W. (1999). A rigorous framework for optimization of expensive functions by surrogates. *Structural optimization*, *17*, 1–13. https://doi.org/10.1007/BF01197708

Breiman, L. (1984). *Classification and regression trees* (1st ed.). Routledge.

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*, 123–140. https://doi.org/https://doi.org/10.1007/BF00058655

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Brochu, E., Cora, V., & Freitas, N. (2010). A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. *CoRR*, *abs/1012.2599*. https://doi.org/https://doi.org/10.48550/arXiv.1012.2599

Cavagnaro, D. R., Myung, J. I., Pitt, M. A., & Kujala, J. V. (2010). Adaptive design optimization: A mutual information-based approach to model discrimination in cognitive science. *Neural computation*, *22*(4), 887–905.

Constrained Bayesian Optimization with Noisy Experiments, author=Benjamin Letham and Brian Karrer and Guilherme Ottoni and Eytan Bakshy. (2018).

Denzer, S. (2016). *Influence of Spatial Word Presentation in an Auditory ERP Paradigm for Brain-Computer Interfaces* (Master's thesis). Albert-Ludwigs-University Freiburg. Georges-Köhler-Allee 73, 79110 Freiburg i. Br.

Dewancker, I., McCourt, M., Clark, S., Hayes, P., Johnson, A., & Ke, G. (2016). A stratified analysis of Bayesian optimization methods. *arXiv preprint arXiv:1603.09441*.

Falkner, S., Klein, A., & Hutter, F. (2018). BOHB: Robust and efficient hyperparameter optimization at scale. *International conference on machine learning*, 1437–1446.

Frazier, P. I. (2018). A tutorial on Bayesian optimization. *ArXiv*. https://doi.org/arXiv:1807.02811

Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q., & Wilson, A. G. (2018). GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration. *Advances in Neural Information Processing Systems*.

Goldberg, P., Williams, C., & Bishop, C. (1997). Regression with Input-dependent Noise: A Gaussian Process Treatment. In M. Jordan, M. Kearns, & S. Solla (Eds.), *Advances in neural information processing systems*. MIT Press.

Gonsalvez, C., Barry, R., Rushby, J., & Polich, J. (2007). Target-to-target interval, intensity, and P300 from an auditory single-stimulus task. *Psychophysiology*, *44*, 245–50. https://doi.org/10.1111/j.1469-8986.2007.00495.x

Gui, X., Chuansheng, C., Zhong-Lin, L., & Qi, D. (2010). Brain Imaging Techniques and Their Applications in decision-Making Research. *Xin li xue bao. Acta psychologica Sinica*, *42*(1), 120–137. https://doi.org/10.3724/SP.J.1041.2010.00120

Höhne, J., & Tangermann, M. (2012). How stimulation speed affects Event-Related Potentials and BCI performance. *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 1802–1805. https://doi.org/10.1109/EMBC.2012.6346300

Holm, S. (1979). A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics*, *6*(2), 65–70.

Hutter, F., Xu, L., Hoos, H. H., & Leyton-Brown, K. (2014). Algorithm Runtime Prediction: Methods & Evaluation. *Artificial Intelligence*, *206*, 79–111. https://doi.org/https://doi.org/10.1016/j.artint.2013.10.003

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, *13*(4), 455. https://doi.org/10.1023/A:1008306431147

Kersting, K., Plagemann, C., Pfaff, P., & Burgard, W. (2007). Most Likely Heteroscedastic Gaussian Process Regression. *Proceedings of the 24th International Conference on Machine Learning*, 393–400. https://doi.org/10.1145/1273496.1273546

Kolb, B., Whishaw, I. Q., & Teskey, G. C. (2016). *An introduction to brain and behaviour* (5th ed.). Worth Publishers.

Kurian, J. J., Dix, M., Amihai, I., Ceusters, G., & Prabhune, A. (2021). BOAT: A Bayesian Optimization AutoML Time-series Framework for Industrial Applications. *2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService)*, 17–24. https://doi.org/10.1109/BigDataService52369.2021.00008

Kushner, H. J. (1964). A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise. *Journal of Basic Engineering*, *86*, 97–106. https://doi.org/10.1115/1.3653121

Lam, R., Willcox, K., & Wolpert, D. (2016). Bayesian optimization with a finite budget: An approximate dynamic programming approach. *Advances in Neural Information Processing Systems*, 883–891.

Lázaro-Gredilla, M., & Titsias, M. K. (2011). Variational Heteroscedastic Gaussian Process Regression. *Proceedings of the 28th International Conference on International Conference on Machine Learning*, 841–848.

Le, Q. V., Smola, A. J., & Canu, S. (2005). Heteroscedastic Gaussian process regression. *Proceedings of the 22nd international conference on Machine learning*, 489–496. https://doi.org/10.1145/1102351.1102413

Ledoit, O., & Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of multivariate analysis*, *88*(2), 365–411.

Lewi, J., Butera, R., & Paninski, L. (2009). Sequential optimal design of neurophysiology experiments. *Neural computation*, *21*(3), 619–687.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *The journal of machine learning research*, *18*(1), 6765–6816.

Long, J., Li, Y., Wang, H., Yu, T., Pan, J., & Li, F. (2012). A hybrid brain computer interface to control the direction and speed of a simulated or real wheelchair. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *20*(5), 720–729. https://doi.org/10.1109/TNSRE.2012.2197221

Lorenz, R., Monti, R. P., Violante, I. R., Anagnostopoulos, C., Faisal, A. A., Montana, G., & Leech, R. (2016). The Automatic Neuroscientist: A framework for optimizing experimental design with closed-loop real-time fMRI. *NeuroImage*, *129*, 320–334. https://doi.org/https://doi.org/10.1016/j.neuroimage.2016.01.032

McFarland, D. J., & Wolpaw, J. R. (2008). Brain-Computer Interface Operation of Robotic and Prosthetic Devices. *Computer*, *41*(10), 52–56. https://doi.org/10.1109/MC.2008.409

Močkus, J. B., & Močkus, L. J. (1991). Bayesian Approach to Global Optimization and Application to Multiobjective and Constrained Problems. *J. Optim. Theory Appl.*, *70*(1), 157–172.

Močkus, J. (1975). On Bayesian methods for seeking the extremum. *Optimization Techniques IFIP Technical Conference: Novosibirsk, July 1–7, 1974*, 400–404.

Musso, M., Hübner, D., Schwarzkopf, S., Bernodusson, M., LeVan, P., Weiller, C., & Tangermann, M. (2022). Aphasia recovery by language training using a brain–computer interface: a proof-of-concept study. *Brain Communications*, *4*(1). https://doi.org/10.1093/braincomms/fcac008

Myung, J., Pitt, M., Tang, Y., & Cavagnaro, D. R. (2009). Bayesian adaptive optimal design of psychology experiments. *Proceedings of the 2nd International Workshop in Sequential Methodologies (IWSM2009)*, 1–6.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pillow, J. W., & Park, M. (2016). Adaptive Bayesian methods for closed-loop neurophysiology. In A. El Hady (Ed.), *Closed loop neuroscience* (pp. 3–18). Elsevier.

Quadrianto, N., Kersting, K., Reid, M. D., Caetano, T. S., & Buntine, W. L. (2009). Kernel conditional quantile estimation via reduction revisited. *2009 Ninth IEEE International Conference on Data Mining*, 938–943. https://doi.org/10.1109/ICDM.2009.82

Rai, A., Antonova, R., Meier, F., & Atkeson, C. G. (2019). Using simulation to improve sample-efficiency of Bayesian optimization for bipedal robots. *The Journal of Machine Learning Research*, *20*(1), 1844–1867.

Ravden, D., & Polich, J. (1999). On P300 measurement stability: habituation, intra-trial block variation, and ultradian rhythms. *Biological psychology*, *51*(1), 59–76.

Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, *58*(5), 527–535.

Smith, B. J., Yan, J., & Cowles, M. K. (2008). Unified Geostatistical Modeling for Data Fusion and Spatial Heteroskedasticity with R Package ramps. *Journal of Statistical Software*, *25*(10), 1–21. https://doi.org/10.18637/jss.v025.i10

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, *25*. https://doi.org/10.48550/arXiv.1206.2944

Sobol', I. (1967). On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, *7*(4), 86–112. https://doi.org/https://doi.org/10.1016/0041-5553(67)90144-9

Sosulski, J., Hübner, D., Klein, A., & Tangermann, M. (2022). Online Optimization of Stimulation Speed in an Auditory Brain-Computer Interface under Time Constraints. https://doi.org/10.48550/ARXIV.2109.06011

Sosulski, J., & Tangermann, M. (2022). Introducing block-Toeplitz covariance matrices to remaster linear discriminant analysis for event-related potential brain–computer interfaces. *Journal of Neural Engineering*, *19*(6), 066001. https://doi.org/10.1088/1741-2552/ac9c98

Srinivas, N., Krause, A., Kakade, S., & Seeger, M. (2010). Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design, 1015–1022.

Stein, M. L. (1999). *Interpolation of spatial data: Some theory for kriging*. Springer Science & Business Media.

Sugi, M., Hagimoto, Y., Nambu, I., Gonzalez, A., Takei, Y., Yano, S., Hokari, H., & Wada, Y. (2018). Improving the Performance of an Auditory Brain-Computer Interface Using Virtual Sound Sources by Shortening Stimulus Onset Asynchrony. *Frontiers in Neuroscience*, *12*. https://doi.org/10.3389/fnins.2018.00108

Therneau, T., & Atkinson, E. (1997). An introduction to Recursive Partitioning Using the RPART Routines. *Mayo Clinic*, *61*.

Thornton, C., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2013). Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 847–855. https://doi.org/10.1145/2487575.2487629

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Williams, C. K., & Rasmussen, C. E. (2006). *Gaussian processes for machine learning* (Vol. 2). MIT press Cambridge, MA.

Wolpaw, J. R. (2007). Brain-computer interfaces (BCIs) for communication and control. *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility*, 1–2.

Zilinskas, A. (1978). Algorithm AS 133: Optimization of One-Dimensional Multimodal Functions. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *27*(3), 367–375.

List of Abbreviations

AutoML automated machine learning. 7, 60, 61

BCI brain-computer interfaces. 6, 7, 8, 22, 23, 24, 25, 26, 30, 32, 33, 60, 61, 67, 68

BO Bayesian optimization. 7, 11

c-VEP code-modulated visual evoked potential. 59

EEG electroencephalography. 6, 7, 10, 26, 59

ERP event-related potential. 6, 7, 10, 11, 23, 25, 26, 29, 36, 39, 41, 42, 43, 45, 46, 47, 49, 56, 59, 60

fMRI functional magnetic resonance imaging. 23

HSV hue, saturation, value. 24

iid independent and identically distributed. 7, 15, 16, 61

IMSPE integrated mean squared prediction error. 20, 21, 33, 34

LDA linear discriminant analysis. 26, 27, 28, 30, 56

MAP maximum a posteriori. 16

MCMC Markov Chain Monte Carlo. 16

MLE maximum likelihood estimation. 32

MSE mean squared error. 35

ONR objective to noise ratio. 24

ROC AUC area under the receiver operating characteristic curve. 4, 23, 25, 26, 27, 28, 30, 38, 44, 56, 57, 68, 69, 70, 71, 72, 73, 75

SEM standard error of the mean. 40

SNR signal-to-noise ratio. 6, 7

SOA stimulus onset asynchrony. 23, 24, 59

UCB upper confidence bound. 19

Appendix A

Fine-tuning the architecture of the Bayesian optimization algorithm

A.1 Evaluating different Bayesian optimization architectures

In Chapter 4, it becomes clear that there are multiple ways to define and parameterize a Bayesian optimization algorithm. Naturally, some architectures or parameter combinations might be more suited to the problem at hand than others. To find out what parametrization is appropriate for the simulated optimization of BCI parameters, different Bayesian optimization architectures are tested and evaluated. The results of the evaluations are presented in this chapter. Note that these evaluations are not expected to lead to a strong support for certain parameter combinations. Rather the results are used to provide the optimization algorithm with a slightly more informed initialization than a random choice of parameters.

The evaluation of the architectures is based on the metrics that have been discussed in Section 4.5.1. The tested architectures differ in the balance that is struck between exploration and exploitation, the replicator that is used and the convergence measure that is applied. These three aspects of the algorithm, which will be referred to as "components", are optimized in sequential steps:

1. The balance between the exploration and exploitation is set by tuning the β parameter of the acquisition function.
2. Given the chosen β, a replicator is selected.
3. Using the selected β and replicator, the convergence measure is chosen.

