Master's thesis

Explaining Sequential Model-Based Optimization

Author Federico Croppi

Supervisor Giuseppe Casalicchio

Submitted 29.09.2021

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

DEPARTMENT OF STATISTICS

Acknowledgements

With this thesis, my master's degree comes to an end, and I would like to spend a word for the people that accompanied me and made this journey possible. First of all, I would like to thank my buddies Julian, Esteban, Alex, Sebastian, and Chris, with whom I shared many coffees and laughs during the past years. I will miss spending time with them in the CIP-Pool. A special thank goes to Julian, who has been more than a great friend from day one and always spurred me on to improve myself. Then, I would like to thank my sister Cristina for the countless hours spent studying together during the past year. A very special thank is for my girlfriend Chiara, my rainbow on a rainy day. Finally, my deepest gratefulness is for my parents, Arianna and Bruno, that have always supported me, encouraged me, and guaranteed me a cheerful student life. This work is dedicated to them.

Abstract

This work aims at improving the transparency of sequential model-based optimization (SMBO), a powerful algorithm for hyperparameter (HP) tuning, by explaining the proposals of the acquisition function (AF) with the help of the Shapley value (SV). The SV guarantees a fair distribution of the desirability of a proposed configuration among the involved parameters. What is more, with the linearity axiom and the confidence bound (CB) criterion, the utility of a parameter can be further decomposed into desirability for mean optimization and uncertainty reduction. The contribution of this work is encapsulated in a new tool called `ShapleyMBO`. In the first analysis, using a synthetic setting, we show that our tool provides consistent results with our expectations and that the SV reveals interesting insights of the explore-exploit-trade-off (EETO). In the second analysis, we demonstrate that our tool can also furnish useful diagnostics results in real tuning examples.

Contents

List of Tables

List of Figures

List of Algorithms

Glossary

AA ablation analysis. iv, 5–7, 12, 14, 26, 27, 31, 32

AF acquisition function. ii, 1–3, 5–7, 9, 10, 14, 15, 19, 25, 29, 31, 32, 38

AutoML automated machine learning. 1, 3–5, 13

BO Bayesian optimization. 1

BS Baseline Shapley value. 30, 31, 33

CB confidence bound. ii, 9, 10

CES Conditional Expectation Shapley value. v, 13, 30, 54

EETO explore-exploit-trade-off. ii, 1, 2, 6, 10, 14, 32, 33

EI expected improvement. 9

EPM empirical performance model. 4, 6, 8, 18

fANOVA functional ANOVA. 5, 6

GP Gaussian process. 8, 9

HP hyperparameter. ii, 3–7, 14, 18, 19, 31

HPI hyperparameter importance. 3–5, 7, 14, 33

HPO hyperparameter optimization. 1, 3–5, 8, 31, 32

IC infill criterion. 9

IML interpretable machine learning. 13

LCB lower confidence bound. vii, 1, 2, 4, 10, 14, 16–26, 28, 29, 32, 33, 38

LHS latin hypercube sampling. 16

LPI local parameter importance. 5–7, 14, 21, 32

MBO model-based optimization. 1

ML machine learning. vii, 1, 4, 8

MLP multilayer perceptron. iv, 18, 19, 33

SM surrogate model. vii, 1, 8–10, 14, 15, 54

SMBO sequential model-based optimization. ii, 1–5, 7, 8, 10, 11, 16–18, 31–33, 38, 54

SV Shapley value. ii, iv, 2, 3, 7, 11–15, 19, 26, 27, 29–33, 52, 53

List of Symbols

$\boldsymbol{\Theta}$ hyperparameter space, configuration space of $\boldsymbol{\theta}$

$\boldsymbol{\theta}$ hyperparameter setting, configuration of a ML algorithm

$\boldsymbol{\theta}^*$ solution of an optimization problem

$\boldsymbol{\theta}_t^{new}$ proposed configuration by the acquisition function in iteration t

$\tilde{\boldsymbol{\theta}}$ configuration to explain, explicand

$\hat{f}, \hat{g}$ prediction model, empirical performance model

λ control parameter of the lower confidence bound

$\mathcal{A}$ ML algorithm

$\mathcal{D}$ data set, with $\mathcal{D}_{train}$ training set and $\mathcal{D}_{valid}$ validation set

$\mathcal{L}$ loss metric

$\Pi(P)$ set of all permutations of the grand coalition with elements π

$contr(j)$ contribution of hyperparameter j or θ_j

m mean prediction of the surrogate model, equivalent to $\hat{\mu}$

P HP set with p parameters, grand coalition with p players

$Pre_\pi(j)$ set of predecessors of player j in permutation π

S subset of P

se uncertainty prediction of the surrogate model, equivalent to $\hat{\sigma}$

v characteristic or contribution function

1 Introduction

With the advent of automated machine learning (AutoML) automated hyperparameter optimization (HPO) has recently become a popular research field. Automated HPO is particularly beneficial because, among others, it can improve the performance of algorithms and facilitate the use of machine learning (ML) applications [12, p.3]. Out of the many methods available (refer to [25] for an overview) SMBO, also known as Bayesian optimization (BO) or simply model-based optimization (MBO), has shown to be a valid alternative to well established methods such as random search [2]. SMBO has gained much of its success because of its sample efficient capacity to optimize expensive black-box target functions, that are, like in the case of some ML algorithms' performance, analytically intractable and expensive to evaluate. Starting from a set of observations with evaluated target values (initial design), the same steps are repeated in an iterative fashion. First, a surrogate model (SM) is fitted to the given design. Then an AF, for instance the lower confidence bound (LCB), uses the predictive distribution of the SM to choose which point in the input space to evaluate. Among all candidates, the chosen point has the highest acquisition and it is considered the most promising and desirable point for improving the optimization. Essentially the choice of the AF is based on mean and uncertainty prediction of the SM. A good mean prediction is desirable because it can indicate good target values. High uncertainty is appealing because it helps to increase knowledge about the target space. Ideally, proposals fulfill both desirability properties, but eventually this is not possible and the algorithm is faced with a decision problem, which is known in the literature as the EETO [8, p.2]. The phenomenon states that it is impossible to optimize a promising region locally (mean optimization) and simultaneously explore unknown areas of the space (uncertainty reduction). This decision problem is indeed tackled by the AF, which proposes points balancing exploration and exploitation and finally allows the algorithm to search for the optimum efficiently.

Undoubtedly the AF, or more precisely the choice of the AF, plays a fundamental role in the optimization problem, and yet it lacks transparency. When new instances are proposed, explanations behind the choice of the AF are possible only in low dimensional problems. For instance, take a look at figure 1, which displays the results of a univariate BO optimization run. In the first iteration the AF proposes a point with a relative good mean prediction, but not the best, and high uncertainty, but not the highest (explore-exploit-trade-off)[1]. Both mean and uncertainty positively

[1] Here, relative means compared to other (candidate) points in the domain.

influence the desirability of the proposal. This must not necessarily hold for the entire process and their influence can change over time. Indeed as shown in the third and final iteration of the process, the choice is mostly dictated by the mean as the proposed point has a very low mean prediction and relatively small uncertainty. When dealing with higher dimensional problems, which are far from rare in SMBO applications, such explanations becomes more complicated, or even impossible, because multiple, possibly interacting parameters influence the AF choice. To give an intuition on the seriousness of the problem, consider the same optimization problem, but with three dimensions (results are displayed in table 1). Which parameter among x_1, x_2 and x_3 has the biggest influence on the LCB? Was the value of x_1 primarily proposed because of its effect on the mean or uncertainty of the proposal? How do contributions between parameters differ? How do the contributions of each parameter change over time?

iter	x_1	x_2	x_3	y	$\hat{cb}$	$\hat{se}$	$\hat{m}$
1	0.15	0.41	0.47	0.33	0.16	0.32	0.48
2	0.61	0.16	0.43	0.67	0.18	0.27	0.45
3	-0.30	0.32	0.49	0.39	0.17	0.23	0.40

Table 1: Same problem as in figure 1 but in three dimensions. $\hat{cb}, \hat{se}$ and $\hat{m}$ indicate the (predicted) LCB, uncertainty and mean of the proposals.

Given the limited visualization possibilities in higher dimensions and the scarce information about the black-box function in real life, users are usually left with little information regarding the choice of specific parameter values. On the contrary, users shall understand *why* the algorithm selects specific configurations. This information is essential to increasing trust and transparency of SMBO, which otherwise can only be treated as a black-box algorithm. To the best of our knowledge, there is no current work that tackles this problem and this work aims indeed to fill this research gap. Without much further ado, the desirability concept mentioned previously can be applied to multidimensional problems by splitting the desirability of a proposal into individual desirability scores of its parameters [2]. As such, the solution to the problem reduces to solving a **local parameter attribution problem**. Among possible alternatives, we will show that the best solution to reveal previously unknown aspects of the EETO is the Shapley value (SV) [34], a concept that originates form cooperative game theory but has become a state of the art local interpretation method for prediction models. This method is preferred over its competitors because of its (i) fair distribution of the payout between interacting parameters and

[2] A parameter value, or simply a parameter, is said to be desirable if it has a positive influence on the acquisition of the proposal and hence a positive influence on the mean or uncertainty prediction.

(ii) strong theoretical properties, especially the linearity axiom. To the best of our knowledge, this is the first time the SV is used to explain SMBO and as such the main goals of this work are to give exploratory evidence that the SV is a valid method to explain the choices of AF and to introduce a new framework named `ShapleyMBO`, which users can benefit from to hopefully increase trust and transparency of their SMBO tuning applications. To accomplish that, the analysis will be twofold. In the first part, results for different AF settings are tested against our expectations using a function with known analytical properties. In the second part, a tuning example is conducted to see how `ShapleyMBO` reacts in a real scenario and to show how users might benefit from it. The remainder of the thesis is structured as follows. After useful definitions and notations in section 2, an overview of related works regarding explanation tools for AutoML in general and HPI specifically is given in section 3. Then, selected theoretical concepts regarding SMBO (section 4) and the SV (section 5.1) are explained. Section 5 is further extended in 5.2 with a brief introduction of the SV in the context of SMBO and the presentation of `ShapleyMBO`. Before getting to the results of both analyses in section 7, the experimental setup is described separately in section 6. Finally, methodological aspects of `ShapleyMBO` are discussed in section 8. The paper ends in section 9 with a summary and outlook for future work. Last but least in the appendix part all figures and algorithms, as well as additional materials, can be found. All data and codes are available at `https://github.com/croppi-f/ShapleyMBO`.

2 HPO: problem definition and notation

In this section the hyperparameter optimization problem is formalized and the notation adopted in this paper is introduced. For a comprehensive overview about current topics in HPO the curious reader is referred to [12]. The HPO problem is also known in the literature as the algorithm configuration problem [17, p.8], which can be defined as follows. Let $\mathcal{A}$ be an algorithm with hyperparameters $\theta_1, \ldots, \theta_p$ and domain $\Theta_1, \ldots, \Theta_p$ such that $\boldsymbol{\Theta} = \Theta_1 \times \ldots \times \Theta_p$ is the HP or configuration space, which can be real-valued, categorical, binary, conditional or even mixed [12, p.5]. Further, let $\mathcal{L}(\mathcal{A}_{\boldsymbol{\theta}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ quantify the validation loss of algorithm $\mathcal{A}$ achieved on $\mathcal{D}_{valid}$, that has been previously trained with configuration $\boldsymbol{\theta}$ on $\mathcal{D}_{train}$. Then

given data set $\mathcal{D}$ the goal is to find the optimal $\boldsymbol{\theta}^*$ for equation 1a.

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \mathbb{E}_{\mathcal{D}_{train},\mathcal{D}_{valid}\sim\mathcal{D}} \mathcal{L}(\mathcal{A}_{\boldsymbol{\theta}}, \mathcal{D}_{train}, \mathcal{D}_{valid}) \tag{1a}$$

$$\approx \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}(\mathcal{A}_{\boldsymbol{\theta}}, \mathcal{D}_{train}^{(k)}, \mathcal{D}_{valid}^{(k)}) \tag{1b}$$

$$\approx \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \hat{g}(\boldsymbol{\theta}) \tag{1c}$$

Because of limited data availability in practice the expectation in equation 1a can be approximated with the help of resampling strategies like $K\text{-}fold$ cross-validation [5, p. 254] (equation 1b). To speed up the process, recently Hutter et al. suggest approximating the interesting quantity with a so-called empirical performance model (EPM) $\hat{g} : \boldsymbol{\Theta} \to \mathbb{R}$, namely a powerful prediction model [20, p.79]. The optimization problem can then be summarized with equation 1c, which reflects a hypothetical optimization problem that can be solved using SMBO.

Several synonyms in the HPO literature define the same object. Throughout the thesis, the HP space $\boldsymbol{\Theta}$ will be called configuration, parameter, or input space, and an element thereof $\boldsymbol{\theta}$ will be called HP setting, configuration, instance, or observation. To avoid confusion with the HP of the SMBO algorithm, like the control parameter λ of the LCB, if needed, we will explicitly mention what we are referring to. In conclusion, note that concepts in sections 4 and 5 are usually applied to features $\boldsymbol{x}$ of a ML model. Still, the same concepts can be applied to configurations $\boldsymbol{\theta}$ because within SMBO for tuning HP are effectively parameters or "features" of the optimization problem. Hence notations will be adjusted accordingly. Finally, as regarding HPI, notice that importance, contribution, and influence will be used as synonyms.

3 Related work

A recent study conducted by Drozdal et al. highlighted the need for transparency in AutoML applications and HPO [10]. In particular the authors identify three types of transparency: *model-oriented* (e.g. shows various performance metrics), *data-oriented* (e.g. shows the data distribution) and finally *process-oriented* (e.g. shows how AutoML performs HPO) [10, p.300]. To overcome this issue several frameworks have been offering solutions to explain the results of a tuning process [15, 22, 23, 28, 40]. While all of them offer abundant analysis tools, only [15, 22, 23] implement

the HPO task within their framework[3] and in particular only Google Vizier uses, among other algorithms, SMBO as a tuning strategy [15, p.1491][4]. This shrinks the universe of the comparable related work within the AutoML applications to only Google Vizier. The major advantage of Google Vizier's analysis tool is the interactive dashboard, with which the user can for instance display various "performance-over-time metrics" (*model-oriented*) or analyze HP trajectories with parallel plots (*data-oriented*) [15, p.1490]. To the best of our knowledge, this is much of what Google Vizier offers in terms of transparency and, if this is the case, it does not directly provide a solution to our research question, namely a tool that rather explains how SMBO performs HPO (*process-oriented* transparency).

Outside the context of AutoML, HPO, seen as an independent research field, has a long history [12, p.4] and in recent years, much attention within this field was given to assessing the influence of HPs on the performance of an algorithm. This branch of HPO is known as hyperparameter importance (HPI) and state-of-the-art methods today include (global) functional ANOVA (fANOVA) [18], local parameter importance (LPI) [4] and ablation analysis (AA) [11]. Since the goal in this work is to assess the importance of the HP on the AF, basically any of the just mentioned HPI methods can be adapted to our purposes by replacing the performance metric with the acquisition function[5]. In the following paragraphs, though, we will explain why these methods are not sufficient.