By sequentially optimizing the three components, one naively makes the assumption that the different components of the algorithm do not interact. While this assumption is clearly wrong[1], it is very time-consuming to test and evaluate each combination of components. Moreover, doing so would give rise to overfitting. Therefore, an effort has been made to decouple the components as much as possible, as discussed in the next section.

Decoupling the components

From the three components, the acquisition function and the replicator are heavily entwined in particular. Their dependence goes both ways. Firstly, the verdict of the replicator is governed by the exploration-exploitation strategy that is applied. This is because the replicator considers the sample that is proposed by the acquisition function. To illustrate, recall the two replicators that have been introduced in Section 4.3.4. From these replicators, the variance replicator is expected to make a lot of replications if exploitation is preferred, since all proposed locations are close to the sample that is associated with the highest evaluated outcome. In contrast, the sequential replicator, which aims to minimize the sampling uncertainty, might overrule exploiting

[1] This is explained in more detail in Section A.1.

proposals in favour of replicates that support more exploratory behaviour.

Secondly, the acquisition function benefits in its turn from the information that is provided by the replications. The replications are presumed to shed a light on the noise around the objective function. Therefore, the presence of replicates could improve the estimate of the posterior variance. As the acquisition function directly depends on the variance (see Equation 4.5), a better estimate of the posterior variance is expected to lead to a better-informed acquisition function.

In an attempt to separate the first two optimization steps, β is tuned in a situation where there is a lot of information about the noise around the objective function available. To model the aforementioned circumstances, each location that is proposed by the acquisition function is evaluated five times, as if there is an overzealous replicator at work. Provided with this well-informed posterior variance, the β that corresponds to the best performing algorithm is hypothesized to represent the optimal exploration-exploitation tradeoff. Following the selection of β, the two replicators are tested to discover what replicator is the best replacement for their (too expensive) overzealous counterpart. The three selection procedures are discussed in more detail in the following paragraphs.

Finding β

As β is a continuous parameter that influences the performance of the optimization algorithm, the optimization of β can be regarded as a black box problem on its own. In this problem, the classification ROC AUC scores that are found by the Bayesian optimization algorithm that is parameterized with β should be maximized. As a consequence, β could be optimized using a second, simpler Bayesian optimization process.

The double optimization procedure that has been used is shown in Figure A.1. To avoid ambiguity, the following naming scheme is used to make a distinction between the two processes and their parameters:

- The Bayesian optimization process that tunes the parameters of the BCI classification pipeline will be referred to as BO1. The parameters of this process are subscripted with a "1". For example, β_1 is used to indicate the β parameter of BO1.
- The Bayesian optimization process that is used to optimize β_1 will be referred to as BO2. The parameters of this process are subscripted with a "2", in a similar fashion as the parameters of BO1.

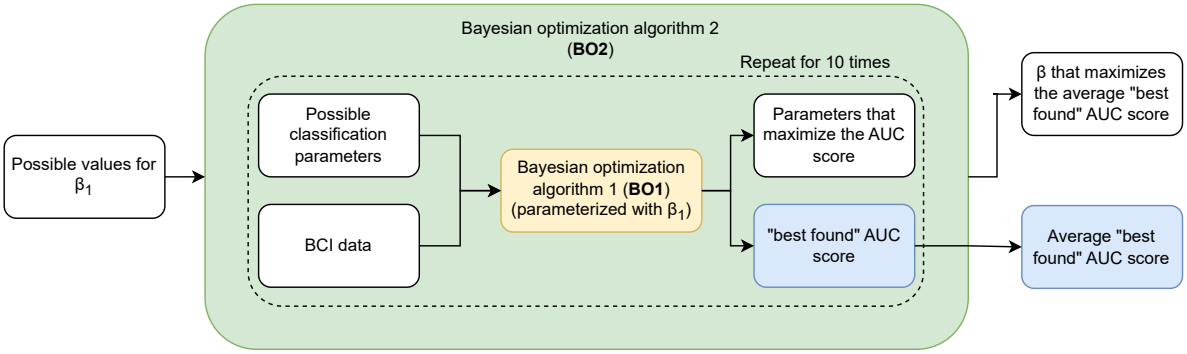


Figure A.1: An overview of the nested Bayesian optimization process that is used to find the optimal value for β_1 in BO1, the inner process. BO1 is the optimization algorithm that eventually will be used to optimize the classification parameters given the simulated BCI data. In order to produce a more informed estimate of the optimization performance that relates to a particular β_1, BO1 is repeated 10 times with different initialization samples.

In essence, the performance of BO1 can be seen as the objective function of BO2. To model this, BO1 is executed for each evaluation of the value of β_1 that is proposed by the acquisition function of BO2. It is important to note that the outcomes of BO1 have a probabilistic nature due to the randomness in the initialization samples of the optimization algorithm. Therefore, BO1 is run repeatedly with ten different sets of initialization samples to obtain a better estimate of its performance, as done in Dewancker et al., 2016. The average "best found" ROC AUC score over these ten runs is used as the objective for BO2.

BO2 has been parameterized with no replicator, no convergence measure and $\beta_2 = 0.2$ to keep the algorithm simple. For each of the ten runs, BO2 has been initialized using 11 random samples on the domain [0, 1]. Next, the algorithm makes 5 informed iterations. The variance within the classification ROC AUC scores that are calculated for the ten different runs of BO1 is expected to be dependent on the value of β_1. That is, when β_1 is close to 1, which facilitates exploration, the "best-found" ROC AUC scores may differ more than the scores that are found with β_1 values that encourage exploitation. Therefore, the Most likely heteroskedastic Gaussian process model, which is able to model data with heteroskedastic noise Kersting et al., 2007, has been used as the surrogate model of BO2.

A.1.1 Choosing a replication strategy and convergence measure

The selection of the replication strategy and the convergence measure is characterized by a discrete, binary choice. Therefore, there is no necessity to fit a Gaussian process on the "best found" and ROC AUC score that result from the ten runs that are executed for each module. Instead, the module that performs the best across the subjects is chosen.

A.2 Results

A.2.1 Trading Exploration for exploitation

The mean "best found" metric scores that calculated over the ten runs for the different β_1 values are presented per dimensionality and subject. That is, Figure A.2 describes the results for the one, two and seven-dimensional optimization problems for subject VPpblz_15_08_14, while Figures A.3 and A.4 do so for respectively subject VPpboa_15_08_11 and VPpbob_15_08_13. The "best found" scores are presented in combination with the posterior distribution of the Most likely heteroskedasic Gaussian process that is fit on the scores. The highest value of the posterior mean of the Gaussian process is highlighted with the vertical green bar. The 95% confidence interval of the Gaussian process is described with the light-blue shaded area. Lastly, the scores that have been obtained by classifying the data with the preprocessing methods that have been provided in Sosulski and Tangermann, 2022, are presented with the horizontal red bar.

The optimal values that have been found for β_1 vary between the evaluated subjects and dimensionalities. Given the different optima, there seems to be a preference towards exploitative strategies (i.e. $\beta_1 < 0.5$). The only exception to the latter observation is formed by the one-dimensional optimization problem for subject VPpboa_15_08_11, which is presented in the left pane of Figure A.3. Yet, the degree of exploitation or exploration is not guaranteed to strongly influence the performance of the optimization algorithm. In particular for subject VPpboa_15_08_11, the function described by β_1 is rather flat, indicating that changes in β_1 do not have a large effect on the optimization process. This phenomenon could be fueled by the fact that the data from this subject is difficult to classify in general, as is indicated with a paper ROC AUC score of 0.51 that close to chance level (0.5). If the input data is not informative, then there is no use changing the classification parameters, which renders the entire optimization process, and therefore the selection of β fruitless.

Another observation that is worth noting is that the size of the 95% confidence interval of the Gaussian process tends to increase when the dimensionality of the optimization problem increases. In contrast, the relation between the values the mean "best found" ROC AUC scores and the dimensionality is less evident. One might expect that higher-dimensional objective functions are harder to optimize. Yet, the optima that are found for the seven-dimensional optimization problem are not always inferior to the optima that are associated with the lower-dimensional optimization problems: counterexamples are provided by the Figures A.3 and A.4.

Given the large variation between subjects and dimensionalities, there is no silver bullet for the selection of β_1. Instead, β_1 is set to 0.187, the average over the posterior optima of all subjects and dimensionalities. As this parameter choice complies with the preference towards exploiting strategies, it is expected to provide the Bayesian optimization with a warmer start than choosing a random value for β_1.

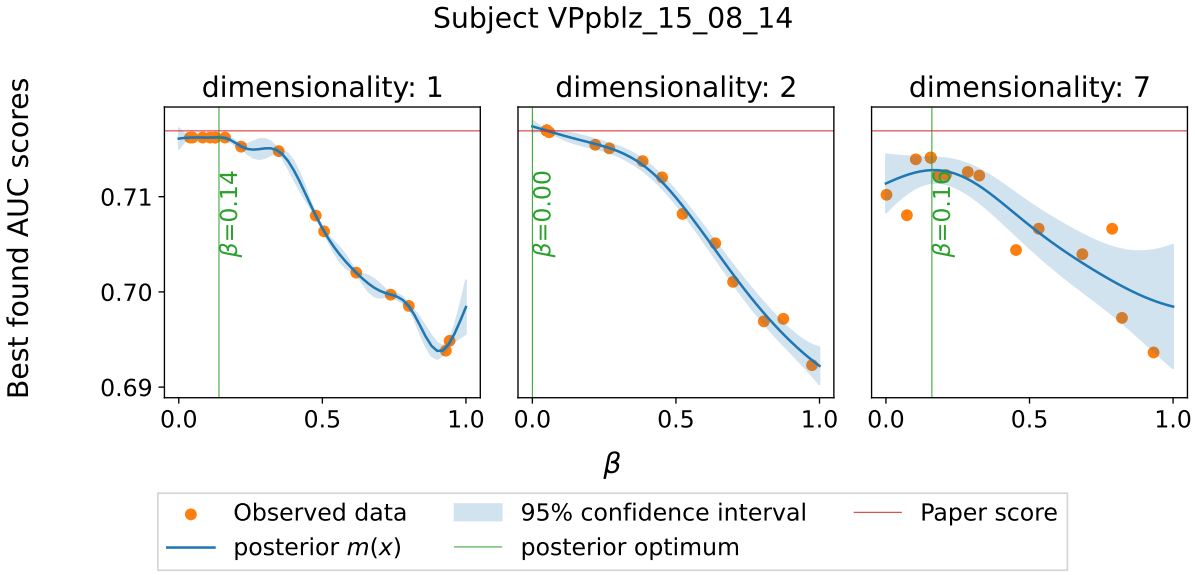


Figure A.2: The "best found" ROC AUC scores for the optimization of β_1, obtained for subject VPpblz_15_08_14. The three different panes describe the observations that have been calculated for the different dimensionalities of the objective function. To model the function of β_1, a Gaussian process has been fitted on the observations. The resulting posterior distribution is shown in blue. The vertical green bar highlights the optimum of the mean of the Gaussian process posterior. The β_1 value that corresponds to this optimum is annotated in the figures. The horizontal red bar indicates the classification ROC AUC that has been obtained with the preprocessing according to Sosulski and Tangermann, 2022.