LPI and fANOVA are strictly related, as the former was introduced as the local variant of the latter. Since this work aims to explain individual configurations, namely the proposals in each iteration of the SMBO process, we exclude the latter method as a possible solution but notice that the considerations regarding LPI also apply to fANOVA. The idea of LPI is based on the concept of functional decomposition, after which a function can be decomposed into additive components of parameter interactions [16]. Among others, functional decomposition fulfills a property called variance decomposition [16, p.714]. LPI exploits this property and decomposes the variance of the algorithm's performance to measure the importance of a parameter in the neighborhood of a configuration (therefore the name *local*) [4, p.124]. Given a configuration space $\boldsymbol{\Theta}$ of p hyperparameters with $P = \{1, 2, \ldots, p\}$ being the set of all HP, a configuration $\boldsymbol{\theta}$ and validation loss $\mathcal{L}(\mathcal{A}_{\boldsymbol{\theta}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ estimated with

[3] As stated in [28, p.1] and [40, p.2] their framework can be integrated within AutoML systems, yet we do not know if they provides specific solutions for interpreting SMBO results.

[4] To the best of our knowledge both Hypertuner and H2O do not implement SMBO as a tuning strategy [22, p.1] [23, p.19].

[5] Notice also that Golovin et al. do not mention any HPI analysis tools in their paper and we can not say if Google Vizier offers any.

EPM $\hat{g}$, the local contribution of parameter $j \in P$ with domain Θ_j can be formulated as

$$contr_{LPI}(j) = \frac{Var_{a \in \Theta_j} \hat{g}(\boldsymbol{\theta}[\theta_j = a])}{\sum_{l \in P} Var_{b \in \Theta_l} \hat{g}(\boldsymbol{\theta}[\theta_l = b])}, \tag{2}$$

where $\boldsymbol{\theta}[\theta_j = a]$ indicates configuration $\boldsymbol{\theta}$ with value of parameter j equal to a. A HP is said to be important if it accounts for a large fraction of the variance of the performance [39, p.2370]. Notice that Biedenkapp et al. indeed use an EPM to approximate the performance [4, p.124]. Now imagine LPI is used to assess the desirability of a configuration proposed by the AF (in equation 2 $\hat{g}$ is replaced with the AF). Since $contr_{LPI}$ is element of $\mathbb{R}_0^+$ the contribution of a parameter can only be assessed in absolute terms. To explain the EETO it is essential to distinguish also the sign of the contribution because the AF might choose configuration values that are convenient for one scope but not for the other. Finally fANOVA and therefore also LPI, are limited to the decomposition of the variance up to low order interactions [18, p.757]. This could bring misleading results if higher order interactions play a substantial role.

Previous to LPI, AA was proposed as a complementary local method to fANOVA [11]. Starting from a source configuration $\boldsymbol{\theta}^s$ each parameter value is sequentially flipped according to the value of a target configuration $\boldsymbol{\theta}^t$. Normally as source a default configuration provided in libraries and as target, a tuned configuration is chosen [11, p.435]. The flipping sequence is determined to maximize the gain over previous configurations[6]. The bigger the improvement the more important a HP. For a detailed description of the method refer to [11, p.436]. As shown in equation 3 within AA the contribution of a HP is element of $\mathbb{R}$ and therefore, unlike LPI, it would allow different contribution signs. However, this method hides a disadvantage, implicit in its greedy and iterative way of measuring importance. The problem occurs when parameters interact. Consider again the notation of the previous paragraph and notice that also AA has been improved with EPMs [3]. The idea is to measure the importance of parameter j by computing the marginal contribution of its target value θ_j^t at the time it has been flipped. For instance, in the first ablation round, it is defined as

$$contr_{AA}(j) = \hat{g}(\boldsymbol{\theta}^s) - \hat{g}(\boldsymbol{\theta}^s[\theta_j^s = \theta_j^t]). \tag{3}$$

In equation 3 the value of θ_j^s is flipped according to θ_j^t, leaving the remaining HP unchanged to their source values. As the process goes on, one source value in each round is changed until all have been flipped. The particularity of this greedy procedure is that parameter values remain unchanged after being switched. Therefore the marginal contribution of parameters after the first round is computed conditional

[6] This is the general workflow when lower target values indicate higher performance.

on other parameters being previously flipped. In other words, AA does not consider interactions between the HP, which could cause misleading results in the presence of any. Indeed Fawcett and Hoos state in a conclusive remark that an "avenue for further work is to make support for complex parameter interdependencies more flexible, for example [...] to allow sets of parameters without conditional relationships to be modified in the same ablation round" [11, p.456]. For a better intuition on this issue consider the following data generating model

$$u = \theta_1 + \theta_2 \cdot \theta_3 + \epsilon$$
$$\theta_1, \theta_2, \theta_3 \overset{i.i.d}{\sim} \mathcal{U}(0,1),\ \epsilon \sim \mathcal{N}(0, 0.05^2),$$

where u denotes the AF. Further suppose that u is minimized. According to the model for equal parameter values we expect θ_2 and θ_3 to have same contributions and smaller than θ_1 because of their interaction. To assess the importance of the parameters an AA is conducted. A training data set with 10000 instances is simulated and a random forest is trained to predict u. Source and target are respectively $\boldsymbol{\theta}^s = (0.5, 0.5, 0.5)^T$ and $\boldsymbol{\theta}^t = (0, 0, 0)^T$. Results are displayed in table 2. While results for the non interacting θ_1 meet the expectations, results for the interacting parameters do not, since θ_3 is far less important than θ_2 (see relative AA). The former parameter is left with little contribution since most of their joint contribution is absorbed by the latter. This is clearly unfair for θ_3, which has a very similar, unconditional effect in the first round of the analysis (see AA round 1). To assess contributions correctly a method that can better handle interactions is necessary.

parameter	AA round 1	relative AA	relative SV
θ_1	0.483	0.66	0.66
θ_2	0.262	0.31	0.21
θ_3	0.258	0.03	0.17

Table 2: Comparison of ablation analysis and Shapley value results for data generating model in section 3. The source configuration in AA was chosen s.t. similar payouts (quantity distributed) with both methods are obtained. The flipping order resulting from AA was θ_1, θ_2, θ_3. Columns from left to right: parameter, contributions in the first round of the AA, relative importance with AA (final results) and SV. The latter results will be presented later in section 5.2. Relative importance is the marginal contribution divided by the payout. The approximation error of SV ($0.66 + 0.21 + 0.17 > 1$) is negligible and does not change the interpretation of the results.

In conclusion, both HPI methods, LPI and AA, have few disadvantages and as shown later in section 5.2 the Shapley value turns to be a better choice for this work. Before getting there, the core theory of SMBO will be presented in the next section.

4 SMBO for hyperparameter optimization

Sequential model-based optimization is a popular method for the global optimization of expensive black-box functions. One of its most successful applications is indeed HPO [2, 17, 35, 38], as evaluating the performance of a ML algorithm can be quite expensive and the target function, i.e. the validation loss, is usually analytically intractable (*black-box*) [12, p.4]. If an EPM is used to approximate the interesting quantity, based on the notations provided in section 2 the optimization problem can be formalized as $\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \hat{g}(\boldsymbol{\theta})$. It is realistic to assume that the true objective is noisy. That is, repeated target evaluations of the same argument $\boldsymbol{\theta}$ result in different target values. In the case of HPO this observation noise, also called nugget effect, may be caused for instance by randomness implicit in resampling strategies (see equation 1b). For the sake of simplicity assume that this noise ϵ is additive. Then, the objective function becomes $\Psi(\boldsymbol{\theta}) = \hat{g}(\boldsymbol{\theta}) + \epsilon$ with $\Psi : \boldsymbol{\Theta} \to \mathbb{R}$ and the optimization problem can be finally summarized as $\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \Psi(\boldsymbol{\theta})$.

The core steps of the SMBO procedure are now explained along with algorithm 1. For further details, we refer the curious reader to the tutorials provided in [8, 33]. Given a set of instances with evaluated target values $\mathcal{D}$, also known as design, the first step is to approximate Ψ by fitting a surrogate (prediction) model $\hat{f}$ on $\mathcal{D}$. It is essential that for any instance $\boldsymbol{\theta}$ the SM provides an estimate of the target value as well as an estimate for the uncertainty of that prediction. A popular choice is the Gaussian process (GP) regression model [21][7]. Before getting to the regression model, let us clarify important concepts of a Gaussian process. A GP is an infinite-dimensional stochastic process, an infinite set of random variables, with any finite sample thereof being Gaussian distributed [8, p.7].

Definition 1 (Gaussian Process)**.** *A function $f(\boldsymbol{\theta})$ is generated by a Gaussian process $\mathcal{GP}\left(m(\boldsymbol{\theta}), k(\boldsymbol{\theta}, \boldsymbol{\theta'})\right)$ if for any finite set of points $\{\boldsymbol{\theta}^{(i)}\}_{i=1}^{n}$ the vector $\boldsymbol{f} = (f(\boldsymbol{\theta}^{(1)}), \ldots, f(\boldsymbol{\theta}^{(n)}))$ follows a (multivariate) Gaussian distribution*

$$\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{K})$$

$$\boldsymbol{m} = (m(\boldsymbol{\theta}^{(1)}), \ldots, m(\boldsymbol{\theta}^{(n)})), \; \boldsymbol{K} = \begin{pmatrix} k(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(1)}) & \cdots & k(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(n)}) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{\theta}^{(n)}, \boldsymbol{\theta}^{(1)}) & \cdots & k(\boldsymbol{\theta}^{(n)}, \boldsymbol{\theta}^{(n)}) \end{pmatrix},$$

where $\boldsymbol{m}$ is the mean vector and $\boldsymbol{K}$ is the covariance or kernel matrix.

Hence, a GP places a probability distribution over functions and, like a Gaussian

[7] Another possibility would be to use Random Forests [19].

distribution, it is fully specified by a mean function m and kernel function k. While the former reflects the trend, the latter, in a nutshell, determines the smoothness of the functions $\boldsymbol{f}$. An equivalent interpretation would be that the kernel function measures the similarity between two points $\boldsymbol{\theta}$ and $\boldsymbol{\theta'}$ [17, p.132]. A common choice for the kernel function is the $\frac{3}{2}$-Matérn kernel [32, p.85], which for any two points is defined as

$$k(\boldsymbol{\theta}, \boldsymbol{\theta'}) = \left(1 + \frac{\sqrt{3}\|\boldsymbol{\theta} - \boldsymbol{\theta'}\|}{\ell}\right) \exp\left(-\frac{\sqrt{3}\|\boldsymbol{\theta} - \boldsymbol{\theta'}\|}{\ell}\right), \tag{4}$$

where ℓ denotes the characteristic length scale and $\|\boldsymbol{\theta} - \boldsymbol{\theta'}\|$ the distance between two points (e.g. the euclidean norm). Using the properties of the Gaussian distribution and the notions just explained, the predictive *posterior* distribution can be derived.

Definition 2 (Predictive Posterior Distribution)**.** *Given a GP with* $\boldsymbol{m} = 0$ *and kernel function* k*, design* $\mathcal{D} = \{(\boldsymbol{\theta}^{(i)}, f(\boldsymbol{\theta}^{(i)})\}_{i=1}^{n}$ *of input matrix* $\boldsymbol{\Theta}$ *and vector* $\boldsymbol{f}$ *and finally observation noise* $\epsilon \overset{i.i.d}{\sim} \mathcal{N}(0, \sigma^2_{noise})$*. Then, the predictive (posterior) distribution of a function* f *at input* $\boldsymbol{\theta}^{(n+1)}$*, or simply* $f(\boldsymbol{\theta}^{(n+1)})$*, is*

$$f|\, \boldsymbol{\theta}^{(n+1)}, \boldsymbol{\Theta}, \boldsymbol{f} \sim \mathcal{N}\left(\mu,\, \sigma^2\right)$$

$$\textit{where}$$

$$\mu = \mu(\boldsymbol{\theta}^{(n+1)}) = \boldsymbol{k}^T \left[\boldsymbol{K} + \sigma^2_{noise}\boldsymbol{I}\right]^{-1} \boldsymbol{f},$$

$$\sigma^2 = \sigma^2(\boldsymbol{\theta}^{(n+1)}) = k(\boldsymbol{\theta}^{(n+1)}, \boldsymbol{\theta}^{(n+1)}) + \sigma^2_{noise} - \boldsymbol{k}^T \left[\boldsymbol{K} + \sigma^2_{noise}\boldsymbol{I}\right]^{-1} \boldsymbol{k},$$

$$\boldsymbol{k} = (k(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(n+1)}), \ldots, k(\boldsymbol{\theta}^{(n)}, \boldsymbol{\theta}^{(n+1)})),$$

$\boldsymbol{K}$ *is the Kernel matrix of definition 1 and* $\boldsymbol{I}$ *is the Identity matrix.*

Now explaining the surrogate model is straightforward. The GP regression model $\hat{f}$ is the estimated predictive posterior distribution of definition 2, where predictions for target values are obtained with estimated posterior mean $\hat{\mu}$ and their uncertainty with the estimated posterior variance $\hat{\sigma}^2$.

After the SM has been trained, in a second step both mean and uncertainty prediction are passed on to the acquisition function, also known in the literature as infill criterion (IC), which has the role to guide the search for the optimum [8, p.11]. Thereby the AF is optimized and unlike the target, this optimization problem is computationally inexpensive [8, p.6]. For a better intuition consider the AF as a utility function $u : \boldsymbol{\Theta} \to \mathbb{R}$, which measures the utility of a point $u(\boldsymbol{\theta}|\mathcal{D})$ conditional on the surrogate model previously fitted on design $\mathcal{D}$. Among many infill criteria proposed so far, the confidence bound (CB), first introduced by Cox and John [9], is arguably a valid alternative to more complex criteria like the EI [6, p.6]. When deal-

ing with a minimization problem the CB is usually known as the lower confidence bound.

Definition 3 (Lower Confidence Bound)**.** *The LCB of an instance* $\boldsymbol{\theta}$ *is*

$$cb(\boldsymbol{\theta}, \lambda) = \hat{\mu}(\boldsymbol{\theta}) - \lambda\hat{\sigma}(\boldsymbol{\theta}),$$

where $\lambda > 0$ *is a control parameter set by the user before the optimization starts.*

For a better intuition, throughout the paper we will refer to the *cb* simply as $m - \lambda se$. The simplicity of the LCB is evident: an instance is desirable (has a high utility) if (i) the mean prediction m is low (the target value is expected to be low) or (ii) the uncertainty prediction *se* is high (the model has scars information about the target function in that area). When the AF subsequently proposes points with a better mean and relatively low uncertainty, then the algorithm is said to *exploit* a potentially promising region in the input space. Conversely, when the AF proposes points with high or increasing uncertainty, then it is said to *explore*. Given the nature of the optimization, it is hard to say which goal is more important. Whereas exploitation is important to converge towards an optimum, in the case of multi-modal objective functions, exploration can avoid getting stuck in local optima and consequently avoid missing the global optimum. As mentioned in the introductory section, both duties can not be fulfilled (EETO), yet the algorithm's behavior can be controlled with λ and higher values force it to explore more. Although inexpensive, the optimization of the AF requires numerical methods, so-called *infill optimizers*, because, as the target function, the AF is analytically intractable. Bischl et al. propose a method called *focussearch*, a search over the input space that enables to focus on promising regions by repeatedly shrinking the space around points with highest utility [6, p.6]. One of the strengths of SMBO relies indeed in the infill optimization because through the EETO the AF can select points efficiently and this implicitly minimizes the number of target function's evaluations in the optimization. [8, p.3].