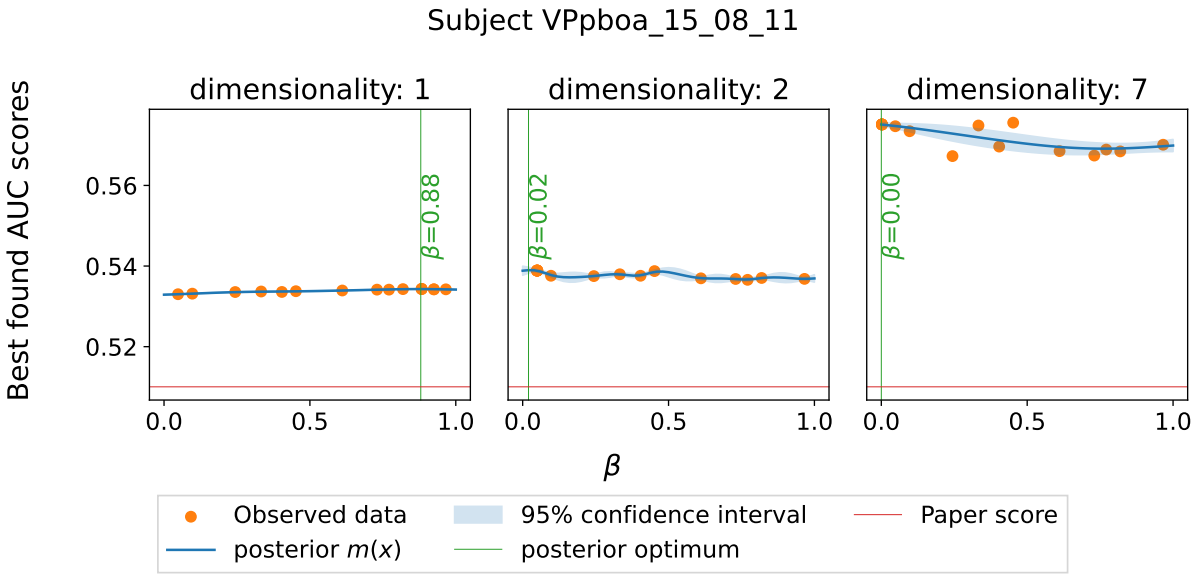


Figure A.3: The "best found" ROC AUC scores for the optimization of β_1, obtained for subject VPpboa_15_08_11. The results are presented as described in Figure A.2.

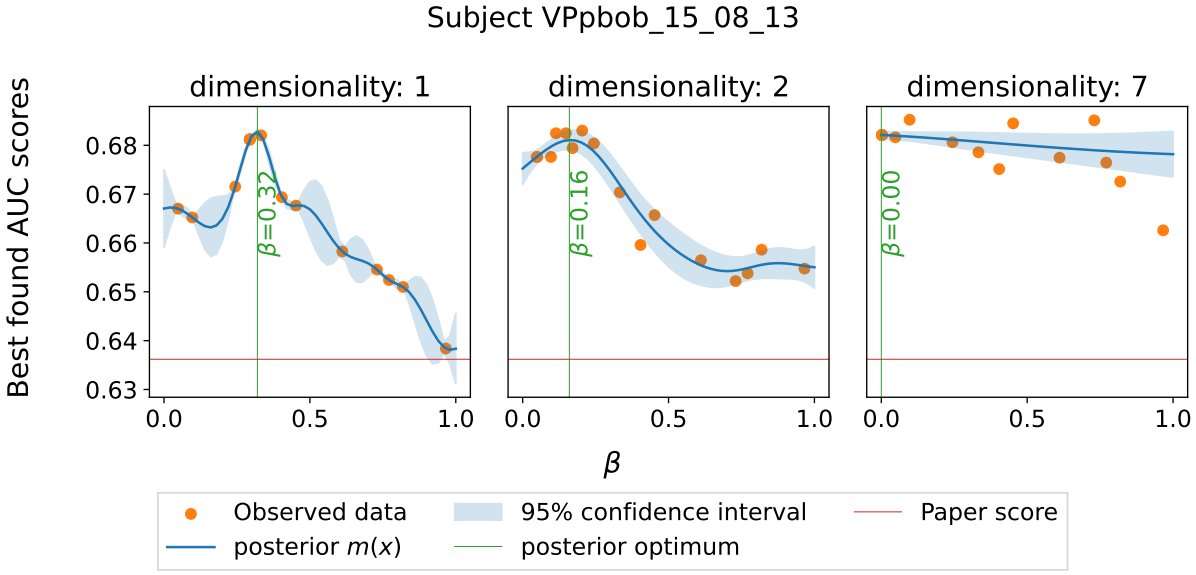


Figure A.4: The "best found" ROC AUC scores for the optimization of β_1, obtained for subject VPpbob_15_08_13. The results are presented as described in Figure A.2.

A.2.2 Choice of replicator

Given the value of $\beta = 0.187$, the performances of the Bayesian optimization architectures that are parameterized with the *sequential* and *variance* replicator are assessed. The results are described in Figures A.5-A.7. The results are again grouped by the subject. Each figure shows the boxplots of the distributions of "best found" ROC AUC scores that have been obtained for the ten runs per participant, replication strategy and dimensionality. To put the results into perspective, the score that has been obtained using the preprocessing in Sosulski and Tangermann, 2022 has been plotted in red.

While looking at the boxplots and paying attention to the median and lower whiskers in particular, it becomes clear that the sequential replication strategy is underperforming when the

one and two-dimensional optimization problems are considered. The performance of the *sequential* replicator is on par with, or better than the other strategies when the seven-dimensional optimization problem is considered. The addition of the *variance* replicator yields a median performance that is comparable to, or better than an optimization algorithm that does not include any replications. However, the variance between the runs tends to be smaller when the *variance* replicator is used, albeit the one-dimensional optimization problem of subject Ppbob_15_08_13 (Figure A.7) forms an exception to this observation.

Table A.1 describes the average number of replications that have been made by each replicator. This table, and the discussion thereof, can be found below the Figures A.5-A.7.

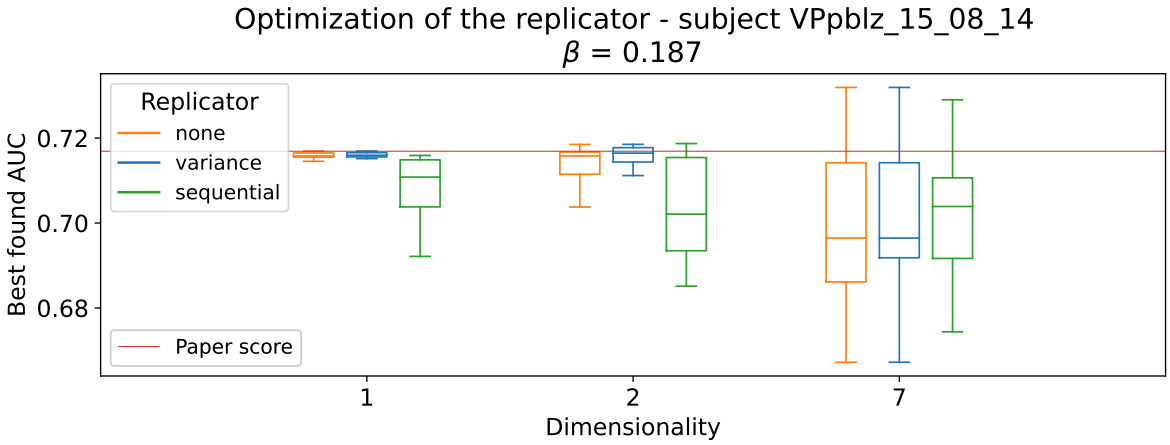


Figure A.5: The "best found" ROC AUC scores that have been obtained with different replication strategies for subject VPpblz_15_08_14.

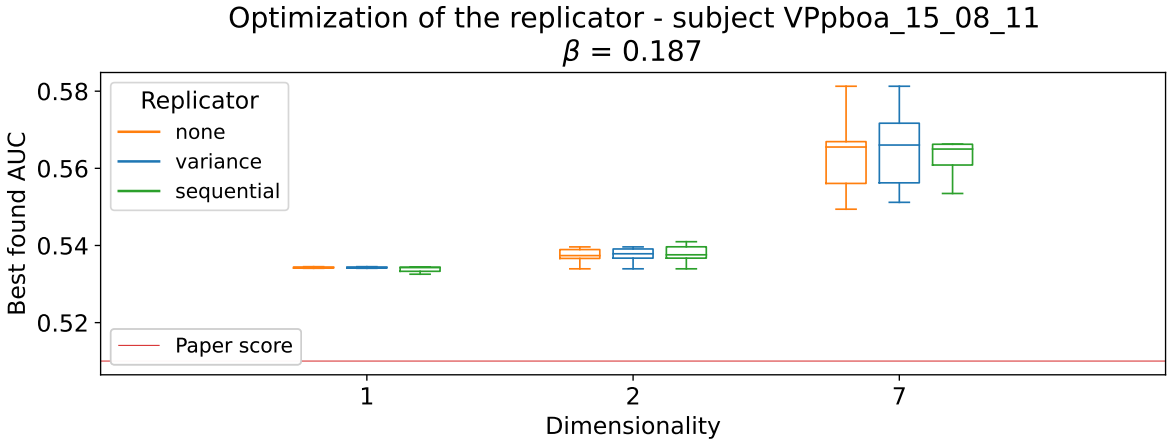


Figure A.6: The "best found" ROC AUC scores that have been obtained with different replication strategies for subject VPpboa_15_08_11.

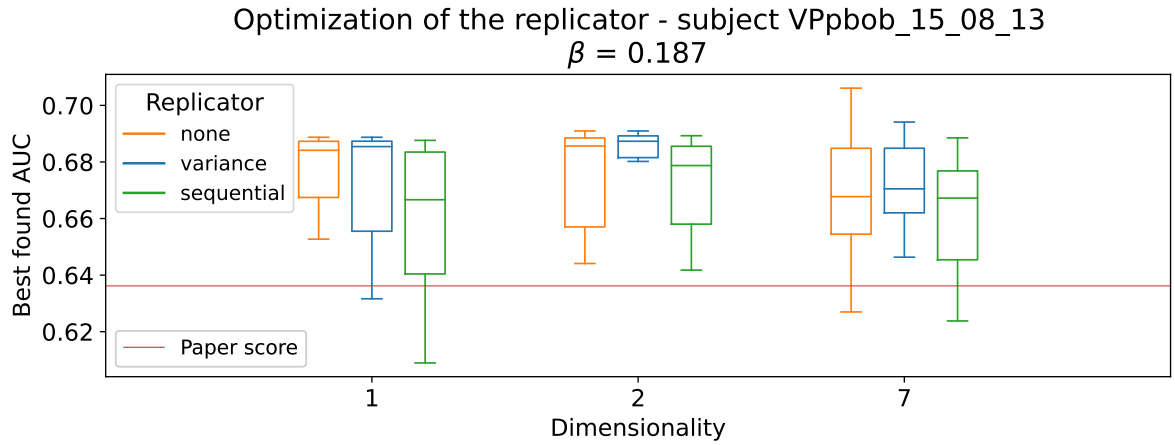


Figure A.7: The "best found" ROC AUC scores that have been obtained with different replication strategies for subject VPpbob_15_08_13.

The average number of replications that have been made by each replication strategy is presented in Table A.1. The observations are grouped per participant (within each subtable) and per dimensionality (within each column). The Bayesian optimization architectures that were not equipped with a replicator were not able to make any replications. Hence, the number of replications for the *None* replicator is equal to 0 across participants and dimensions. From the tables, it seems that the *variance* replicator tends to make fewer replications than the *sequential* replicator. Furthermore, there seems to be a downwards trend in the number of replications that are made by the *variance* replicator if the dimensionality of the optimization problem increases. For the *sequential* replicator however, this effect seems absent.

	Dimensionality		
Replicator	1	2	7
None	0	0	0
Variance	2.8	1.8	0.5
Sequential	20.1	19	20.4

(a) VPpblz_15_08_14

	Dimensionality		
Replicator	1	2	7
None	0	0	0
Variance	3.1	0.8	0.4
Sequential	19.5	21	20.7

(b) VPpboa_15_08_11

	Dimensionality		
Replicator	1	2	7
None	0	0	0
Variance	2.3	1.3	0.5
Sequential	19.9	22.8	19.6

(c) VPpbob_15_08_13

Table A.1: The number of replications that have been made per replicator, averaged over the 10 runs of the optimization algorithm. The three different tables present the results per participant.

The downward trend that has been found for the *variance* replicator could be caused by the relative sparsity of the samples in higher dimensions. Recall from Section 4.3.4 that the *variance* replicator as well as the *sequential* replicator base their decisions on the modelling uncertainty[2], $\check{\sigma}^2(x)$. But, in contrast to the *sequential* replicator, the *variance* replicator also takes the Euclidean distance between the proposed sample and the current, best-performing existing design into account. That is, a replication is only made if the proposed sample is the nearest neighbour of the best-performing existing design.

Nevertheless, this strategy comes with a few snags if a higher-dimensional objective function is considered, as the notion of (Euclidean) distance or proximity is less meaningful in high dimensions Aggarwal et al., 2001. A possible consequence of this phenomenon is that, despite the exploitation-favouring initialization of $\beta = 0.187$, the proposed locations x^*_{acf} are not necessarily the nearest neighbour of the best-performing existing design. Thus, in a high-dimensional

[2]For the *variance* replicator, the notion of the modelling uncertainty is embedded in the posterior variance $\sigma(x)$.

setting, the distance is not an adequate measure to take the verdict of the acquisition function into account. Given that the proposed samples are less likely to be the nearest neighbour of the optimal existing design, relatively fewer replications are made. This hypothesis would explain why the *sequential* replicator does not display this phenomenon: this strategy is not based on a notion of proximity.

Yet, the phenomenon that is described above is not necessarily a bad thing, as reflected in the performance of the three strategies that are presented in the Figures A.5-A.7. A higher-dimensional objective function comes with a larger input space that needs to be probed by the optimization algorithm. If the sampling budget is spent on replications, then these samples cannot be used anymore to explore the large input space. In particular when a lot of samples are replicated, optimization algorithm could miss the optima that would have been found in other cases. This is illustrated in the Figures A.5-A.7, where the upper whiskers of the *sequential* boxplot are generally lower than for the other two strategies for the seven-dimensional optimization problem. Therefore, the inclusion of the proximity could be regarded as a natural constraint on the replication budget in high dimensional spaces. In particular, if the downward trend within the number of replicates can be extrapolated to higher dimensions, then the behaviour of an optimization algorithm with the *variance* replicator is expected to match the behaviour of an algorithm that makes use of no replicator at all.

In light of the performance of the three replicators, the *variance* replicator is selected as the optimal replicator. The performance of an algorithm that makes replications is presumed to be better than the performance of an algorithm that does not make any replications at all. This observation is in line with hypothesis H_4:.

A.2.3 Convergence analysis

Now that $\beta = 0.187$ and the *variance* replicator have been chosen, the two different convergence measures, *MSE* and *length scale* are evaluated in the same fashion as the replicators. The boxplots that visualise the distribution of the "best found" AUC scores of the 10 runs are shown in the Figures A.8-A.10. When inspecting the results, it seems that the addition of a convergence measure to the selector often does not result in an improvement of the optimization process. In fact, the performance of algorithms that have been initialized with a convergence measure is comparable to, or worse than, the performance of the algorithm that does not use a convergence measure. Again, the only exception is formed by the one-dimensional case of participant VPpbob_15_08_13, where the *MSE* convergence measure slightly outperforms the other strategies. Therefore, it is decided against adopting a convergence measure in the final architecture.

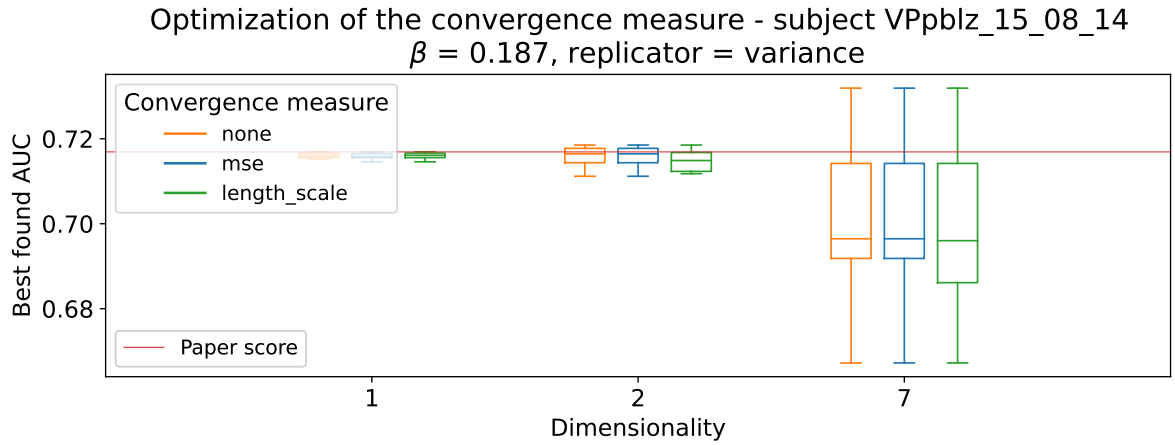


Figure A.8: The "best found" ROC AUC scores that have been obtained with different convergence measures for subject VPpblz_15_08_14.

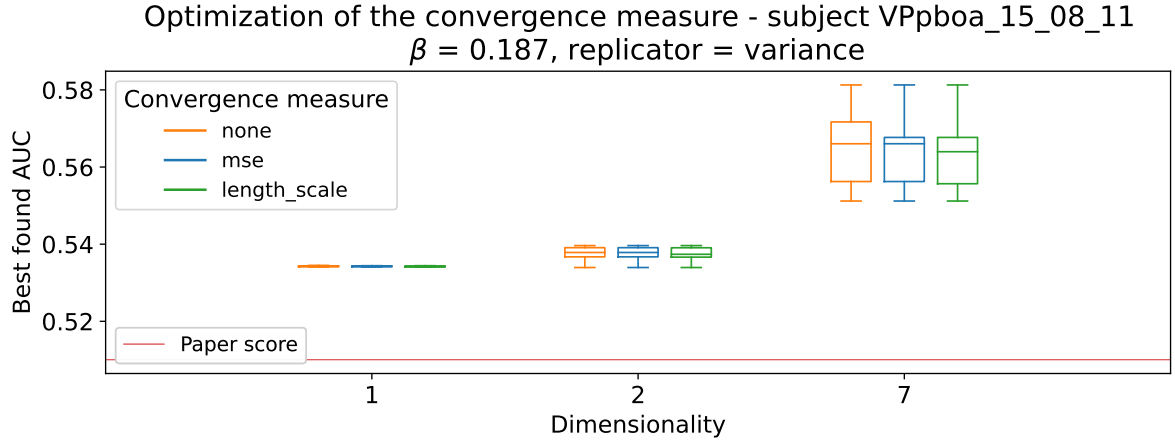


Figure A.9: The "best found" ROC AUC scores that have been obtained with different convergence measures for subject VPpboa_15_08_11.

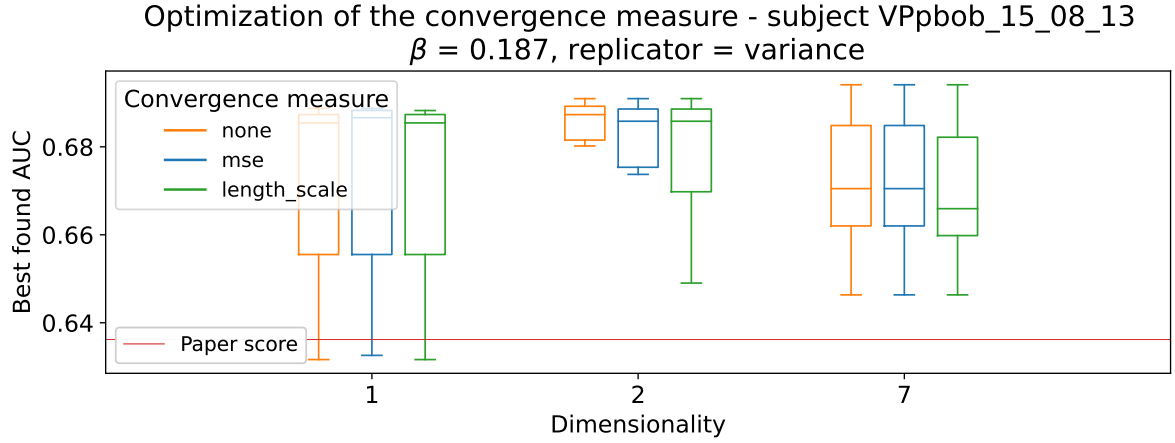


Figure A.10: The "best found" ROC AUC scores that have been obtained with different convergence measures for subject VPpbob_15_08_13.

Appendix B

Optimization traces of the best found ROC AUC scores

B.1 Homoskedastic noise

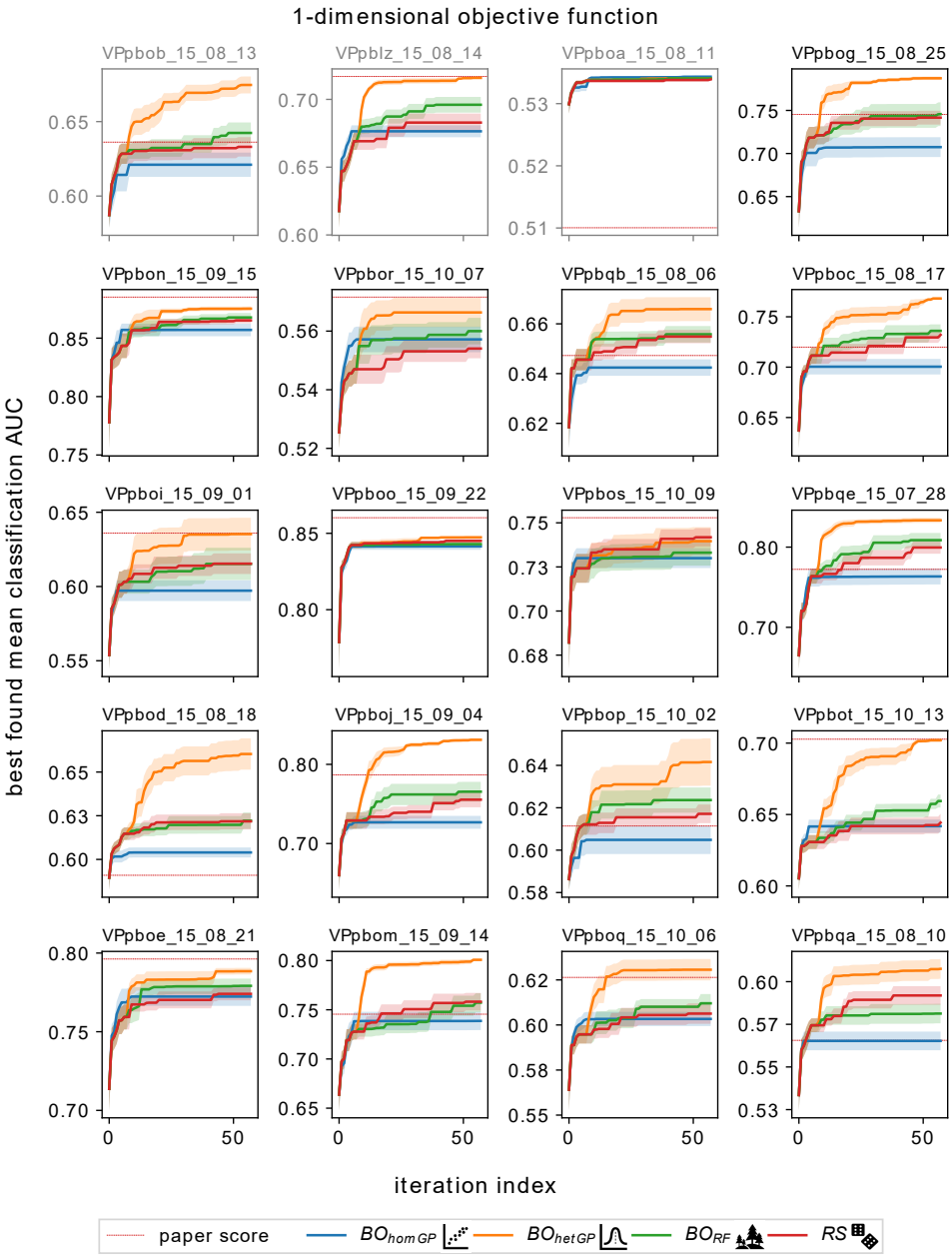


Figure B.1: The highest ROC AUC score that has been found at each iteration. The curves represent the ROC AUC scores that have been obtained by optimizing a one-dimensional objective function.