Once the best point $\boldsymbol{\theta}^{new}$ is found the target function is evaluated $\Psi(\boldsymbol{\theta}^{new})$ and the design $\mathcal{D}$ is updated with the pair $(\boldsymbol{\theta}^{new}, \Psi(\boldsymbol{\theta}^{new}))$. Training a SM and optimizing AF are the heart of SMBO and these two steps are repeated until a termination condition ends the process. When the optimization is limited by a budget, one possible termination condition is to set a maximal number of target evaluations [6, p.7]. Finally, when the optimization is over, the best point observed in the process or, in case of a noisy function, the point with the best prediction according to the final SM is returned [6, p.7].

5 The Shapley value

In section 5.1 the theoretical background, as well as the computation of the SV, are presented. In section 5.2 we describe its application to SMBO and present `ShapleyMBO`.

5.1 Theory and computation

Most of the theoretical aspects in the upcoming paragraphs are taken from chapters 9, 17, and 18 of [29] and, if not otherwise stated, we refer to it using only page numbers in brackets. The Shapley value was originally introduced by Lloyd Shapley as a solution concept to attribution problems in the field cooperative games with transferable utility (TU-game), which can be defined as follows [p.153].

Definition 4 (TU-game)**.** *A cooperative game with transferable utility is a pair* (P, v)*, where* $P = \{1, \dots, p\}$ *with* $p \in \mathbb{N}$ *is the set of players, and* $v : 2^P \to \mathbb{R}$ *is a function assigning to each coalition* S*, i.e. to each subset* $S \subseteq P$ *a real number* $v(S)$*, such that* $v(\emptyset) = 0$*. The function* v *is known as the characteristic function, or contribution function, and* $v(S)$ *is the worth of coalition* S*. The coalition* P *is called the grand coalition. A payoff distribution for coalition* S *is a vector of real numbers* $(x_i)_{i \in S}$*.*

Given (P, v), let $\Pi(P)$ be the set of all permutations of the grand coalition and π be a permutation in $\Pi(P)$. Further let $Pre_\pi(j)$ be the coalition consisting of the predecessors of player j in π and $v(Pre_\pi(j) \cup \{j\}) - v(Pre_\pi(j))$ the marginal contribution of player j to that coalition. Then, the Shapley value is defined as follows.

Definition 5 (Shapley Value)**.** *The Shapley value* ϕ *is a map, which assigns to every TU-Game* (P, v) *a unique payoff distribution* $\boldsymbol{\phi}(v) = (\phi_1(v), \dots, \phi_p(v))$*, where* $\phi_j(v)$ *is the average marginal contribution of player* j *and*

$$contr_{SV}(j) = \phi_j(v) = \frac{1}{p!} \sum_{\pi \in \Pi(P)} v(Pre_\pi(j) \cup \{j\}) - v(Pre_\pi(j)). \tag{5}$$

In a nutshell, the SV of player j is the average of all its marginal contributions. What is more, the marginal contribution of player j is path independent [p.324]. In other words, it is irrelevant how the players before as well as after j are ordered in each permutation of $\Pi(P)$ because the worth of the coalition does not change. As a consequence, exactly $|S|!\,(p-1-|S|)!$ times an arbitrary order $Pre_\pi(j)$ in equation 5

is represented by the same coalition S, a subset of P not containing j ($S \subseteq P \backslash \{j\}$). The first factor refers to the number of permutations of S (players preceding j) and the second factor to the permutations of $P \backslash (S \cup \{j\})$ (players after j) [p.307]. This considerations lead to an equivalent, permutation independent, definition of $\phi_j(v)$ as the weighted sum of a player j's marginal contributions **over all possible coalitions** in game (P, v) (equation 6). Notice that players involved in a coalition are considered equally responsible for its worth and hence they are remunerated equally within that coalition [p.311]. This is a crucial distinction to the ablation analysis.

$$contr_{SV}(j) = \phi_j(v) = \sum_{S \subseteq P \backslash \{j\}} \frac{|S|!\,(p-1-|S|)!}{p!} \left[v(S \cup j) - v(S)\right] \tag{6}$$

The Shapley value is a fair solution because it fulfills the axiomatic properties of dummy player, efficiency, linearity, and symmetry. The dummy player property says that if a player only contributes his worth to any coalition, then it should also be rewarded accordingly [p.309][8]. The efficiency axiom tells that the game payout to be distributed among all participating players equals the worth of the grand coalition (given $v(\emptyset) = 0$). Following the linearity axiom, there should not be any difference between the reward of a combined game and the reward obtained in a linear combination of separated games [p.308, 331]. The symmetry axiom states, that if two players have equal contributions to any coalition they should be rewarded equally in the game [p.308].

Dummy player. *If* $v(S \cup \{j\}) - v(S) = v(j)$ *for player* j *and all* $S \subseteq P \backslash \{j\}$*, then* $\phi_j(v) = v(j)$.

Efficiency. $\sum_{j=1}^{p} \phi_j(v) = v(P) - v(\emptyset)$

Linearity. *Given two games* (P, v_1) *and* (P, v_2)*, for* $a, b \in \mathbb{R}$ *it holds*

$$\phi_j(av_1 + bv_2) = a\phi_j(v_1) + b\phi_j(v_2)$$

Symmetry. *If* $v(S \cup \{j\}) = v(S \cup \{l\})$ *for players* j, l *and every* $S \subseteq P \backslash \{j, l\}$*, then* $\phi_j(v) = \phi_l(v)$.

A particularity of the SV is the flexibility of the contribution function, which does not require any specific properties and the SV can hence be used in many different applications [13, p.3]. Indeed recently, not only it has become a state of the art

[8] Note that this property also includes the Null-player property, after which a player should be rewarded with 0 if its contribution is zero to any coalition including $S = \emptyset$ [p.308].

method within the field of interpretable machine learning (IML), but it has also been adapted in AutoML to explain algorithm selection problems [13]. Within the context of IML the SV falls under the umbrella of the model agnostic local interpretation methods. In particular, it breaks down a model's prediction into feature contributions for single instances (feature attribution method). While it is intuitive that the features become the players, coalitions become feature interactions, and their worth becomes the model's prediction, it is unclear how to define the contribution function. Up to today many variants have been proposed (refer to [37] for an overview) and arguably the most prominent choice is the conditional expectation of the model prediction [24,36]. This method justifies the concept of missing or ignored features excluded from the coalition by computing the expectation of the model output conditional only on the subset of known features that form the coalition. Let $\hat{f} : \boldsymbol{\Theta} \to \mathbb{R}$ be a prediction model, $\tilde{\boldsymbol{\theta}}$ the instance to explain or *explicand*, then the worth of a coalition of features S is given by $v(S) = \mathbb{E}[\hat{f}(\boldsymbol{\theta})|\boldsymbol{\Theta}_S = \tilde{\boldsymbol{\theta}}_S]$. It follows that the expected model output given no information about the feature values $v(\emptyset)$ is $\mathbb{E}[\hat{f}(\boldsymbol{\Theta})]$. A common assumption of the Conditional Expectation Shapley value (CES) is feature independence. More formally, if $\bar{S} = P \backslash S$ is the complement of S then $v(S)$ simplifies to

$$\begin{aligned} v(S) &= \mathbb{E}[\hat{f}(\boldsymbol{\theta})|\boldsymbol{\Theta}_S = \tilde{\boldsymbol{\theta}}_S] \\ &= \mathbb{E}[\hat{f}(\boldsymbol{\theta}_S, \boldsymbol{\theta}_{\bar{S}})|\boldsymbol{\Theta}_S = \tilde{\boldsymbol{\theta}}_S] \\ &= \int \hat{f}(\boldsymbol{\theta}_S, \boldsymbol{\theta}_{\bar{S}})\, p(\boldsymbol{\theta}_{\bar{S}}|\boldsymbol{\Theta}_S = \tilde{\boldsymbol{\theta}}_S)\, d\boldsymbol{\theta}_{\bar{S}} \\ &\approx \int \hat{f}(\boldsymbol{\theta}_S, \boldsymbol{\theta}_{\bar{S}})\, p(\boldsymbol{\theta}_{\bar{S}})\, d\boldsymbol{\theta}_{\bar{S}}. \end{aligned} \tag{7}$$

Last but not least the exact computation of the SV has an exponential time complexity [36, p.651] and therefore approximations are almost inevitable to reduce the computational burden. `ShapleyMBO` uses (through the {`iml`} package) the method proposed by Štrumbelj and Kononenko, which is based on Monte Carlo sampling. The core steps of the estimation are summarized in algorithm 2. Notations in the algorithm are also taken from section 5.9.3.3 in [26]. Although the choice of the optimal Monte Carlo samples K is less clear [26, section 5.9.5], if enough are provided, then according to the central limit theorem

$$\hat{\phi}_j(v) \overset{a}{\sim} \mathcal{N}\left(\phi_j(v),\ \frac{\sigma_j^2}{K}\right),$$

that is $\hat{\phi}_j(v)$ is an unbiased and consistent estimator of the true SV $\phi_j(v)$ [36, p.652]. Before proceeding, here are the main characteristics of the SV: it has solid theoretical properties, can nicely handle interactions and finally the contribution of a player or feature is computed relative to the reference value $v(\emptyset)$ (efficiency axiom).

5.2 The Shapley value within SMBO

When moving from prediction models to acquisition functions the use of the SV is straightforward, since an AF is nothing but a transformed surrogate model. Hence the SV can be applied to any AF to assess the desirability of the chosen parameter values. Yet among all AFs, the LCB, thanks to its simple functional form, makes the SV the perfect method for this work. Let the choice of the LCB be a TU-game defined as (P, cb), or two games (P, m) and (P, se), with P being the grand coalition of HP involved in the choice problem and respectively m and se the mean and uncertainty prediction of the SM. Then according to the linearity axiom the cb contribution of any parameter j of explicand $\tilde{\boldsymbol{\theta}}$ can be decomposed into **mean contribution** and **uncertainty contribution** (equation 8).

$$\phi_j(cb) = \phi_j(m - \lambda se) = \phi_j(m) - \lambda\phi_j(se) \tag{8}$$

The linearity axiom, therefore, enables not only to assess for each parameter the overall desirability of the chosen parameter value ($\phi(cb)$), but it allows to go one step further and bring to light previously unknown aspects of the explore-exploit-trade-off, namely to understand how both contributions $\phi(m)$ and $\phi(se)$ influence and motivate the choice of proposed parameters' values. While the interpretation of the dummy player axiom and the symmetry axiom are straightforward, it is worth spending a few words on the efficiency axiom. It states that the payout to be distributed is the difference between the desirability of the proposal $\hat{cb}(\boldsymbol{\theta})$ and expected desirability or utility in space $\mathbb{E}[\hat{cb}(\boldsymbol{\Theta})]$. This resembles the idea of the AF comparing different candidate points globally in the configuration space and finally picking the best one.

The SV has also other advantages compared to the HPI methods presented in section 3[9]. The major drawback of LPI was that parameters have contributions in $\mathbb{R}_0^+$. The SV does not have this limitation because contributions are in $\mathbb{R}$, which is more appropriate to explain the EETO. The biggest problem with the ablation analysis was the poor consideration of interactions. Notice how similar AA and SV are: both methods compute importance relative to a reference value. Yet the latter method better incorporates interactions by including all possible coalitions in the computation. To see why this is advantageous, take again the data generating model of section 3 but this time the SV is used to explain the target instance $(0, 0, 0)^T$.

[9]To the best of our knowledge the SV is the only competing method that fulfills the Linearity property. In [4, 11, 18] we have not found any explicit declaration for a linearity property. We do not exclude that some methods fulfills it, but at least it is was not as clear as for the SV.

We expect θ_2 and θ_3 to have same contributions and smaller than θ_1 because of their interaction. The results are displayed in the last column of table 2. While both methods seem to behave similarly with the non-interacting parameter θ_1 it is quite clear that the SV distributes the payout more equally among interacting and similarly important parameters θ_2 and θ_3 leading to more realistic and trustworthy results[10].

After motivating the choice of the SV the remainder of this section is dedicated to the introduction of `ShapleyMBO`, whose main steps are summarized in algorithm 3. Note that the method is built upon the {`mlrMBO`} package and can be therefore used only in combination with it. After the optimization is terminated, users can choose which iteration(s) should be analyzed. For the estimation of the SV, a sampling population is required, which provides the basis for the Monte Carlo sampling (see algorithm 2 line 2) and the global average prediction. More importantly, the **global** search of the AF should be simulated. Hence $1000 \cdot p$ points, with p being the dimension of the HP space, are sampled at random with Latin Hypercube Sampling from the configuration space. Note that this sampling method is computationally cheap and independent of the infill optimizer. To compute $\hat{\boldsymbol{\phi}}(m)$ and $\hat{\boldsymbol{\phi}}(se)$ the {`iml`} package is used [27]. While the estimation of the former is straightforward using the SM $\hat{f}_m$, the estimation of the latter is easily solved with a customized function $\hat{f}_{se}$ that predicts the uncertainty of the model. Finally $\hat{\boldsymbol{\phi}}(cb)$ is constructed using the linearity axiom (see Appendix B for more details). Thereby it is important that both objects, $\hat{\boldsymbol{\phi}}(m)$ and $\hat{\boldsymbol{\phi}}(se)$, differ only in the prediction function (same sampling population and Monte Carlo samples), otherwise the resulting $\hat{\boldsymbol{\phi}}(cb)$ would be wrong. After `ShapleyMBO` has run, another method called `plotShapleyMBO` facilitates the visualization of the results. Users can choose between individual iterations or so-called *desirability paths*, which display how contributions evolve throughout the optimization process. With the help of confidence intervals the estimation uncertainty can also be plotted. Since $\hat{\phi}_j \overset{a}{\sim} \mathcal{N}(\phi_j, \frac{\sigma_j^2}{K})$ the $1-\alpha$ confidence interval is $CI_{1-\alpha} = [\hat{\phi}_j \pm t_{(1-\frac{\alpha}{2}, K-1)} \frac{\hat{\sigma}_j}{\sqrt{K}}]$, where the quantiles of the t-Distribution with $K-1$ degrees of freedom are taken since σ_j^2 is unknown.

[10] Note that the exact SV is $(-\frac{1}{2}, -\frac{1}{8}, -\frac{1}{8})^T$ leading to a relative importance of $(\frac{2}{3}, \frac{1}{6}, \frac{1}{6})^T$, where θ_2 and θ_3 would get the exact same importance.

6 Experimental setup

In this section the experimental setting for the analysis is presented, in 6.1 for the synthetic and in 6.2 for real data application example. In addition, the aim and the hypothesis will be stated. All analyses are executed on a local computer (2,2 GHz Intel Core i7 quad-core CPU, 16GB RAM) with R software version 3.6.3 [31] in combination with the {`mlrMBO`} package version 1.1.4 [6]. For the SMBO hyperparameters, where possible, default values are used. In the following table common choices for the SMBO setting are displayed.

hyperparameter	value
min objective function	TRUE
noisy objective	TRUE
initial design	size $4p$ sampled with maximin LHS
surrogate model	GP regression
kernel function	$\frac{3}{2}$-Matérn
acquisition function	LCB
min acquisition function	TRUE
infill optimizer	focussearch ($n_r = 3, n_i = 5, n_p = 1000$)
termination condition	max. evaluations $20p$

Table 3: Common SMBO set up for the analysis. For focussearch subscripts r, i, p stand for *restart*, *iters*, *points*. Termination condition is set according to [30, p.614].