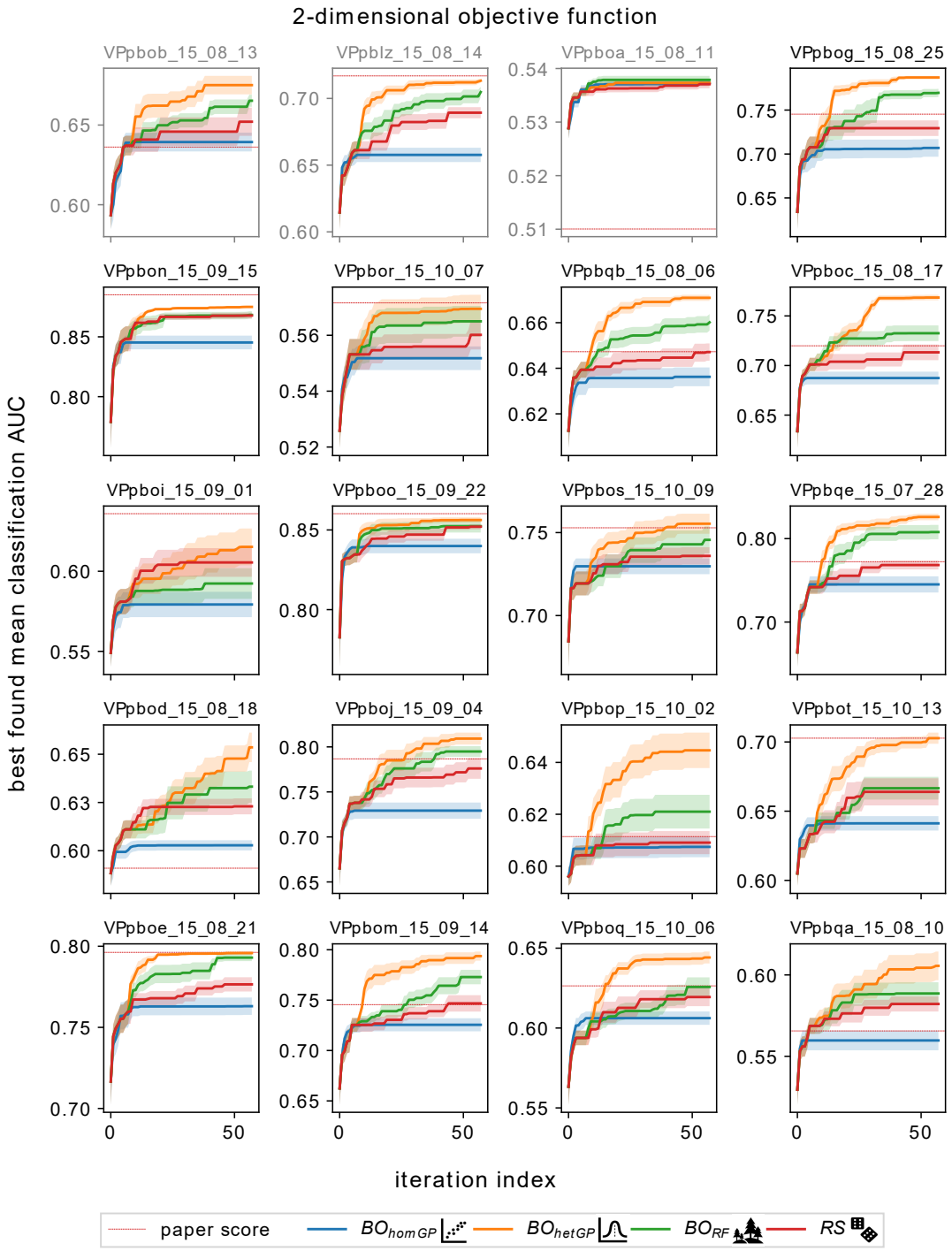


Figure B.2: The highest ROC AUC score that has been found at each iteration. The curves represent the ROC AUC scores that have been obtained by optimizing a two-dimensional objective function.

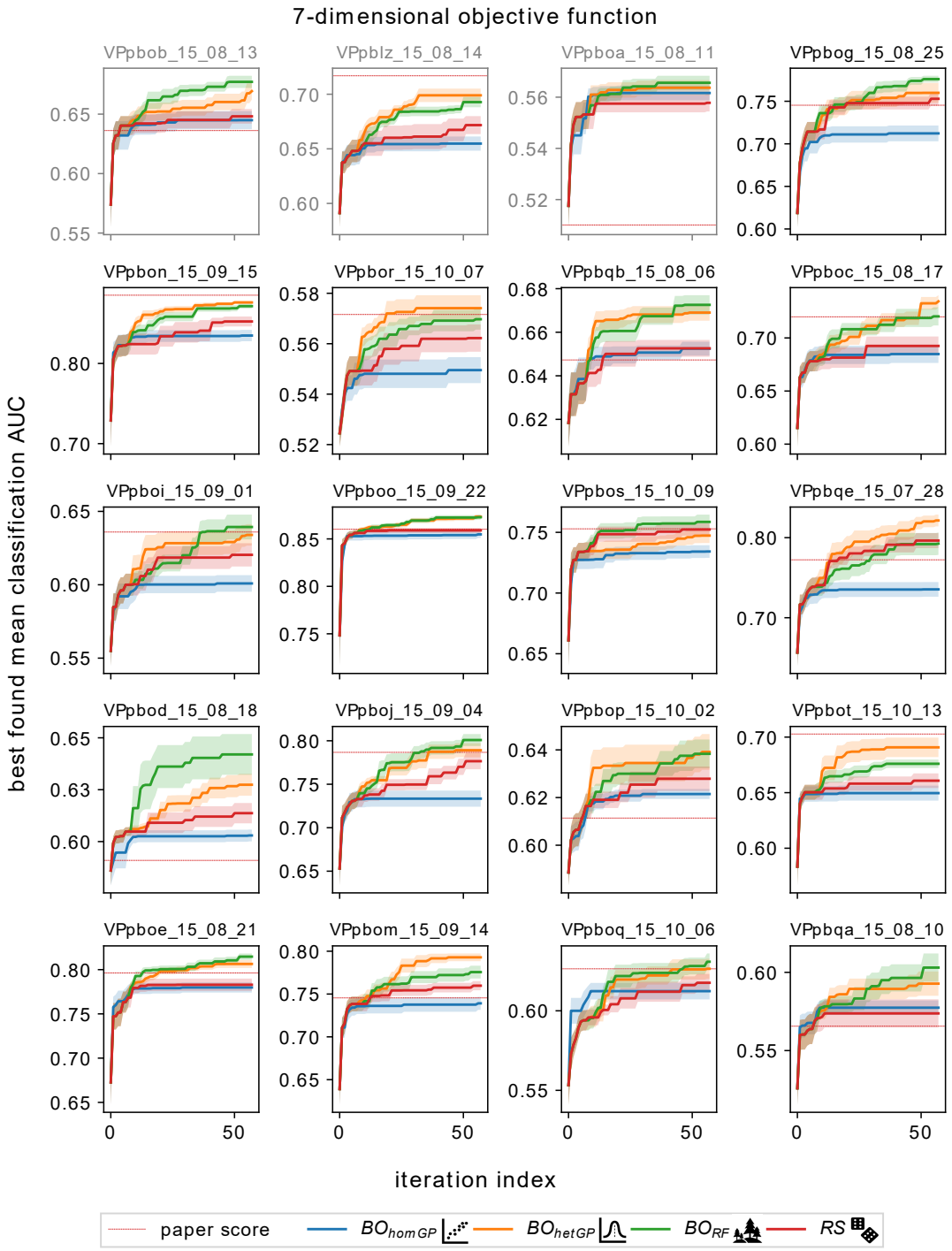


Figure B.3: The highest ROC AUC score that has been found at each iteration. The curves represent the ROC AUC scores that have been obtained by optimizing a seven-dimensional objective function.

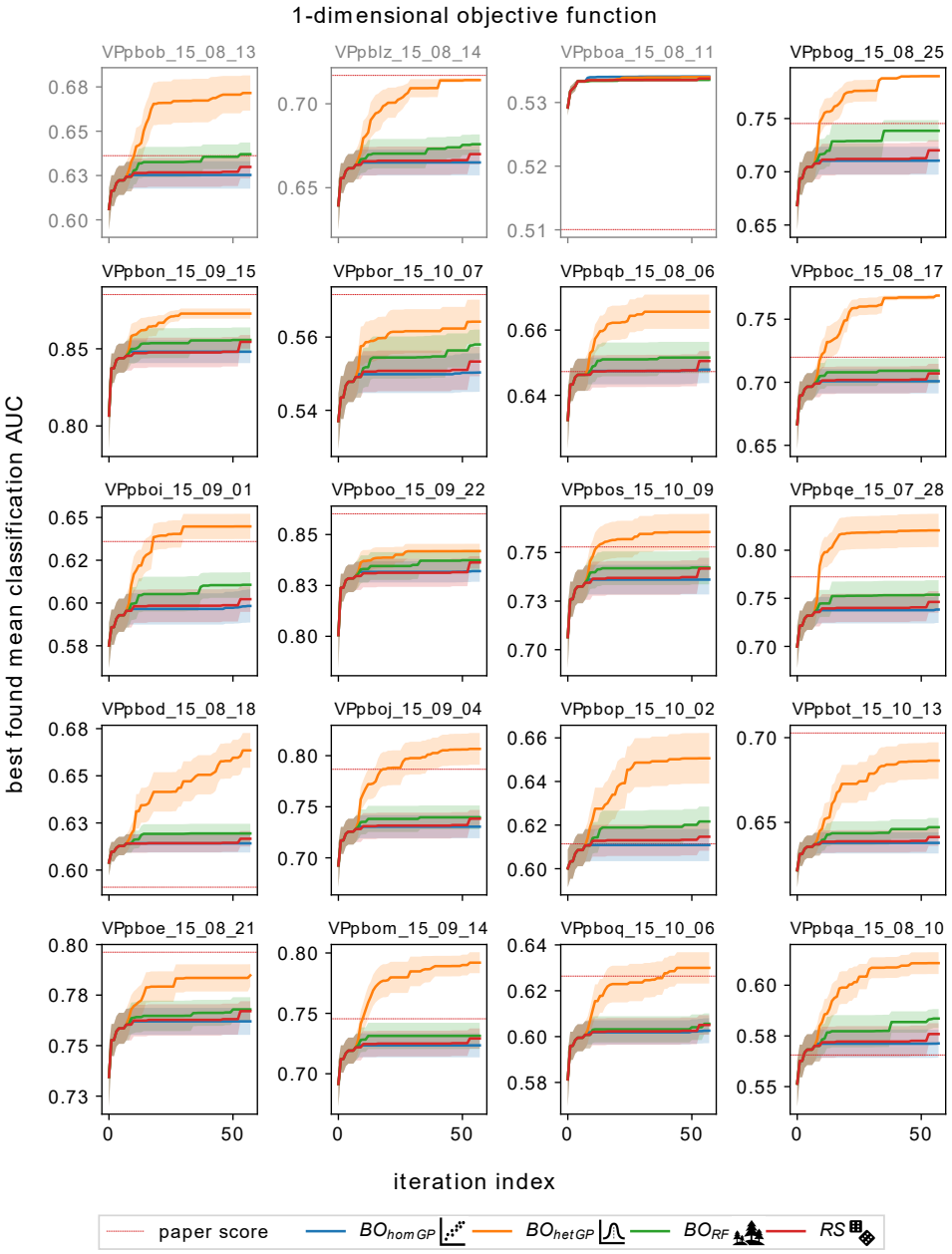


Figure B.4: The highest ROC AUC score that has been found at each iteration. The curves represent the ROC AUC scores that have been obtained by optimizing a one-dimensional objective function.