6.1 Application on test function

This analysis aims to explore if `ShapleyMBO` is a valid method to explain the choices of the LCB for different λ parameters. The method is tested with a synthetic function and results are finally compared to our expectations. Using a synthetic function is essential to validate the method. First of all, it allows to formulate concrete expectations for the contributions, in particular its functional form for the mean and the domain of the parameters for the uncertainty. Second, together with the algorithm's optimization path, it allows to better understand and justify the results. As objective function the $4p$ Hyper-Ellipsoid $f : \mathbb{R}^4 \to \mathbb{R}_0^+$ provided by the {`smoof`} package [7] is used

$$f(\boldsymbol{\theta}) = \sum_{j=1}^{4} j \cdot \theta_j^2 \ , \ \theta_j \in [-5.12, 5.12] \text{ for } j = 1, \ldots, 4$$

$$\boldsymbol{\theta}^* = (0, 0, 0, 0)^T \text{ and } f(\boldsymbol{\theta}^*) = 0.$$

To make the optimization problem more realistic *i.i.d* Gaussian noise with zero-mean and standard deviation set as 5% of the estimated standard deviation of the objective function is artificially added [30, p.613] (see figure 2). The baseline LCB control parameter is $\lambda = 1$, which is the default {`mlrMBO`} value for this optimization problem. To control for the stochastic behavior of SMBO 30 optimization problems are run. After each optimization problem is over `ShapleyMBO` is applied using sample size $K = 1000$. Results in each iteration are then averaged overall runs. Finally, the same analysis is repeated with $\lambda = 10$ to see how `ShapleyMBO` reacts to a more explorative algorithm[11].

Results obviously depend on the feature values proposed by the LCB. As regarding the *mean* we expect the contribution of a parameter to depend on the distance to its optimal value: the closer a parameter is to its optimal value θ_j^* the bigger its mean contribution. Further, according to the functional form of the Hyper-Ellipsoid replacing higher parameters causes bigger changes of the target. Hence, in general, we expect higher parameters to be more important[12]. Intuitively for the *uncertainty* the more a specific parameter value is isolated (higher model uncertainty) the better its uncertainty contribution should be. What is more, its contribution might be also influenced by how much the dimension has been explored previously. If a parameter dimension is explored less the contribution of an uncertain parameter value should be lower compared c.p. to a better explored dimension since in the former case proposing other values is likely to reduce similarly uncertainty in space. In other words, the less a dimension is explored, the less special an uncertain parameter value is. That said, assuming similar parameter values and explored dimensions c.p. we expect similar uncertainty contributions because parameters have the same domain. It is hard to guess which contribution is dominant for the overall desirability of a parameter and how contributions evolve because algorithm trajectories are not exactly predictable. Yet, given the rather simple test function, we expect the algorithm to converge towards the optimum smoothly after initial exploration rounds. If so, the mean should become relatively more important during convergence than the uncertainty, as was also captured in the figure 1. Hence, we expect parameters' contributions on the LCB to be initially dominated by the uncertainty and then by the mean. Finally when increasing exploration with $\lambda = 10$ the mean

[11] As shown in section 7.1 with $\lambda = 1$ the algorithm does not really explore the space and hence we need to increase λ to test how `ShapleyMBO` reacts in different regions of the domain.

[12] Take for instance $\boldsymbol{\theta} = (0.05, 0.15, 0.25, 0.35)^T$. By individually replacing these values in the noise-free function e.g. with 2.56 (average distance to the optimal value 0), we can see that replacing θ_4 causes the biggest objective change although this parameter is the most distant from the optimum $(f(\boldsymbol{\theta}) - f(0.05, 0.15, 0.25, 2.56) = -25.72)$.

contribution should not show many differences, whereas we expect the uncertainty to influence the results more heavily than with $\lambda = 1$. The following table sums up our expectations in the Hyper-Ellipsoid analysis.

	expectations
$\phi(m), \lambda = 1$	• depends on distance to optimum: the closer, the bigger • for similar values higher parameters are more important
$\phi(se), \lambda = 1$	• depends on uncertainty value and dimension exploration: the more isolated and explored, the bigger • for similar values and equally explored dimensions, parameters are similarly important
$\phi(cb), \lambda = 1$	• initially dominated by uncertainty, then by mean
$\phi(m), \lambda = 10$	• like $\lambda = 1$
$\phi(se), \lambda = 10$	• like $\lambda = 1$ but with stronger effect
$\phi(cb), \lambda = 10$	• same as $\lambda = 1$ with pronounced uncertainty effect

Table 4: Expectations for the results of the Hyper-Ellipsoid analysis. Expectations on $\phi(cb)$ should be used carefully as the results strongly depend on the algorithm trajectories.

6.2 Application on real data

The goal of this analysis is to provide an application example of `ShapleyMBO` on a real data set to show how our framework can be used to explain tuning results. For that, seven HP of a multilayer perceptron (MLP), summarized in table 5, for a speech recognition classification task are tuned with one single SMBO run[13]. Being a real black-box application example without comparable literature we do not have any specific expectations regarding the results. Instead of evaluating each proposed configuration, the validation performance is predicted with a random forest (EPM as an objective function), which was trained on data taken from LCBench[14]. The LCB is used with $\lambda = 1$ and after the optimization problem is over results are explained with `ShapleyMBO` using sample size $K = 20000$. The sample size is found with `checkSampleSize`, a new method proposed to find a sufficiently high sample size for individual explanations. In this analysis, the "optimal" sample size is computed once for the proposed configuration with the best-predicted value of the process (using

[13] Additional information regarding the task can be found on OpenML `https://www.openml.org/d/1489`

[14] More precisely the EPM predicts the accuracy on the validation set but the metric being optimized (here minimized) is the validation error. Additional information regarding LCBench can be found at `https://github.com/automl/LCBench`.

the final surrogate model) and then applied to all other proposals for consistency. Further information on `checkSampleSize` are found in Appendix C.

HP	Notation	Type	Lower	Upper	Trafo
batch size	bs	numeric	$\log_2 16$	$\log_2 512$	2^x
max dropout	md	numeric	0	1	
max units	mu	numeric	$\log_2 64$	$\log_2 1024$	2^x
number of layers	nl	integer	1	5	
learning rate	lr	numeric	0	0.01	
momentum	mom	numeric	0.1	1	
weight decay	wd	numeric	0	0.1	

Table 5: Parameter set of the MLP application example. Parameter *max unit* indicates the maximum number of units per layer and *max dropout* is the dropout rate.

7 Results

In the following sections, the results of the analysis are presented. Since the AF is minimized in both analyses, the LCB of the proposed points will be lower than the average LCB in space and the payout to distribute negative. The lower the acquisition of a proposal the higher its desirability (the bigger its utility). Consequently, the SV of a parameter will be negative in general and, the smaller the bigger the contribution. To avoid confusion with the interpretation, when the payout and the SV of a parameter is smaller than zero we say that the payout and the contribution is positive. If the SV decreases over time (goes to $-\infty$) we say that the contribution increases. The same logic applies to mean and uncertainty contributions.

7.1 Hyper Ellipsoid

The analysis' results are displayed in figure 5 and 6. We will start by explaining the former and then pass on the latter figure. In figure 5 the results in iteration 59 are visualized on top for $\lambda = 1$ (i and ii) and at the bottom for lambda $\lambda = 10$ (iii and iv). Additional information is provided in tables 6, 7 and 8. By focusing on $\lambda = 1$ first, notice how every parameter contributes positively on the LCB but higher pars more than lower (plot ii). For instance, replacing the value of θ_1 is expected to change the utility of the proposed configuration by only 5.35 c.p., decreasing its desirability only a little compared to a change of e.g θ_4 (32.56). The plot on the left shows how the desirability of each parameter is made of. As the proposed configuration is

close to its optimum, every parameter has a positive mean contribution, but higher parameters are more influential as expected. Take for instance θ_4, which is not as close to zero as θ_3, but replacing it would cause the biggest change in the mean prediction: replacing it is expected to increase c.p. the predicted target value on average by 36.58.

		θ_1	θ_2	θ_3	θ_4
$\lambda = 1$	$\hat{\boldsymbol{\phi}}(cb)$	-5.35 (2.12)	-12.14 (3.33)	-21.12 (4.11)	-32.56 (4.44)
	$\hat{\boldsymbol{\phi}}(m)$	-7.20 (2.4)	-14.64 (3.83)	-24.63 (4.46)	-36.58 (4.83)
	$\hat{\boldsymbol{\phi}}(se)$	1.85 (0.42)	2.50 (0.66)	3.51 (0.47)	4.02 (0.67)
$\lambda = 10$	$\hat{\boldsymbol{\phi}}(cb)$	-3.82 (0.7)	-8.14 (0.65)	-12.32 (1.26)	-18.77 (1.5)
	$\hat{\boldsymbol{\phi}}(m)$	-8.09 (0.71)	-16.22 (0.75)	-25.31 (0.84)	-36.99 (1.09)
	$\hat{\boldsymbol{\phi}}(se)$	4.27 (0.81)	8.08 (0.74)	12.99 (1.37)	18.22 (1.64)

Table 6: Contributions in iteration 59 for both λ. Results for each are averaged over the 30 BO runs. Negative values are associated with positive contributions and vice versa. Standard deviation of the results in brackets.

Contrary to the mean, the uncertainty has a low and negative influence. By looking at table 7 evidently its influence is weak because the payout $\mathcal{P}(se)$ is relatively small. Also, the uncertainty $\hat{se}$ of the proposed configuration is lower than the average $\bar{se}$ and therefore the payout is greater than zero (negative contributions). As a consequence, the payout of the LCB is smaller than the mean payout in absolute terms. In other words, uncertainty contributions push down the mean effect, making parameters appear less desirable overall. Bad influences are somehow counter-intuitive, but recall that the algorithm also desires to explore unknown regions in space. With that in mind, proposing a configuration with uncertainty **prediction lower than the average** can be interpreted as an exploration **sacrifice** (in favor of mean reduction). Obviously, the sacrifice can be further divided among the parameters such that worse uncertainty contributions indicate bigger exploration sacrifice in the respective dimensions. In the case of $\lambda = 1$ the sacrifice seems similar for all configurations and replacing any of those values would increase the desirability of the proposal by roughly the same amount (see $\hat{\boldsymbol{\phi}}(se)$ in table 6). The reason for similar contributions might be, as mentioned in section 6.1, that all dimensions have been equally explored up to that point and apparently this is the case, as shown in table 8[15].

[15]To measure the exploration of dimension j the standard deviation of the average distance of the configuration value from its optimal value is computed. Formally, given the average distance in iteration t $d_j^{(t)} = \frac{1}{R}\sum_r |\tilde{\theta}_j^{(r)} - 0|$ where r stands for run and $R = 30$, the exploration of dimension j is $\sqrt{\frac{1}{T-1}\sum_t (d_j^{(t)} - \bar{d}_j)^2}$, with $T = 58 + 16$ if initial design is included and 58 otherwise.

	y	$\hat{cb}$	$\bar{cb}$	$\mathcal{P}(cb)$	$\hat{m}$	$\bar{m}$	$\mathcal{P}(m)$	$\hat{se}$	$\bar{se}$	$\mathcal{P}(se)$
$\lambda = 1$	0.34	-1.41	71.13	-72.54	0.73	85.38	-84.64	2.14	14.25	12.11
$\lambda = 10$	0.66	-19.83	24.57	-44.40	0.72	89.47	-88.75	2.06	6.49	44.3

Table 7: Predictions (actual and average), payout $\mathcal{P}$ and target value y of explicand in iter 59 for both λ. Values are averaged over 30 BO runs. The payout is the difference between prediction and average prediction (scaled with $-\lambda$ for the se).

	with id	θ_1	θ_2	θ_3	θ_4
$\lambda = 1$	yes	0.95	0.99	1.01	0.98
$[1 - 58]$	no	0.24	0.15	0.08	0.07
$\lambda = 10$	yes	1.29	1.26	1.15	1.07
$[1 - 58]$	no	1.40	1.31	1.11	0.97

Table 8: Exploration of each dimension up to iteration 58 (iteration 1 to 58 are included). Column *with id* indicates if the initial design was included or not. The exploration is measured as the standard deviation of the average distance of the configurations from its optimal value $\theta_j^* = 0$.

To see how contributions for a similar configuration[16] change when increasing λ the focus is moved to plots iii and iv in figure 5. Also here, higher parameters are more appealing but this time contributions are smaller. Plot iii on the left again indicates that only the mean contribution positively influences parameters' desirability. It is interesting to see how similar mean contributions are to $\lambda = 1$. Even more impressive is the drastic change in the uncertainty contributions, which are even worse now. Apparently, when exploration is more important the "same" configuration causes a bigger sacrifice, that pushes down even stronger cb contributions. To understand more about the relationship between λ and the exploration sacrifice consider the LCB payout, which is defined as

$$\begin{aligned} \mathcal{P}(cb) &= \hat{cb} - \bar{cb} \\ &= \hat{m} - \lambda \hat{se} - \bar{m} + \lambda \bar{se} \\ &= (\hat{m} - \bar{m}) + \lambda(\bar{se} - \hat{se}), \end{aligned} \tag{9}$$

where the left part is the mean payout $\mathcal{P}(m)$ and the right part, included λ, is the uncertainty payout $\mathcal{P}(se)$. In equation 9 notice how, unlike LPI (see section 3), potentially different payouts for mean and uncertainty allow contrasting contributions

[16] This iteration was chosen because the proposals were the most similar ones according to the euclidean distance. The goal was to assess how contributions for similar proposals but different λ differ. Although parameter values differ, both mean $\hat{m}$ and uncertainty $\hat{se}$ predictions are very similar (table 7). Average uncertainty $\bar{se}$ is lower for $\lambda = 10$ because, as predictable, space is explored more.

within the same parameter. Recall that the LCB is minimized and therefore if $\mathcal{P}(cb)$ increases (goes to $-\infty$) more payout is distributed and contributions will be higher. Now it is evident that the effect of an exploration sacrifice ($\hat{se} < \bar{se}$) is c.p. more drastic when λ increases.[17] By decreasing λ, i.e. to $\lambda = 1$, the $\mathcal{P}(cb)$ of the same explicand would improve from -44.40 to -84.32 and get closer to -72.54. Speaking in terms of cooperative game theory we could say, that a higher λ decreases c.p. the worth of the grand coalition (of the proposed parameter values). Going back to the uncertainty contributions in plot iii, it looks like for higher parameters the sacrifice is higher. By looking again at table 8 indeed higher dimension have been explored less up to that point (for instance the standard deviation of θ_1 is 1.2 times higher the one of θ_4).