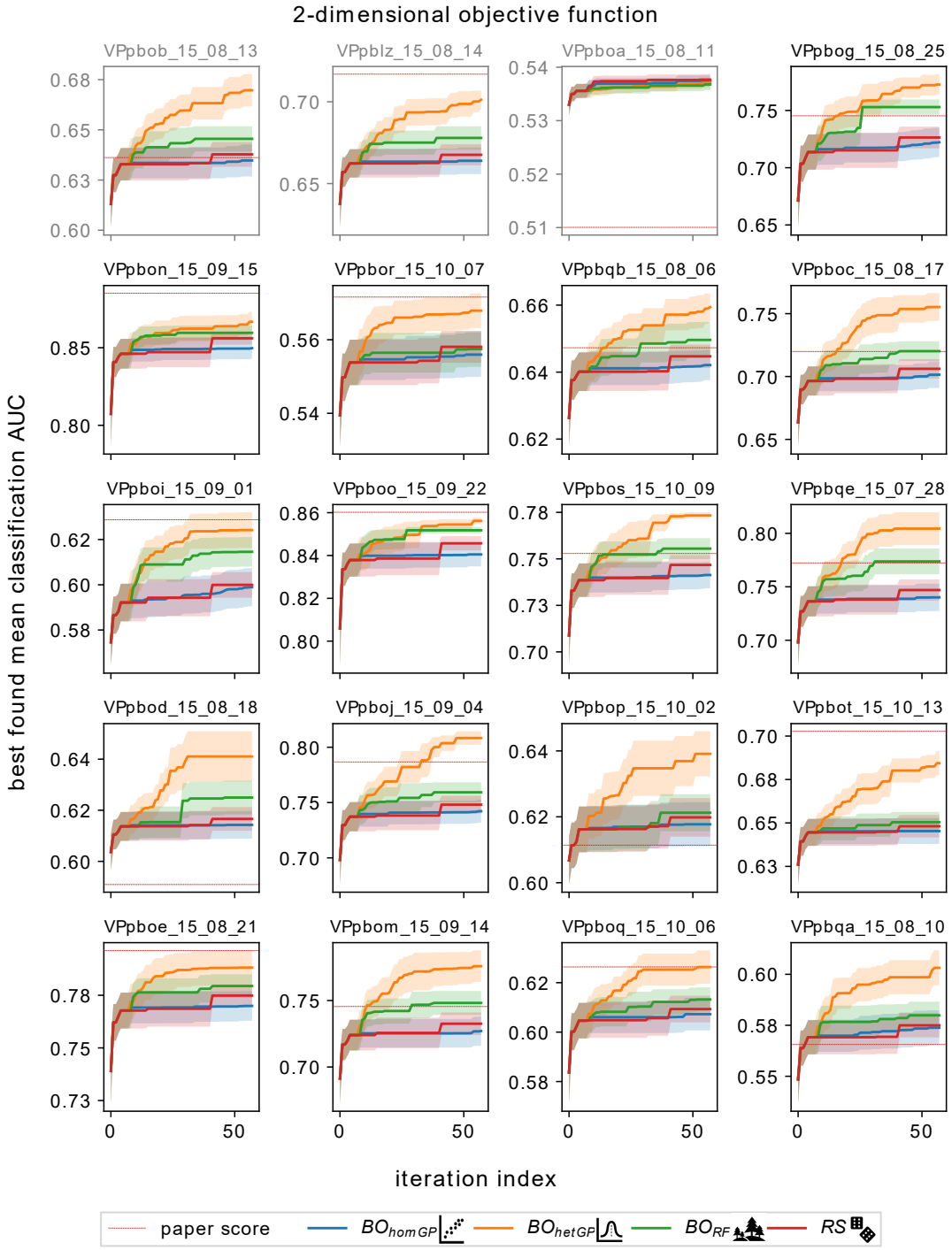


Figure B.5: The highest ROC AUC score that has been found at each iteration. The curves represent the ROC AUC scores that have been obtained by optimizing a two-dimensional objective function.

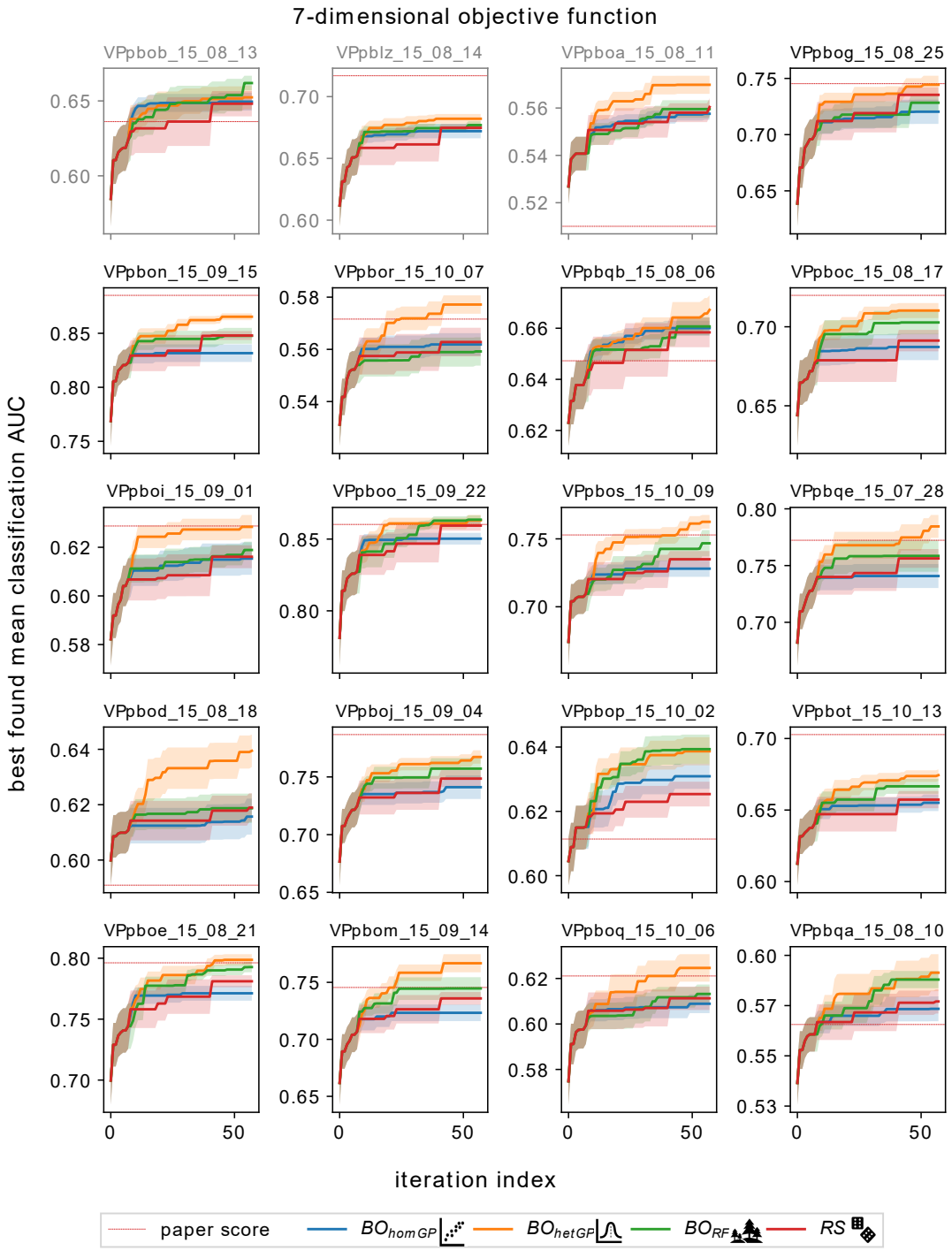


Figure B.6: The highest ROC AUC score that has been found at each iteration. The curves represent the ROC AUC scores that have been obtained by optimizing a seven-dimensional objective function.

Appendix C

Best found ROC AUC scores

C.1 Homoskedastic noise

The mean ROC AUC scores of the four optimization algorithms obtained from the one-dimensional objective function

Participant	BO_{homGP}		BO_{hetGP}		BO_{RF}		RS	
	μ	SEM	μ	SEM	μ	SEM	μ	SEM
VPpbob_15_08_13	0.6210	0.0026	**0.6746**	0.0018	0.6424	0.0022	0.6330	0.0021
VPpblz_15_08_14	0.6765	0.0014	**0.7159**	0.0001	0.6960	0.0018	0.6829	0.0021
VPpboa_15_08_11	**0.5343**	0.0000	0.5341	0.0001	0.5341	0.0001	0.5339	0.0001
VPpbog_15_08_25	0.7074	0.0036	**0.7873**	0.0007	0.7457	0.0044	0.7416	0.0025
VPpbon_15_09_15	0.8570	0.0017	**0.8752**	0.0006	0.8677	0.0012	0.8652	0.0008
VPpbor_15_10_07	0.5572	0.0014	**0.5664**	0.0016	0.5600	0.0014	0.5540	0.0010
VPpbqb_15_08_06	0.6424	0.0010	**0.6659**	0.0015	0.6558	0.0010	0.6549	0.0008
VPpboc_15_08_17	0.7005	0.0024	**0.7681**	0.0004	0.7360	0.0020	0.7321	0.0016
VPpboi_15_09_01	0.5972	0.0022	**0.6353**	0.0035	0.6152	0.0034	0.6154	0.0022
VPpboo_15_09_22	0.8415	0.0007	**0.8474**	0.0001	0.8429	0.0007	0.8451	0.0004
VPpbos_15_10_09	0.7300	0.0018	0.7396	0.0025	0.7331	0.0023	**0.7418**	0.0016
VPpbqe_15_07_28	0.7634	0.0032	**0.8334**	0.0010	0.8087	0.0028	0.7995	0.0025
VPpbod_15_08_18	0.6040	0.0009	**0.6603**	0.0028	0.6222	0.0015	0.6218	0.0015
VPpboj_15_09_04	0.7268	0.0026	**0.8310**	0.0006	0.7655	0.0039	0.7554	0.0030
VPpbop_15_10_02	0.6049	0.0021	**0.6416**	0.0035	0.6236	0.0019	0.6171	0.0014
VPpbot_15_10_13	0.6419	0.0015	**0.7020**	0.0004	0.6594	0.0014	0.6443	0.0020
VPpboe_15_08_21	0.7723	0.0019	**0.7885**	0.0007	0.7792	0.0015	0.7741	0.0013
VPpbom_15_09_14	0.7387	0.0029	**0.8008**	0.0003	0.7572	0.0030	0.7582	0.0028
VPpboq_15_10_06	0.6033	0.0013	**0.6307**	0.0019	0.6120	0.0016	0.6063	0.0018
VPpbqa_15_08_10	0.5653	0.0017	**0.6075**	0.0018	0.5813	0.0018	0.5919	0.0017
All	0.6743	0.0045	**0.7153**	0.0048	0.6919	0.0046	0.6884	0.0046

Table C.1: The mean "best found" ROC AUC scores that have been found per participant. The scores are averaged over the ten runs. The scores have been obtained from the one-dimensional objective function.

The mean ROC AUC scores of the four optimization algorithms obtained from the two-dimensional objective function

Participant	BO_{homGP}		BO_{hetGP}		BO_{RF}		RS	
	μ	SEM	μ	SEM	μ	SEM	μ	SEM
VPpbob_15_08_13	0.6393	0.0019	**0.6751**	0.0018	0.6653	0.0012	0.6521	0.0028
VPpblz_15_08_14	0.6577	0.0017	**0.7133**	0.0005	0.7048	0.0012	0.6893	0.0013
VPpboa_15_08_11	0.5373	0.0002	0.5373	0.0002	**0.5379**	0.0002	0.5371	0.0002
VPpbog_15_08_25	0.7069	0.0032	**0.7871**	0.0006	0.7696	0.0015	0.7294	0.0028
VPpbon_15_09_15	0.8453	0.0018	**0.8750**	0.0002	0.8683	0.0010	0.8678	0.0008
VPpbor_15_10_07	0.5518	0.0013	**0.5695**	0.0016	0.5650	0.0017	0.5602	0.0017
VPpbqb_15_08_06	0.6362	0.0013	**0.6709**	0.0004	0.6601	0.0012	0.6471	0.0012
VPpboc_15_08_17	0.6874	0.0020	**0.7685**	0.0009	0.7325	0.0025	0.7133	0.0025
VPpboi_15_09_01	0.5793	0.0025	**0.6152**	0.0036	0.5923	0.0030	0.6055	0.0028
VPpboo_15_09_22	0.8399	0.0014	**0.8562**	0.0006	0.8524	0.0013	0.8520	0.0005
VPpbos_15_10_09	0.7297	0.0015	**0.7553**	0.0019	0.7456	0.0026	0.7359	0.0017
VPpbqe_15_07_28	0.7451	0.0031	**0.8260**	0.0012	0.8078	0.0027	0.7683	0.0017
VPpbod_15_08_18	0.6028	0.0008	**0.6536**	0.0024	0.6332	0.0026	0.6230	0.0013
VPpboj_15_09_04	0.7292	0.0028	**0.8092**	0.0021	0.7949	0.0019	0.7760	0.0036
VPpbop_15_10_02	0.6074	0.0013	**0.6447**	0.0022	0.6210	0.0021	0.6091	0.0014
VPpbot_15_10_13	0.6412	0.0016	**0.7027**	0.0013	0.6666	0.0026	0.6640	0.0031
VPpboe_15_08_21	0.7630	0.0017	**0.7958**	0.0003	0.7930	0.0007	0.7765	0.0014
VPpbom_15_09_14	0.7255	0.0021	**0.7937**	0.0014	0.7729	0.0023	0.7468	0.0026
VPpboq_15_10_06	0.6062	0.0013	**0.6442**	0.0012	0.6257	0.0019	0.6194	0.0018
VPpbqa_15_08_10	0.5597	0.0019	**0.6056**	0.0028	0.5886	0.0022	0.5822	0.0015
All	0.6695	0.0044	**0.7149**	0.0048	0.6999	0.0048	0.6877	0.0045

Table C.2: The mean "best found" ROC AUC scores that have been found per participant. The scores are averaged over the ten runs. The scores have been obtained from the two-dimensional objective function.

The mean ROC AUC scores of the four optimization algorithms obtained from the seven-dimensional objective function

Participant	BO_{homGP}		BO_{hetGP}		BO_{RF}		RS	
	μ	SEM	μ	SEM	μ	SEM	μ	SEM
VPpbob_15_08_13	0.6449	0.0024	0.6693	0.0023	**0.6771**	0.0016	0.6482	0.0019
VPpblz_15_08_14	0.6548	0.0021	**0.6990**	0.0020	0.6928	0.0015	0.6717	0.0026
VPpboa_15_08_11	0.5616	0.0009	0.5637	0.0009	**0.5656**	0.0009	0.5578	0.0011
VPpbog_15_08_25	0.7123	0.0029	0.7599	0.0037	**0.7760**	0.0012	0.7529	0.0026
VPpbon_15_09_15	0.8348	0.0023	**0.8761**	0.0006	0.8715	0.0009	0.8524	0.0020
VPpbor_15_10_07	0.5495	0.0016	**0.5741**	0.0016	0.5697	0.0013	0.5622	0.0017
VPpbqb_15_08_06	0.6525	0.0010	0.6690	0.0012	**0.6726**	0.0014	0.6526	0.0012
VPpboc_15_08_17	0.6848	0.0026	**0.7345**	0.0018	0.7203	0.0027	0.6925	0.0028
VPpboi_15_09_01	0.6009	0.0018	0.6340	0.0023	**0.6394**	0.0027	0.6204	0.0025
VPpboo_15_09_22	0.8548	0.0012	**0.8734**	0.0007	0.8729	0.0008	0.8590	0.0009
VPpbos_15_10_09	0.7342	0.0017	0.7473	0.0019	**0.7586**	0.0019	0.7523	0.0016
VPpbqe_15_07_28	0.7354	0.0029	**0.8214**	0.0023	0.7925	0.0044	0.7964	0.0030
VPpbod_15_08_18	0.6029	0.0009	0.6274	0.0017	**0.6420**	0.0031	0.6137	0.0016
VPpboj_15_09_04	0.7334	0.0029	0.7891	0.0025	**0.8009**	0.0022	0.7765	0.0029
VPpbop_15_10_02	0.6215	0.0007	**0.6391**	0.0024	0.6383	0.0019	0.6279	0.0016
VPpbot_15_10_13	0.6496	0.0021	**0.6908**	0.0028	0.6760	0.0012	0.6608	0.0021
VPpboe_15_08_21	0.7798	0.0018	0.8064	0.0014	**0.8147**	0.0011	0.7829	0.0022
VPpbom_15_09_14	0.7392	0.0024	**0.7929**	0.0014	0.7756	0.0024	0.7598	0.0017
VPpboq_15_10_06	0.6123	0.0016	0.6264	0.0020	**0.6308**	0.0016	0.6176	0.0018
VPpbqa_15_08_10	0.5774	0.0015	0.5928	0.0025	**0.6030**	0.0029	0.5738	0.0029
All	0.6768	0.0042	0.7093	0.0047	**0.7095**	0.0046	0.6916	0.0046

Table C.3: The mean "best found" ROC AUC scores that have been found per participant. The scores are averaged over the ten runs. The scores have been obtained from the seven-dimensional objective function.

C.2 Heteroskedastic noise

The mean ROC AUC scores of the four optimization algorithms obtained from the one-dimensional objective function

Participant	BO_{homGP}		BO_{hetGP}		BO_{RF}		RS	
	μ	SEM	μ	SEM	μ	SEM	μ	SEM
VPpbob_15_08_13	0.6253	0.0024	**0.6716**	0.0031	0.6370	0.0021	0.6298	0.0020
VPpblz_15_08_14	0.6651	0.0024	**0.7141**	0.0004	0.6759	0.0018	0.6700	0.0019
VPpboa_15_08_11	**0.5341**	0.0001	0.5340	0.0001	0.5336	0.0001	0.5338	0.0001
VPpbog_15_08_25	0.7105	0.0041	**0.7898**	0.0002	0.7385	0.0032	0.7201	0.0030
VPpbon_15_09_15	0.8481	0.0023	**0.8728**	0.0010	0.8557	0.0026	0.8545	0.0014
VPpbor_15_10_07	0.5503	0.0017	**0.5642**	0.0019	0.5580	0.0013	0.5533	0.0013
VPpbqb_15_08_06	0.6479	0.0013	**0.6656**	0.0017	0.6516	0.0015	0.6505	0.0010
VPpboc_15_08_17	0.7008	0.0031	**0.7688**	0.0002	0.7091	0.0032	0.7070	0.0024
VPpboi_15_09_01	0.5983	0.0031	**0.6448**	0.0023	0.6107	0.0023	0.6022	0.0024
VPpboo_15_09_22	0.8320	0.0017	**0.8418**	0.0011	0.8374	0.0016	0.8363	0.0010
VPpbos_15_10_09	0.7361	0.0024	**0.7606**	0.0030	0.7422	0.0027	0.7418	0.0018
VPpbqe_15_07_28	0.7383	0.0044	**0.8206**	0.0055	0.7537	0.0048	0.7463	0.0035
VPpbod_15_08_18	0.6142	0.0015	**0.6635**	0.0029	0.6194	0.0016	0.6166	0.0013
VPpboj_15_09_04	0.7305	0.0036	**0.8066**	0.0049	0.7396	0.0037	0.7383	0.0027
VPpbop_15_10_02	0.6108	0.0023	**0.6506**	0.0037	0.6217	0.0022	0.6147	0.0020
VPpbot_15_10_13	0.6379	0.0020	**0.6864**	0.0034	0.6472	0.0017	0.6414	0.0017
VPpboe_15_08_21	0.7620	0.0021	**0.7848**	0.0018	0.7679	0.0019	0.7671	0.0016
VPpbom_15_09_14	0.7235	0.0031	**0.7920**	0.0027	0.7314	0.0034	0.7292	0.0026
VPpboq_15_10_06	0.6026	0.0017	**0.6300**	0.0022	0.6056	0.0014	0.6051	0.0014
VPpbqa_15_08_10	0.5713	0.0022	**0.6109**	0.0017	0.5836	0.0015	0.5759	0.0017
All	0.6720	0.0043	**0.7137**	0.0046	0.6810	0.0043	0.6767	0.0044

Table C.4: The mean "best found" ROC AUC scores that have been found per participant. The scores are averaged over the ten runs. The scores have been obtained from the one-dimensional objective function.

The mean ROC AUC scores of the four optimization algorithms obtained from the two-dimensional objective function

Participant	BO_{homGP}		BO_{hetGP}		BO_{RF}		RS	
	μ	SEM	μ	SEM	μ	SEM	μ	SEM
VPpbob_15_08_13	0.6347	0.0025	**0.6697**	0.0025	0.6455	0.0020	0.6378	0.0019
VPpblz_15_08_14	0.6639	0.0026	**0.7013**	0.0018	0.6779	0.0023	0.6676	0.0020
VPpboa_15_08_11	0.5374	0.0003	0.5371	0.0003	0.5368	0.0003	**0.5377**	0.0003
VPpbog_15_08_25	0.7222	0.0040	**0.7726**	0.0029	0.7529	0.0021	0.7264	0.0030
VPpbon_15_09_15	0.8497	0.0021	**0.8667**	0.0020	0.8596	0.0015	0.8561	0.0013
VPpbor_15_10_07	0.5559	0.0019	**0.5679**	0.0015	0.5575	0.0015	0.5581	0.0014
VPpbqb_15_08_06	0.6421	0.0014	**0.6594**	0.0013	0.6496	0.0017	0.6447	0.0011
VPpboc_15_08_17	0.7015	0.0033	**0.7553**	0.0035	0.7201	0.0025	0.7061	0.0024
VPpboi_15_09_01	0.5987	0.0033	**0.6302**	0.0031	0.6182	0.0026	0.5999	0.0022
VPpboo_15_09_22	0.8405	0.0018	**0.8562**	0.0005	0.8518	0.0005	0.8457	0.0010
VPpbos_15_10_09	0.7413	0.0023	**0.7733**	0.0005	0.7555	0.0018	0.7467	0.0017
VPpbqe_15_07_28	0.7401	0.0040	**0.8044**	0.0048	0.7737	0.0038	0.7470	0.0031
VPpbod_15_08_18	0.6143	0.0017	**0.6411**	0.0031	0.6250	0.0021	0.6166	0.0015
VPpboj_15_09_04	0.7422	0.0033	**0.8083**	0.0019	0.7592	0.0029	0.7481	0.0025
VPpbop_15_10_02	0.6177	0.0021	**0.6391**	0.0022	0.6212	0.0018	0.6198	0.0018
VPpbot_15_10_13	0.6453	0.0023	**0.6844**	0.0022	0.6504	0.0019	0.6481	0.0020
VPpboe_15_08_21	0.7699	0.0022	**0.7881**	0.0026	0.7794	0.0018	0.7749	0.0017
VPpbom_15_09_14	0.7271	0.0035	**0.7758**	0.0038	0.7482	0.0028	0.7325	0.0027
VPpboq_15_10_06	0.6073	0.0021	**0.6263**	0.0021	0.6132	0.0016	0.6093	0.0017
VPpbqa_15_08_10	0.5738	0.0027	**0.6032**	0.0027	0.5799	0.0021	0.5750	0.0019
All	0.6763	0.0043	**0.7080**	0.0047	0.6888	0.0046	0.6799	0.0044

Table C.5: The mean "best found" ROC AUC scores that have been found per participant. The scores are averaged over the ten runs. The scores have been obtained from the two-dimensional objective function.

The mean ROC AUC scores of the four optimization algorithms obtained from the seven-dimensional objective function

Participant	BO_{homGP}		BO_{hetGP}		BO_{RF}		RS	
	μ	SEM	μ	SEM	μ	SEM	μ	SEM
VPpbob_15_08_13	0.6495	0.0019	0.6524	0.0025	**0.6619**	0.0015	0.6481	0.0027
VPpblz_15_08_14	0.6721	0.0018	**0.6821**	0.0012	0.6770	0.0010	0.6748	0.0010
VPpboa_15_08_11	0.5576	0.0008	**0.5698**	0.0012	0.5596	0.0012	0.5604	0.0012
VPpbog_15_08_25	0.7204	0.0034	**0.7444**	0.0025	0.7284	0.0024	0.7354	0.0021
VPpbon_15_09_15	0.8316	0.0026	**0.8653**	0.0010	0.8477	0.0023	0.8480	0.0013
VPpbor_15_10_07	0.5619	0.0014	**0.5772**	0.0011	0.5592	0.0017	0.5628	0.0017
VPpbqb_15_08_06	0.6600	0.0009	**0.6672**	0.0018	0.6607	0.0011	0.6584	0.0018
VPpboc_15_08_17	0.6872	0.0026	**0.7103**	0.0015	0.7027	0.0024	0.6912	0.0021
VPpboi_15_09_01	0.6192	0.0026	**0.6355**	0.0019	0.6236	0.0014	0.6201	0.0019
VPpboo_15_09_22	0.8503	0.0013	0.8631	0.0012	**0.8635**	0.0010	0.8594	0.0011
VPpbos_15_10_09	0.7281	0.0019	**0.7625**	0.0016	0.7468	0.0031	0.7350	0.0019
VPpbqe_15_07_28	0.7408	0.0033	**0.7845**	0.0032	0.7585	0.0044	0.7563	0.0026
VPpbod_15_08_18	0.6157	0.0021	**0.6395**	0.0019	0.6187	0.0017	0.6190	0.0016
VPpboj_15_09_04	0.7411	0.0032	**0.7673**	0.0018	0.7572	0.0029	0.7486	0.0023
VPpbop_15_10_02	0.6309	0.0012	0.6387	0.0014	**0.6393**	0.0014	0.6254	0.0012
VPpbot_15_10_13	0.6551	0.0018	**0.6744**	0.0013	0.6666	0.0017	0.6573	0.0019
VPpboe_15_08_21	0.7712	0.0019	**0.7987**	0.0015	0.7927	0.0013	0.7811	0.0016
VPpbom_15_09_14	0.7233	0.0023	**0.7667**	0.0025	0.7446	0.0031	0.7359	0.0018
VPpboq_15_10_06	0.6111	0.0016	**0.6308**	0.0024	0.6165	0.0016	0.6141	0.0020
VPpbqa_15_08_10	0.5734	0.0020	**0.5914**	0.0029	0.5880	0.0014	0.5771	0.0019
All	0.6800	0.0040	**0.7011**	0.0043	0.6907	0.0043	0.6854	0.0043

Table C.6: The mean "best found" ROC AUC scores that have been found per participant. The scores are averaged over the ten runs. The scores have been obtained from the seven-dimensional objective function.