The configurations just explained were in proximity of the optimum (see also y in table 8). Recall though that contributions depend on parameter values. To assess how results change for different configurations in the following paragraphs, the *desirability paths* will be analyzed, which are displayed in the figure 6. To further investigate the results figures 3, 4, 7, 8 and table 9 are used. For $\lambda = 1$ the paths show a very homogeneous picture: contributions are almost constant and higher parameters are more desirable throughout the entire process. Looking at the decomposition below, also mean and uncertainty contributions are constant and the former clearly dominates while the latter has a little and negative effect on the LCB. These results do not show substantial differences from the results explained previously. In fact, the averaged contributions over the entire process shown in table 9 are close to the ones in iteration 59. The influence of the parameters is almost constant because both mean and se prediction, and therefore also the LCB, rapidly shrink (in less than ten iterations) and since then no longer change (see figure 3). This in turn is determined by the proposed configurations. As seen in the left visualization in figure 4 plot ii, the algorithm quickly spots the optimal region in the input space without leaving it for the rest of the process. There is little exploration, especially for θ_1 in the beginning, but this is barely noticeable compared to the impressive cut between the initial design and the proposals. Indeed, the exploration (until iteration 58) for $\lambda = 1$ drops substantially, particularly the one of θ_4, when the initial design is not included (compare the first two lines in table 8). In conclusion, paths for $\lambda = 1$ are constant because the LCB chose very similar instances in each iteration.

[17] Note that this equation explains in general how $\mathcal{P}(cb)$ is affected by m, se and λ . For instance *exploitation* sacrifices are possible when $\hat{m} < \bar{m}$.

		θ_1	θ_2	θ_3	θ_4
$\lambda = 1$	$\hat{\phi}(cb)$	-5.29 (2.22)	-11.97 (3.57)	-21.6 (4.41)	-32.59 (4.41)
$[1-64]$	$\hat{\phi}(m)$	-7.10 (2.53)	-14.44 (4.09)	-24.66 (4.54)	-36.57 (4.81)
	$\hat{\phi}(se)$	1.81 (0.48)	2.47 (0.68)	3.50 (0.52)	3.98 (0.70)
$\lambda = 10$	$\hat{\phi}(cb)$	-33.04 (16.06)	-30.35 (14.83)	-37.68 (16.62)	-38.88(14.21)
$[1-10]$	$\hat{\phi}(m)$	8.43 (9.06)	10.97 (15.62)	9.05 (27.14)	2.42 (34.65)
	$\hat{\phi}(se)$	-41.47 (21.48)	-41.33 (26.12)	46.73 (37.29)	41.3 (39.38)
$\lambda = 10$	$\hat{\phi}(cb)$	-3.81(0.77)	-8.12(0.8)	-12.31(1.31)	-19.01(1.54)
$[55-64]$	$\hat{\phi}(m)$	-7.79 (1.16)	-16.12 (1.01)	-25.09 (1.47)	-39.96 (1.27)
	$\hat{\phi}(se)$	3.98 (1.06)	8.00 (0.89)	12.78 (1.64)	17.94 (1.76)

Table 9: Contributions averaged over multiple iterations in the process. Like in table 6 negative values are associated with positive contributions and vice versa. Standard deviation of the results in brackets. Range in brackets below λ indicates the iterations included.

Contrary to $\lambda = 1$, with $\lambda = 10$ the algorithm explores the configuration space much better (plot ii on the right in figure 4) and results are more interesting. Also in this case all parameters influence the LCB positively in the entire process, but relative to $\lambda = 1$ two major differences can be identified. First, contributions are initially higher and parameters' desirability is more equal. Second, contributions smoothly decrease over time and, similar to $\lambda = 1$, differences between parameters appear. Yet, other than $\lambda = 1$, higher parameters become less desirable overall [18]. Let's start with the former difference. As shown from the decomposition plots and table 9, in the beginning, the desirability of the proposals is dominated by the uncertainty and not by the mean. Here the other sacrifice shows up: the LCB chooses configurations primarily to reduce uncertainty in the space giving up exploitation (proposing parameter values with bad mean contribution). Because parameters have similar se contributions, which are much stronger than m contributions, parameters' desirability $\hat{\phi}(cb)$ is more uniform and higher compared to $\lambda = 1$ (compare first two line blocks in table 9). It is also worth mentioning how uncertainty contributions of θ_1 and θ_2, unlike those θ_3 and θ_4, even increase in the initial iterations. By looking at figure 7 it can be seen that 9 out 10 proposals for θ_1 are on the edge of the parameter's domain, even more distant than the initial design, which indicates that this dimension is being heavily explored. The negative trend shown in the figure indicates that exploration is generally higher for lower dimensions, which might further explain why the exploitation sacrifice is higher for θ_1 (8.43) than for θ_4 (2.42). Results in the initial iterations should yet be interpreted carefully because

[18] Notice how the gap between the constant black line and the grey curve in figure 6 is bigger for higher parameters.

the standard deviation, of θ_3 and θ_4 in particular, is quite high. To understand why, take a look at the histograms in figure 8, which shows the distribution of all, not the averaged, proposed configurations in the first ten iterations for $\lambda = 10$. While for θ_1 more than 80% of the values lie on the boundary of the domain, for θ_4 only half as many values (ca. 40%) are as far and the rest is symmetrically distributed around zero. As if the algorithm would not invest resources in exploring dimensions that significantly impact the objective function, even when forced to by λ. As a consequence, a mixture of different parameter values is proposed causing high uncertainty in the results.

Despite the initial exploration the algorithm begins to converge. Mean and Uncertainty prediction of the proposals, and consequently also the LCB, decreases and something remarkable happens: the mean and uncertainty contributions curves cross. The positive influence of the uncertainty vanishes (decreasing *se* contribution) and the mean takes the lead of the parameter's desirability (increasing m contribution). The curves do not cross simultaneously for all parameters and for higher ones it happens earlier in the process: for θ_4 already after 10 iterations, while for θ_1 after approximately 25 iterations. By focusing on plot i in figure 4 it can be seen that θ_4 has a steeper convergence curve than θ_1. Therefore the dimension of the former parameter is explored faster, or equivalently exploited longer, and consequently curves cross sooner[19]. As the process goes on contributions slowly stabilize and by the end of the process similarities with the $\lambda = 1$ appear (compare first and final line blocks table 9). While the mean contribution of all parameters clearly converges, the uncertainty contributions of $\lambda = 10$ exceed the ones of $\lambda = 1$, and as already explained in iteration 59, this gap is bigger for higher parameters. The mean contributions nevertheless dominate and therefore *cb* contributions are still positive but, because of the higher sacrifice, these are lower than $\lambda = 1$.

The analysis of the Hyper-Ellipsoid showed many insights and here is a summary of the main results. We started by explaining a configuration in proximity of the optimum. For both lambda, parameters positively impacted the LCB, but contributions were smaller for $\lambda = 10$. By filtering the overall contribution we noticed that parameters were desirable only because of their mean effect and mean contributions were almost equal for the two λ. We further discovered that negative contributions are not unrealistic at all, indeed uncertainty had a negative influence on the desirability of the parameters and this could be interpreted as an exploration sacrifice. While for $\lambda = 1$ this sacrifice is negligible for $\lambda = 10$ the the effect is

[19] Also, in plot i of figure 4, notice how the start of the θ_1 path is further away than the one of θ_4 (average distance of almost 5 compared to lower than 4).

more severe, in particular for less explored dimensions. Because results depend on parameter values we further analyzed desirability paths to assess how contributions evolved throughout the algorithm's convergence. With $\lambda = 1$ the weight of the uncertainty is too low to influence the choice LCB and almost immediately the algorithm jumps to the optimal region in the configuration space, which was rather unexpected. As consequence paths were not much different than results in iteration 59. Contrary to that, paths for $\lambda = 10$ were much more informative. The algorithm converged slower and explored the configuration space better, starting from the domain's boundaries and finally ending in proximity of the optimum. This allowed us to discover valuable insights on the behavior of the AF. Initially, the LCB chooses parameter values to reduce uncertainty in space (positive *se* contributions), occasionally even sacrificing exploitation. Yet during the process, reducing uncertainty becomes less appealing than mean reduction, and the LCB begins to prefer well-performing configurations over isolated, uncertain configurations. As a consequence mean contributions increase and uncertainty contributions decrease. Sacrificing exploration seems inevitable as the algorithm aims at finding the optimum of the target function. The moment when mean reduction becomes more important than uncertainty reduction is indicated by the contributions curves crossing. In general this happens earlier for higher dimensions because they are more important for the optimization. Taking back into consideration table 4 we conclude, that the results provided by `ShapleyMBO` were in line with our expectations, especially for $\lambda = 10$. Hence, we consider the Shapley value a suitable method to explain the choices of the LCB and in the next section we will show how to benefit from it in a real tuning example.

7.2 MLP

After the optimization has terminated, users can conduct two possible analyses: explain single proposals or investigate the desirability paths. There is no specific order to follow and we will simply start with the former analysis. An interesting instance to explain would be the *best-predicted* proposal, namely the proposal with the best mean prediction according to the final surrogate model (see section 4). The best-predicted proposal was the one in iteration 95[20]. In figure 9 its target value and predictions are highlighted within the algorithm path with a red circle. To compute the Shapley Value different sample sizes $\{100, 1000, 5000, 10000, 15000, 20000\}$ were tested with `checkSampleSize` and only the last one was sufficient for cb, m and

[20]This configuration also has the best target value and best prediction among all visited points in the optimization problem, initial design included.

se. Although the sample size is relatively large the computation was still inexpensive (elapsed time of 243 seconds). Contribution results are displayed in figure 10. Although all parameters have a positive contribution on the LCB there are clear desirability differences between them. Batch size and number of layers are the most desirable, followed by max units and learning rate. The remaining parameters have a rather weak contribution. By looking at the decomposition plot it is evident that the desirability of the parameters is entirely made out of their mean contribution. In other words, this configuration was chosen because it is expected to perform well (low validation error). For instance replacing the batch size or the number of layers is expected to increase c.p. the predicted validation error by approximately 0.015. These parameters are quite important and together they account for more than 50% of the mean payout $\mathcal{P}(m)$. While max units and learning rate still have a moderate influence on the mean prediction, the contributions of momentum and max dropout are barely perceptible (see the first line in table 11). Unlike the mean, the uncertainty has little and negative influence (exploration sacrifice) on the desirability of the proposed configuration. For all parameters except one, the effect is negligible. Consequences for the learning rate are more dramatic. The sacrifice for this parameter is so high, that the learning rate appears less desirable than max units although it has a higher mean contribution. The higher sacrifice could indicate the its dimension has not been explored enough up to iteration 95.

	bs	*nl*	*mu*	*lr*	*wd*	*md*	*mom*
$\hat{\boldsymbol{\phi}}(cb)$	-0.014	-0.013	-0.008	-0.007	-0.004	-0.001	-0.001
$\hat{\boldsymbol{\phi}}(m)$	-0.016	-0.015	-0.009	-0.010	-0.005	-0.002	-0.001
$\hat{\boldsymbol{\phi}}(se)$	0.002	0.002	0.001	0.003	0.001	0.000	0.000

Table 10: Contributions of the parameters in iteration 95 for the MLP optimization problem. Actual and average predictions for cb, m and se can be found in figure 10. Payout $\mathcal{P}$ for cb, m and se are respectively $-0.048, -0.058, 0.09$.

Before further investigating the results with the desirability paths, in this paragraph we briefly open a parenthesis to demonstrate again why the Shapley value turns out to be a better choice than ablation analysis. To show that, the mean contributions resulting from an AA are compared to those in table 10. The target configuration is again the best-predicted proposal, as prediction model the surrogate model of iteration 95 is used and finally to have similar payouts, a configuration whose prediction is similar to the SV average prediction is used as source[21]. As can seen be in table 11

[21]To find the source, 7000 instances are sampled at random from space, their prediction is computed using the surrogate model and then the configuration with lowest prediction difference to the SV average prediction in iteration 95 is taken.

there are substantial differences between the methods. Relative to the SV with the AA the contributions of batch size and number of layers are stronger. In addition, weight decay becomes surprisingly more influential. More impressive though, is how little importance is attributed to the learning rate and max units. This anomaly could be caused by the presence of interactions in the surrogate model between the mentioned parameters. To test if this is the case, the interaction strength of the learning rate and max units with the remaining parameters is computed using the H-Statistic proposed by Friedman and Popescu. In a nutshell, the H-Statistic measures how much of the variation of the partial dependence between two features is captured by their interaction effect [14, p.934][22]. As shown in figure 11 plot ii and iii, there is moderate interaction between the learning rate and batch size as well as between max units and number of layers. Additionally, both parameters seem to interact with weight decay. Hence, it might be that both the learning rate and the max units result less important in the AA because their contribution is partly absorbed by the other mentioned parameters, which in turn appear more influential. This is not the case when using the Shapley value, where the payout is distributed more uniformly between the involved parameters (see also figure 11 plot i).

	bs	*nl*	*lr*	*mu*	*wd*	*md*	*mom*
relative SV	0.278	0.257	0.166	0.161	0.089	0.027	0.022
relative AA	0.348	0.307	0.076	0.050	0.128	0.044	0.047

Table 11: Relative m contribution of the parameters for configuration proposed in iteration 95. Relative contribution is the actual contribution divided by the payout.

In the initial paragraph of this section the best-predicted proposal was explained. To see how the contributions of the parameters evolved, desirability paths, displayed in figure 12, shall be analyzed. Because results in each iteration can be computed independently, the computation was run on parallel to keep execution time within reasonable ranges[23]. For most of the optimization problem (approximately initial 85 iterations) contributions are rather stable and do not change much. Batch size and number of layers are the most desirable parameters whereas weight decay, max dropout, and momentum are barely appealing for the LCB. A first difference can be spotted in the learning rate, which seems to be more influential than max units within this time frame. By looking at decomposition plots beneath it becomes clear, that the mean contribution dominates the desirability of the parameters and the uncertainty has little and negative influence. For the learning rate instead, the un-

[22] Refer to equation 44 in the paper for more details. The estimation is done with the `Interaction` method from the {`iml`} package using a grid size of 100.

[23] With the faster version of `ShapleyMBO` computation took approximately two hours using 4 cores.

certainty contribution has a bigger impact. The exploration sacrifice is so big, that the parameter becomes overall less desirable than number of layers, although they have almost identical mean contributions up to that point. Contributions in the optimization problem are not entirely static and in the final iterations of the process (ca. after iteration 85) something remarkable happens. The mean contribution and consequently also the desirability of the learning rate converge to zero. Simultaneously the exploration sacrifice of the same parameter vanishes. Recall that in section 7.1 we hypothesized that, the less one dimension is being explored the bigger the exploration sacrifice would result. If so, this path anomaly might be caused by the sudden decision of the algorithm to explore a bad performing region of the learning rate dimension. The suspect is confirmed in figure 13, in which a final lr exploration rush is clearly visible. Let us dive a little deeper into the reasons behind those exploration rounds. Take for instance the proposal in iteration 108, whose results are compared to the one of proposal 95 in table 12 and figure 14. As seen in the right plot of the figure, the chosen learning rate is, contrary to other parameters and the value in iteration 95, quite distant from the average (from the grey distribution mass). Further, notice how similar the two highlighted configurations are except for the learning rate value, which might have actually caused the increase in the uncertainty of the proposal 108 (see also figure 9 green circle). Finally, note how both proposals have fairly good mean predictions $\hat{m}$ and optimal validation errors and yet their contributions strongly differ. On the one hand, in iteration 108 the learning rate is barely desirable because its influence on the mean prediction is almost invisible (the parameter accounts for only 2% of the mean payout). On the other hand, the exploration sacrifice is substantially lower than in iteration 95 (the parameter accounts for only 6% of uncertainty payout). This indicates that the LCB has proposed values in iteration 95 and 108 for different reasons: the former to improve the validation error and the latter to reduce uncertainty, or increase knowledge, in the lr dimension. As if the algorithm decided to spend the final resources exploring an unknown region of the learning rate. This is arguably a precious insight to avoid false conclusions. For instance, by looking only at the validation error in iteration 108, one could erroneously deduce, that the learning rate exploration might have been beneficial for the optimization. Looking at its mean contribution instead, one should not attribute to the learning rate many credits for the achieved performance.