C.3 Histograms of the averaged Best found ROC AUC scores

C.3.1 Homoskedastic noise

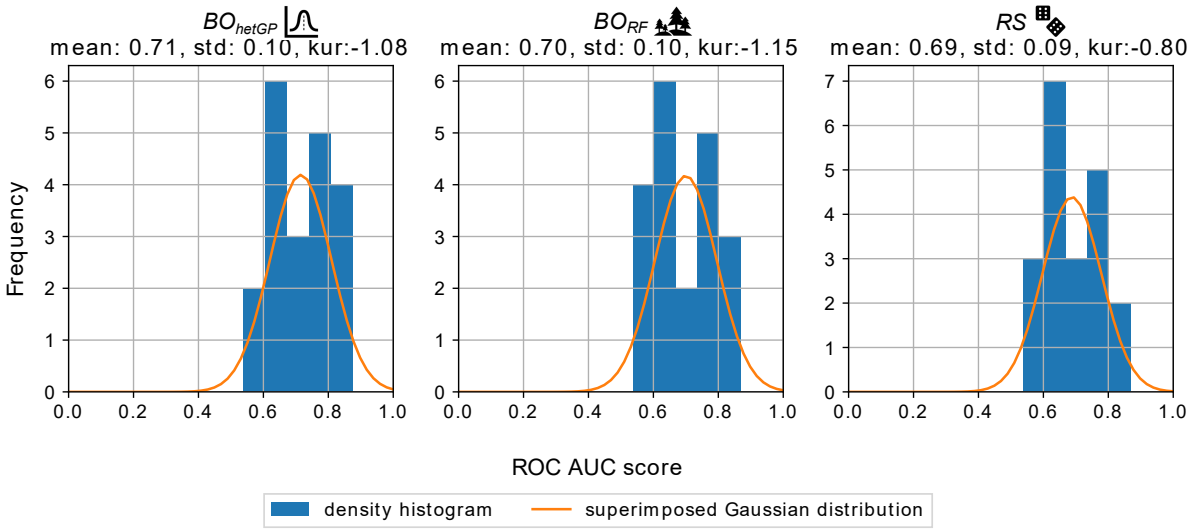


Figure C.1: The distribution of the mean scores per participant, superimposed by a Gaussian distribution that has been fitted upon the data. Obtained from the two-dimensional objective function.

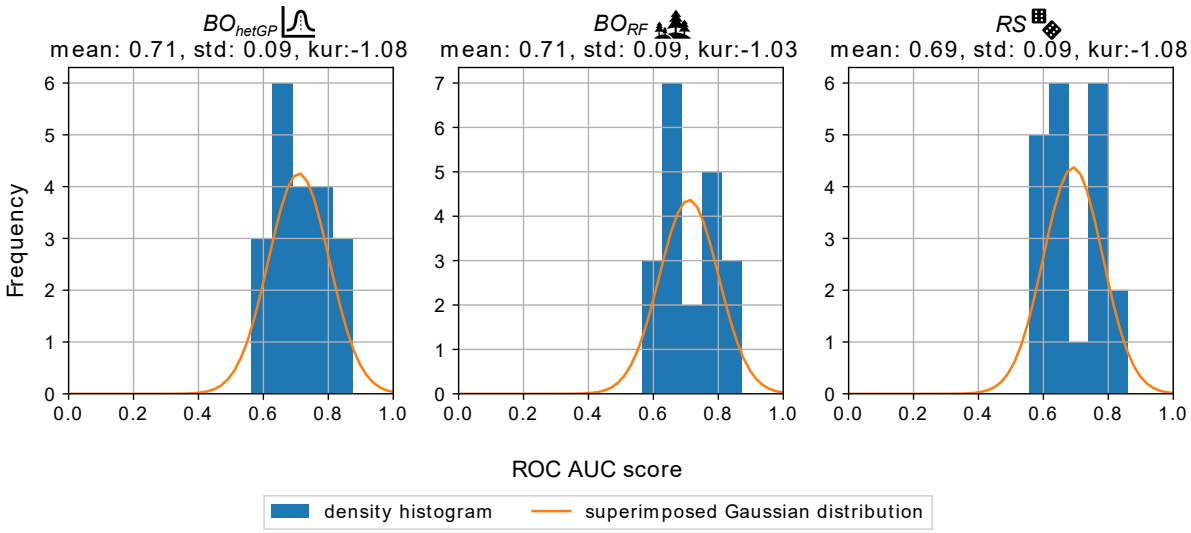


Figure C.2: The distribution of the mean scores per participant, superimposed by a Gaussian distribution that has been fitted upon the data. Obtained from the seven-dimensional objective function.

C.3.2 Heteroskedastic noise

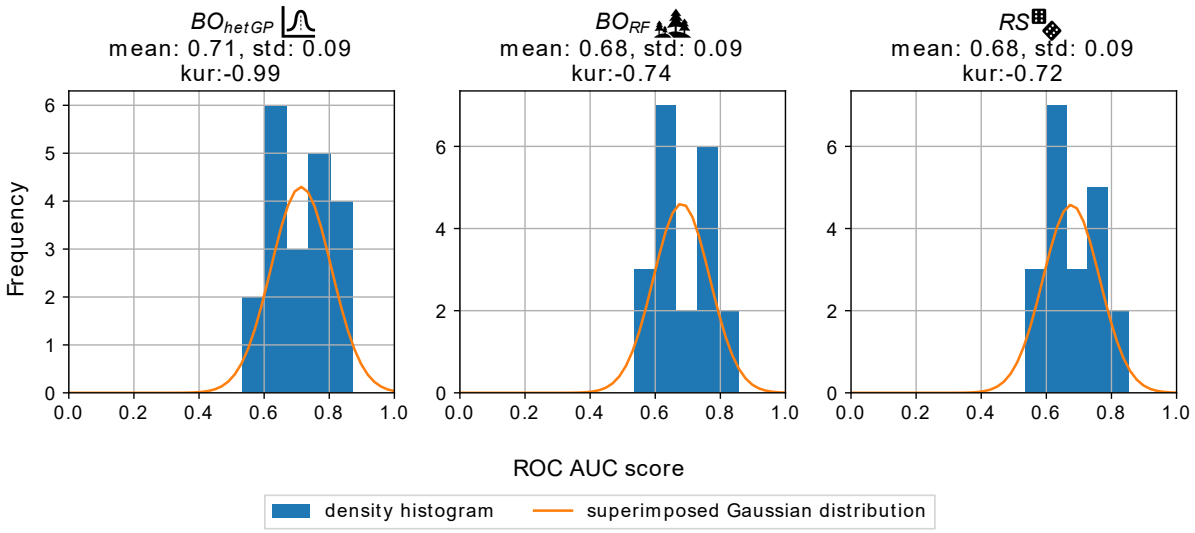


Figure C.3: The distribution of the mean scores per participant, superimposed by a Gaussian distribution that has been fitted upon the data. Obtained from the one-dimensional objective function.

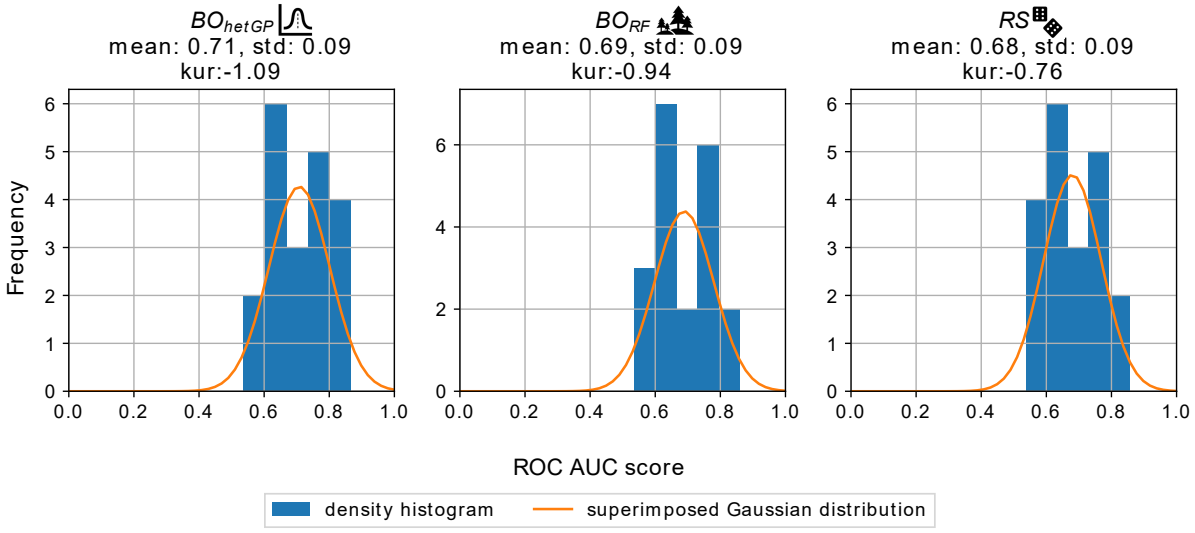


Figure C.4: The distribution of the mean scores per participant, superimposed by a Gaussian distribution that has been fitted upon the data. Obtained from the two-dimensional objective function.

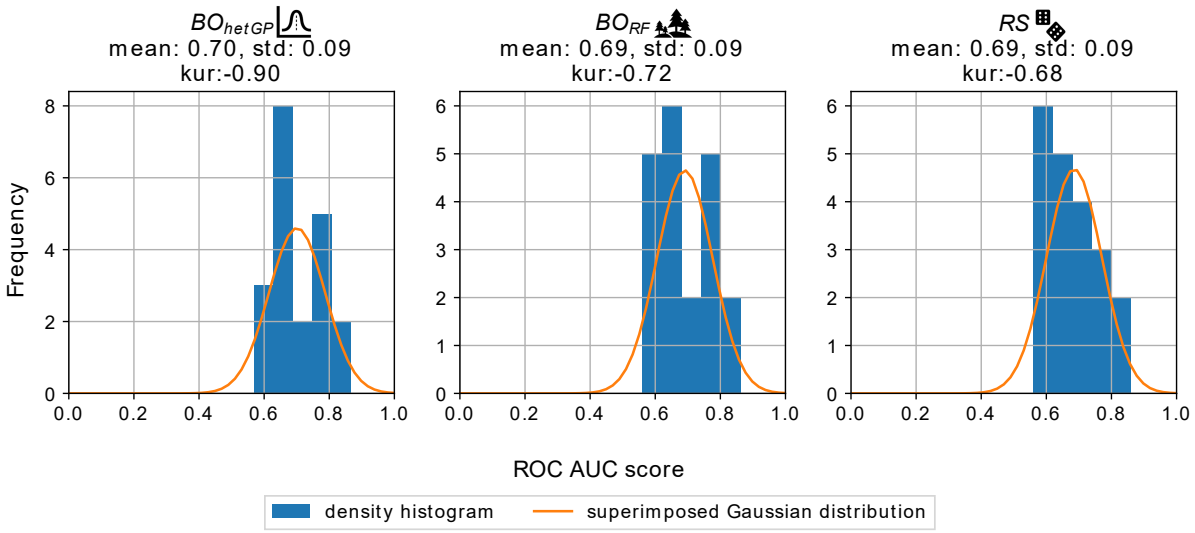


Figure C.5: The distribution of the mean scores per participant, superimposed by a Gaussian distribution that has been fitted upon the data. Obtained from the seven-dimensional objective function.