Before proceeding with the next section, here are the main takeaways from this chapter. In this section a real tuning example was conducted and the results were explained to provide a general workflow example of `ShapleyMBO`. Apparently the mean was the only fundamental contribution to the LCB and only a subset of the parameters was actually determinant. Of particular interest was the learning rate

iter	lr	$\bar{lr}$	error	$\hat{m}$	$\hat{se}$	$\hat{\phi}_{lr}^{rel}(cb)$	$\hat{\phi}_{lr}^{rel}(m)$	$\hat{\phi}_{lr}^{rel}(se)$
95	0.0079	0.007	0.2222	0.2226 (3.6)	0.0003 (8.0)	0.15	0.17	0.27
108	0.0041	0.007	0.2225	0.2229 (12.5)	0.0011 (66.1)	0.01	0.02	0.06

Table 12: Comparison of the learning rate proposals in iteration 95 and 108 in the MLP optimization. From left to right: proposed value (original scale), average value up to inclusive iteration 108, target value, *mean* and *se* prediction, and relative contributions of the learning rate (SV divided by the payout). Target values achieved in these iterations are the best among all visited configurations in the process. The number in brackets is the percentile of the prediction according to the empirical distribution functions of the proposals up to iteration 108.

path, which showed a sudden desirability decrease associated with the exploration a non-promising region. In conclusion, also in this application example, `ShapleyMBO` has proven to be a valid diagnostic tool to investigate the configurations proposed during the optimization.

8 Discussion of the method

This section aims to briefly discuss selected aspects of `ShapleyMBO`, which could be used as a starting point for further developments of the framework. The first and more important topic to discuss is the sampling strategy adopted to explain the choices of the AF and approximate the SV. We intended to approximate the **global** search of the LCB for the best candidate in the configuration space. Essentially the sampling strategy should be (i) independent of the infill optimizer, (ii) computationally inexpensive, (iii) reproducible in each iteration, and more importantly (iv) approximate the parameter space globally. The last point has a pivotal role in the analysis results because the sampled configurations form the average prediction in the SV estimation. On the one hand, covering the entire input space allows for stable and regular payout paths since, as shown in figure 15 plot i, the average mean prediction remains constant and the average uncertainty prediction is expected to decrease throughout the process smoothly. On the other hand, already since the beginning of the optimization, payouts and contributions will be in favour of mean and against uncertainty. More precisely, both mean and uncertainty prediction will be typically much lower than the global average leading to positive m and negative *se* contributions. Hence, as was shown by the results of both analyses, parameters will typically appear desirable only because of their mean contribution. While these results are reasonable during exploitation, they are harder to justify during exploration because we would expect positive uncertainty contributions. In other words, with global sampling mean contributions are over- and more importantly uncertainty

contributions are underestimated. For instance, in the case of the Hyper-Ellipsoid even with $\lambda = 10$ results soon show an exploration sacrifice (figure 15 plot ii), or in the case of the MLP tuning example also for proposals with relatively high uncertainty and bad mean, like in iteration 61, most parameters have negative *se* contributions (see figure 9 and 12). This effect is almost inevitable when predictions are compared to the global average and in light of this weakness, users should pay particular attention to interpret uncertainty contributions correctly. Exploration is not dichotomous and should be considered more as a continuous quantity. An exploration sacrifice (negative *se* contribution) does not mean that a parameter's dimension has not been explored at all, but rather that it has not been explored *strongly enough* to increase or influence positively the desirability of the parameter. As such, users should wisely use the sacrifice as an exploration indicator: the lower the sacrifice, the more a dimension has been explored and vice versa.
To allow more sensitive and impartial results of mean and uncertainty contributions a local sampling strategy could be adopted. So instead of sampling globally, configurations might be sampled around the proposal in each iteration. This method could eventually solve the mentioned issue, but notice that it also hides a few disadvantages. First of all, because local configurations are expected to be similar, payouts would be much smaller and finally, parameters' contributions would be harder to distinguish and expensive to compute. Second, since local sampling causes heterogeneous sampling populations between iterations, average predictions might differ strongly, and, finally, desirability paths might become unstable. To take the best from both worlds, even a hybrid sampling strategy might be adopted, that would include both global and local configurations. Because the interpretation of the results changes with different sampling strategies, they should be considered complementary and not substitutes. Users could for instance investigate paths with a global and, when focused on specific configurations, repeat the computation with local sampling.

The second point to be discussed regards another possible extension of `ShapleyMBO` with an alternative way to compute the SV proposed by Sundararajan and Najmi, which the authors named Baseline Shapley value (BS) [37]. Its major difference to the traditional Conditional Expectation Shapley value (CES) is the contribution function. As the name also indicates the contribution of a parameter is computed relative to a baseline $\boldsymbol{\theta}'$. Formally given a prediction model $\hat{f}$ and an explicand $\tilde{\boldsymbol{\theta}}$, the contribution function for any subset $S \subseteq P$ is given by

$$v(S) = \hat{f}(\tilde{\boldsymbol{\theta}}_S; \boldsymbol{\theta}'_{P \setminus S}).$$

So instead of measuring the worth of S by marginalizing out the "missing" param-

eters and computing the expectation, these values are replaced with their baseline values. It follows that $v(\emptyset) = \hat{f}(\boldsymbol{\theta}')$. This apparently simple technique comes with an advantage. The computation of the SV does not require any, some authors say, unrealistic parameter independence assumption [1, p.2]. Still, the choice of the baseline is critical since an inappropriate one may create unrealistic instances [37, p.19]. In the specific case of SMBO for HPO reasonable choices exist though. As the source in the ablation analysis, one possible baseline might be the default HP configuration proposed by libraries. Another possibility would be to set an adaptive baseline using the incumbent configuration, namely the HP setting with the best target value so far. It is important to mention that in both cases the interpretation of the results changes. With the BS we can no longer explain the choice of the AF relative to the entire configuration space, but instead explain the desirability of the proposals relative to one specific configuration only. Like the local sampling strategy previously discussed, these results would furnish complementary benefit for the SMBO users. To the best of our knowledge, the BS is not yet implemented in any software packages, which sets an additional challenge for integrating this method.

Last but not least, this final paragraph is dedicated to practical implications of a parameter's mean contribution. Undoubtedly, the mean is the part of the AF that guarantees convergence towards an optimum[24] and parameters with a strong mean contribution can be considered the drivers of the optimization. One major benefit of decomposing the desirability of the proposal is that, with the mean contribution, users additionally get an inexpensive surrogate HP importance metric. Blind reliance on this HPI metric, without any deeper analysis, is not recommended yet because these contributions strongly depend on the quality surrogate model[25]. While it is reasonable to use the actual surrogate model to explain the choice of the AF in each iteration, contributions of the same instance might change when different models are used, as is the case for the proposal in iteration 95 (see table 13). Hence, it is important that users take mean contributions only as indicative, treat them with caution and support results with further analysis rather than make hasty conclusions.

[24] Any local and not only the global optimum. When dealing with multimodal target functions convergence to global optimum is supported by exploration, since it avoids the algorithm getting stuck in local optima.

[25] Surrogate models are trained with few observations and are not finely tuned. Hence, they might be unstable and generalize bad when used for prediction purposes.

	bs	*md*	*mu*	*nl*	*lr*	*mom*	*wd*
$\hat{\phi}(m)_{sm_{95}}$	-0.016	-0.002	-0.009	-0.015	-0.010	-0.001	-0.005
$\hat{\phi}(m)_{sm_{113}}$	-0.015	-0.002	-0.012	-0.016	-0.007	-0.001	-0.005

Table 13: Comparison of the mean contributions of the best-predicted proposal (iteration 95) in the MLP tuning example using the actual surrogate model (iteration 95) and the final surrogate model, which is fitted using all visited points in the optimization. For most parameters contribution seems unchanged. For *mu* and *lr* there are moderate changes.

9 Conclusion and outlook

SMBO is a powerful method for tuning, which suffers from bad explainability though. In particular, users do not have much information regarding the configurations chosen by the acquisition function during the optimization process. This thesis aimed indeed to introduce a tool to explain the choices of the AF to improve trust and transparency of the SMBO algorithm. Explaining a proposed configuration can be translated into a parameter attribution problem, in which the utility or desirability of a proposal is distributed among its parameters. Further, the desirability of a parameter can be further split into two components, namely mean and uncertainty contribution. In terms of HPO, this task is equivalent to assessing the importance of the hyperparameters and hence state the art methods like local parameter importance and ablation analysis could be applied. In section 3 we demonstrated, though, that these methods do not fulfill some important requirements for the task. The former does not allow for different contributions signs and the latter can not handle interactions properly. Later in section 5 we showed that the Shapley value not only solves the issues of LPI and AA, but in combination with the lower confidence bound, it also turns to be the perfect solution to the problem. Indeed, thanks to the linearity axiom, we can precisely decompose the desirability of a parameter into its two components. This is fundamental to understand how the EETO influences the choice LCB. After motivating our choice, in section 5.2 we introduced our solution `ShapleyMBO`. In addition to that, we provided a method to plot the results, named `plotShapleyMBO`, and to find a sufficiently big sample size for the SV estimation, named `checkSampleSize`. In the first part of the analysis `ShapleyMBO` was tested using the Hyper-Ellipsoid to see whether the results provided were in line with our expectations. While for $\lambda = 1$ they were rather univocal, for $\lambda = 10$ they showed interesting aspects of the acquisition function's behavior and the EETO: initially, the desirability of a parameter is dominated by the uncertainty contribution, indicating that the algorithm is exploring, yet sooner or later during convergence the mean contribution takes the lead, indicating that the algorithm begins to exploit

promising regions in the input space. What is more, we noticed that opposite contribution signs of mean and uncertainty are not rare, which is an evident sign that the EETO forces the algorithm to make sacrifices. Overall, for both λ, our method provided reasonable and consistent results in line with our expectations. In the second part of the analysis, we wanted to demonstrate how `ShapleyMBO` could be used in a real tuning example. For that, we explained the results of a tuning round of a multilayer perceptron. Next to analyzing specific iterations, attention should also be given to desirability paths that can reveal anomalies in the evolution of a parameter's desirability, as was the case for the learning rate. Here, `ShapleyMBO` helped us to understand that the performance achieved in the final iterations (low validation error) should not be attributed to the learning rate, which had a very poor mean contribution.

In conclusion, the results of both analyses clearly suggested that the SV is a valid method and `ShapleyMBO` is a precious diagnostic tool of great potential. This thesis only set the first milestone to increase the transparency of the SMBO algorithm with the help of the SV and several improvements can be made in future. First of all, `ShapleyMBO` should undergo a more exhaustive validation phase, which involves multiple, more demanding test functions and configuration spaces. In addition, further research should be dedicated to extending the desirability decomposition to other acquisition functions, for instance, the Expected Improvement, as today the decomposition is limited to the LCB only. An interesting possibility to enrich the our framework, as mentioned in section 8, would be to integrate other sampling strategies for the SV computation as well as the Baseline Shapley value. Last but not least, in the same section, we briefly addressed the practical implications of a parameter's mean contribution. We think that a more detailed research could be invested also in this topic because mean contributions might be used as an inexpensive HPI metric, which could potentially avoid more expensive post-hoc HPI analysis after SMBO tuning rounds.

References

[1] K. Aas, M. Jullum, and A. Løland. Explaining individual predictions when features are dependent: More accurate approximations to shapley values. *arXiv preprint arXiv:1903.10464*, 2019.

[2] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *25th annual conference on neural information processing systems (NIPS 2011)*, volume 24. Neural Information Processing Systems Foundation, 2011.

[3] A. Biedenkapp, M. Lindauer, K. Eggensperger, F. Hutter, C. Fawcett, and H. Hoos. Efficient parameter importance analysis via ablation with surrogates. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), Feb. 2017.

[4] A. Biedenkapp, J. Marben, M. Lindauer, and F. Hutter. Cave: Configuration assessment, visualization and evaluation. In *International Conference on Learning and Intelligent Optimization*, pages 115–130. Springer, 2018.

[5] B. Bischl, O. Mersmann, H. Trautmann, and C. Weihs. Resampling methods for meta-model validation with recommendations for evolutionary computation. *Evolutionary computation*, 20(2):249–275, 2012.

[6] B. Bischl, J. Richter, J. Bossek, D. Horn, J. Thomas, and M. Lang. mlrmbo: A modular framework for model-based optimization of expensive black-box functions. *arXiv preprint arXiv:1703.03373*, 2017.

[7] J. Bossek. smoof: Single- and Multi-Objective Optimization Test Functions. *The R Journal*, 9(1):103–113, 2017.

[8] E. Brochu, V. M. Cora, and N. De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[9] D. D. Cox and S. John. A statistical method for global optimization. In *[Proceedings] 1992 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1241–1246. IEEE, 1992.

[10] J. Drozdal, J. Weisz, D. Wang, G. Dass, B. Yao, C. Zhao, M. Muller, L. Ju, and H. Su. Trust in automl: Exploring information needs for establishing trust in automated machine learning systems. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pages 297–307, 2020.

[11] C. Fawcett and H. H. Hoos. Analysing differences between algorithm configurations through ablation. *Journal of Heuristics*, 22(4):431–458, 2016.

[12] M. Feurer and F. Hutter. Hyperparameter optimization. pages 3–38.

[13] A. Fréchette, L. Kotthoff, T. Michalak, T. Rahwan, H. H. Hoos, and K. Leyton-Brown. Using the shapley value to analyze algorithm portfolios. 2015.

[14] J. H. Friedman, B. E. Popescu, et al. Predictive learning via rule ensembles. *Annals of Applied Statistics*, 2(3):916–954, 2008.

[15] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1487–1495, 2017.

[16] G. Hooker. Generalized functional anova diagnostics for high-dimensional functions of dependent variables. *Journal of Computational and Graphical Statistics*, 16(3):709–732, 2007.

[17] F. Hutter. *Automated configuration of algorithms for solving hard computational problems*. PhD thesis, University of British Columbia, 2009.

[18] F. Hutter, H. Hoos, and K. Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *International conference on machine learning*, pages 754–762. PMLR, 2014.

[19] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pages 507–523. Springer, 2011.

[20] F. Hutter, L. Xu, H. H. Hoos, and K. Leyton-Brown. Algorithm runtime prediction: Methods & evaluation. *Artificial Intelligence*, 206:79–111, 2014.

[21] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[22] E. LeDell and S. Poirier. H2o automl: Scalable automatic machine learning. In *Proceedings of the AutoML Workshop at ICML*, volume 2020, 2020.

[23] T. Li, G. Convertino, W. Wang, H. Most, T. Zajonc, and Y.-H. Tsai. Hypertuner: Visual analytics for hyperparameter tuning by professionals. In *Proceedings of the Machine Learning from User Interaction for Visualization and Analytics Workshop at IEEE VIS*, 2018.

[24] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 4765–4774. Curran Associates, Inc., 2017.

[25] G. Luo. A review of automatic selection methods for machine learning algorithms and hyper-parameter values. *Network Modeling Analysis in Health Informatics and Bioinformatics*, 5(1):1–16, 2016.

[26] C. Molnar. *Interpretable Machine Learning*. 2019. `https://christophm.github.io/interpretable-ml-book/`.

[27] C. Molnar, G. Casalicchio, and B. Bischl. iml: An r package for interpretable machine learning. *Journal of Open Source Software*, 3(26):786, 2018.

[28] H. Park, J. Kim, M. Kim, J.-H. Kim, J. Choo, J.-W. Ha, and N. Sung. Visualhypertuner: Visual analytics for user-driven hyperparameter tuning of deep neural networks. In *Demo at SysML Conference*, 2019.

[29] H. Peters. *Game theory: A Multi-leveled approach*. Springer, 2015.

[30] V. Picheny, T. Wagner, and D. Ginsbourger. A benchmark of kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization*, 48(3):607–626, 2013.

[31] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020.

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

[33] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[34] L. S. Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[35] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25:2951–2959, 2012.

[36] E. Štrumbelj and I. Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41(3):647–665, 2014.

[37] M. Sundararajan and A. Najmi. The many shapley values for model explanation. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9269–9278, Virtual, 13–18 Jul 2020. PMLR.

[38] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855, 2013.

[39] J. N. Van Rijn and F. Hutter. Hyperparameter importance across datasets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2367–2376, 2018.

[40] Q. Wang, Y. Ming, Z. Jin, Q. Shen, D. Liu, M. J. Smith, K. Veeramachaneni, and H. Qu. Atmseer: Increasing transparency and controllability in automated machine learning. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2019.

A Figures

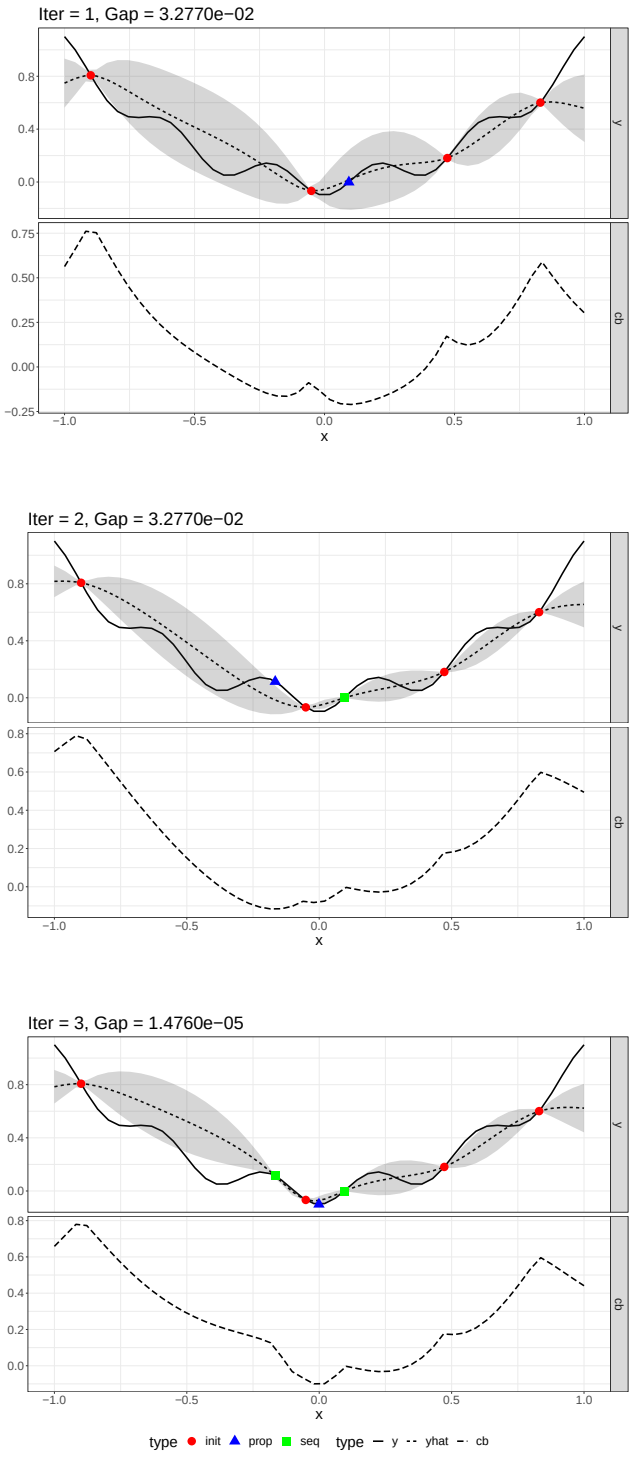

Figure 1: Optimization of the univariate cosine mixture function using SMBO with LCB and $\lambda = 1$ as acquisition function. For each iteration the target function (solid), surrogate model (dashed) with uncertainty estimates (grey areas), initial design (red circles), and proposed point in actual (blue triangle) and previous (green square) iterations are displayed on the top. Below the LCB, low values are associated with better acquisition and desirability because the target function is minimized.

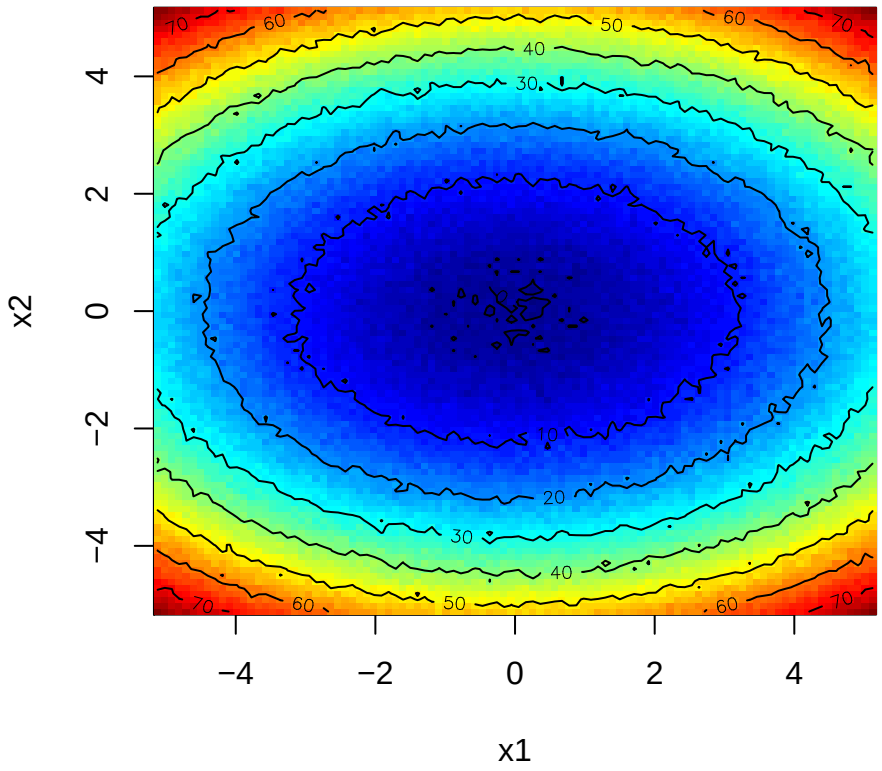

Figure 2: Contour plot of a bivariate Hyper-Ellipsoid sample with $\epsilon \overset{i.i.d}{\sim} \mathcal{N}(0, \sigma^2)$ observation noise, where $\sigma = 0.05 \cdot \hat{\sigma}_{HE}$. For $\hat{\sigma}_{HE}$ 10000 points were sampled at random with Latin Hypercube Sampling and then the standard deviation of their noise free function values was computed.

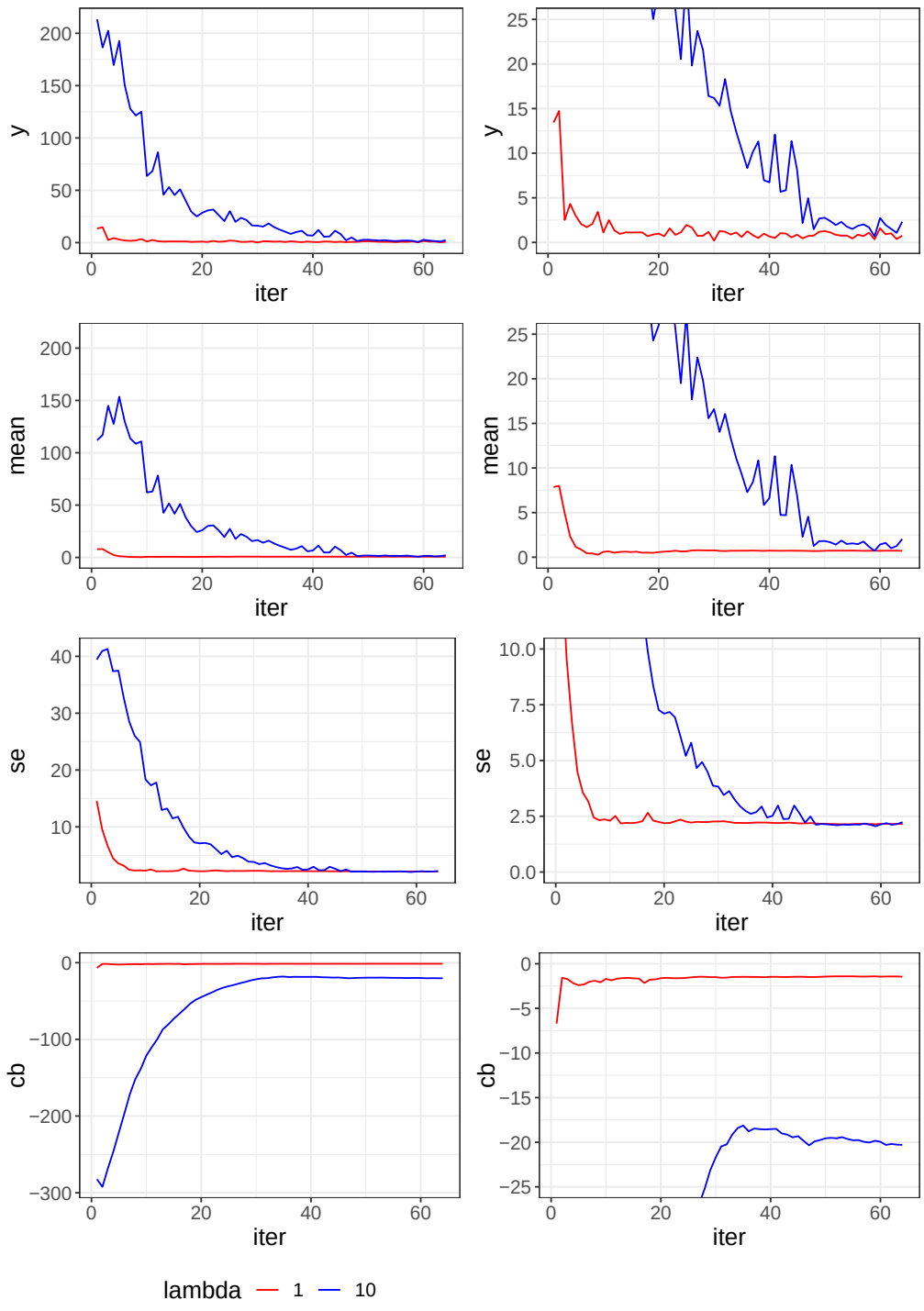

Figure 3: Paths of target y, mean (predicted target), uncertainty and LCB prediction for both λ in the optimization of the Hyper-Ellipsoid. Results are averaged over 30 BO runs. On the left plots are on original scale and plots on the right are a zoomed version thereof.

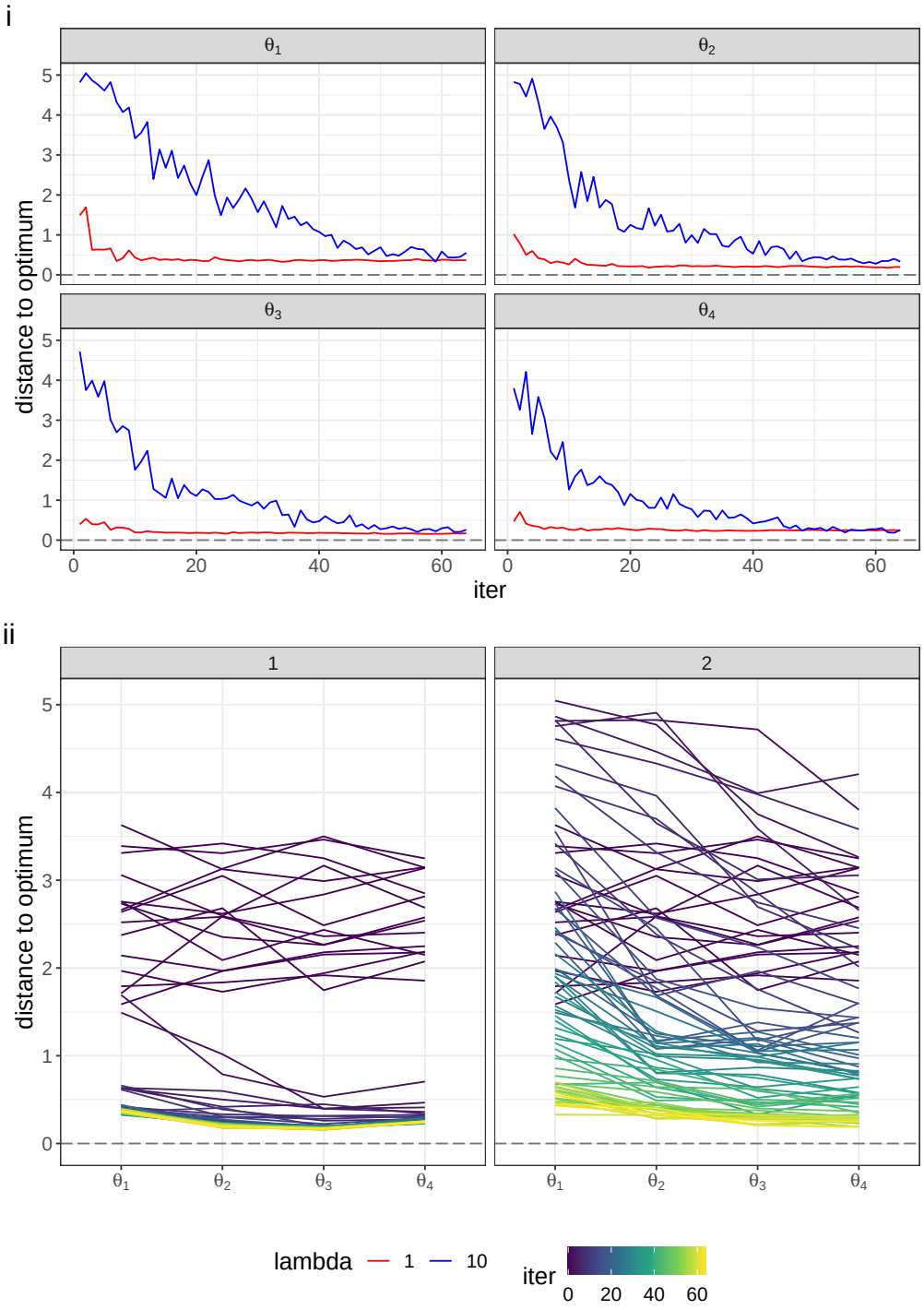

Figure 4: Paths of the configurations seen throughout the Hyper-Ellipsoid optimization for both λ. For each parameter the average distance of the actual parameter value to its optimum $\theta_j^* = 0$ is displayed. Average distance is computed over 30 BO runs. Plot i shows trajectories of proposed values for each parameter (initial design is not included). Plot ii is a parallel plot of the same configurations, but this time initial design is included. Initial design is the same for both λ and can be spotted on the left plot for $\lambda = 1$. The right plot ii is for $\lambda = 10$.

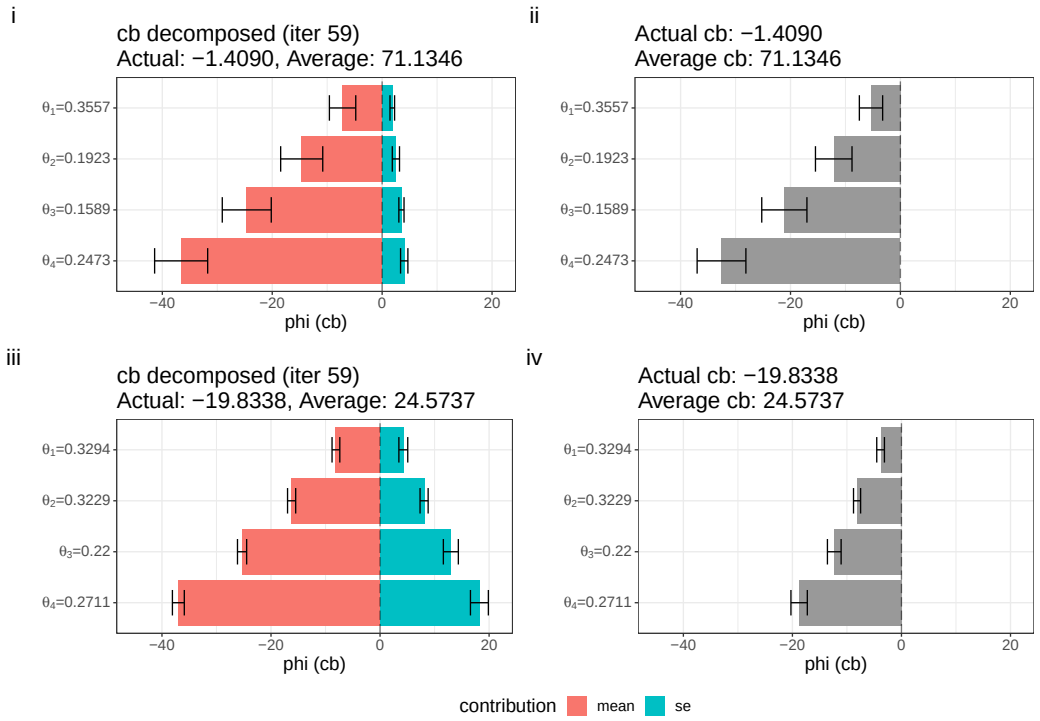

Figure 5: `ShapleyMBO` results in iteration 59. Plots i and ii for $\lambda = 1$ and plots iii and iv for $\lambda = 10$. Contributions are averaged over 30 proposals for each λ. On the right, the overall desirability of the parameters is displayed (*cb* contributions), and on the left the decomposition in m and se contribution. Uncertainty of the estimates is displayed with error bars using one standard deviation. On the vertical axis, instead of displaying the average parameter value, we used the average distance of the proposed configuration from their optimum for a better interpretation.

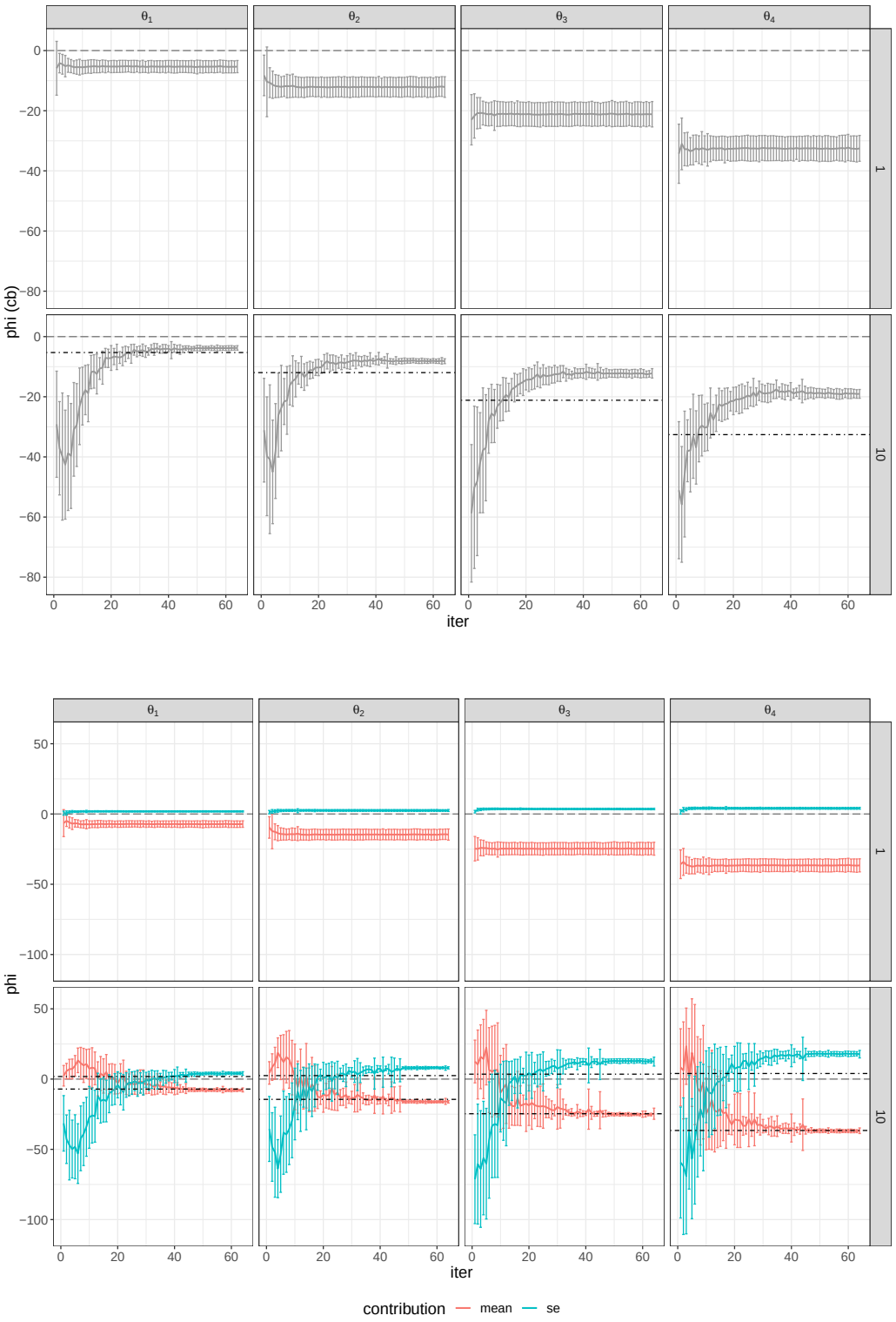

Figure 6: Desirability paths for the Hyper-Ellipsoid optimization. The plot on the top displays *cb* contributions with facets for parameters and λ. Beneath its decomposition in m and se contributions. Like in figure 5 contributions are averaged over 30 proposals in each iteration and the uncertainty of the estimate is displayed with error bars using one standard deviation. The black dot-dashed line in the $\lambda = 10$ plots displays the average contribution of the parameters in the $\lambda = 1$ process (see table 9 for exact values).

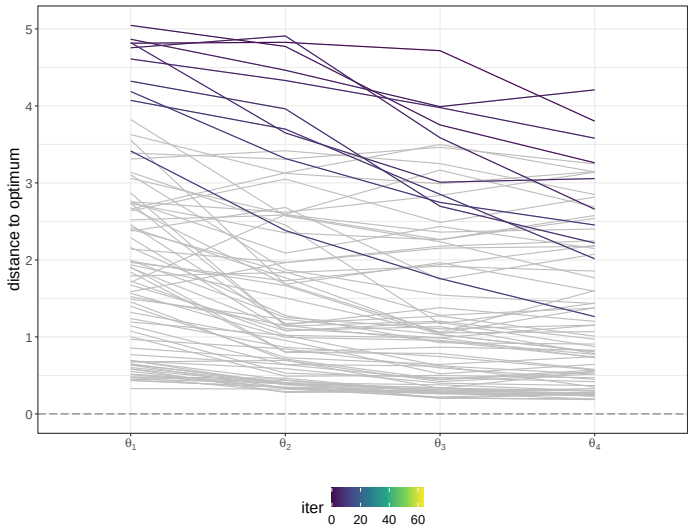

Figure 7: Same plot as figure 4 plot ii for $\lambda = 10$, but this time only the first 10 iterations are highlighted.

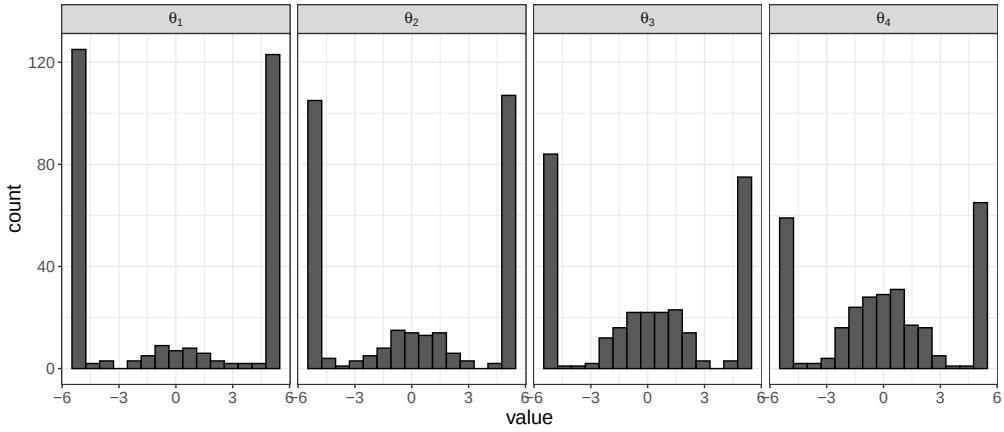

Figure 8: Histogram of the parameters including all proposed configuration values in the first 10 iterations of the Hyper-Ellipsoid optimization for $\lambda = 10$. On the vertical axis is the absolute frequency and on the horizontal axis the domain of the parameter is displayed. In total every histogram contains $10 \cdot 30 = 300$ observations.

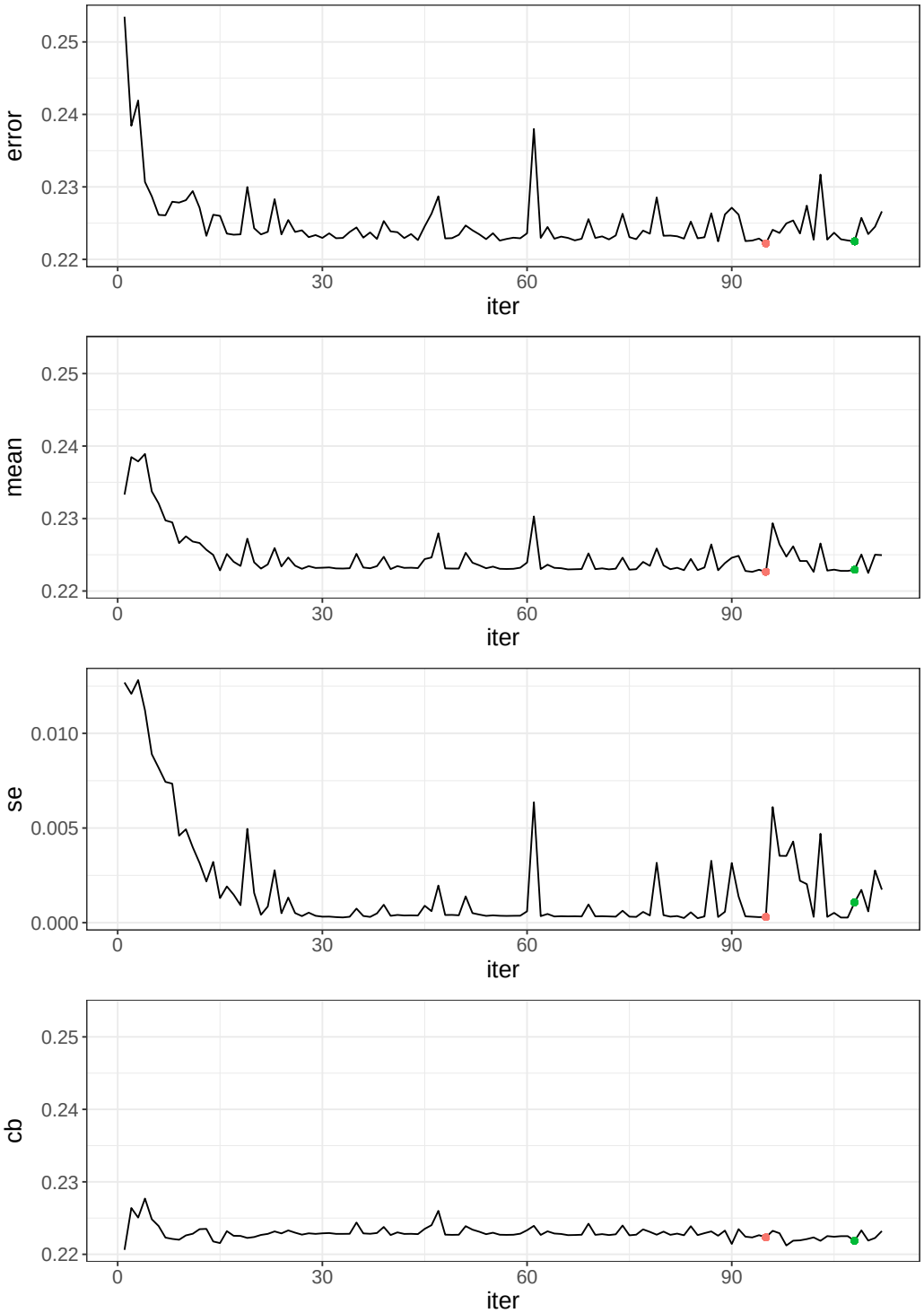

Figure 9: Paths of target value (validation error), mean (predicted error), uncertainty, and LCB prediction in the MLP tuning example. Red circle indicate values in iteration 95 and green in iteration 108.

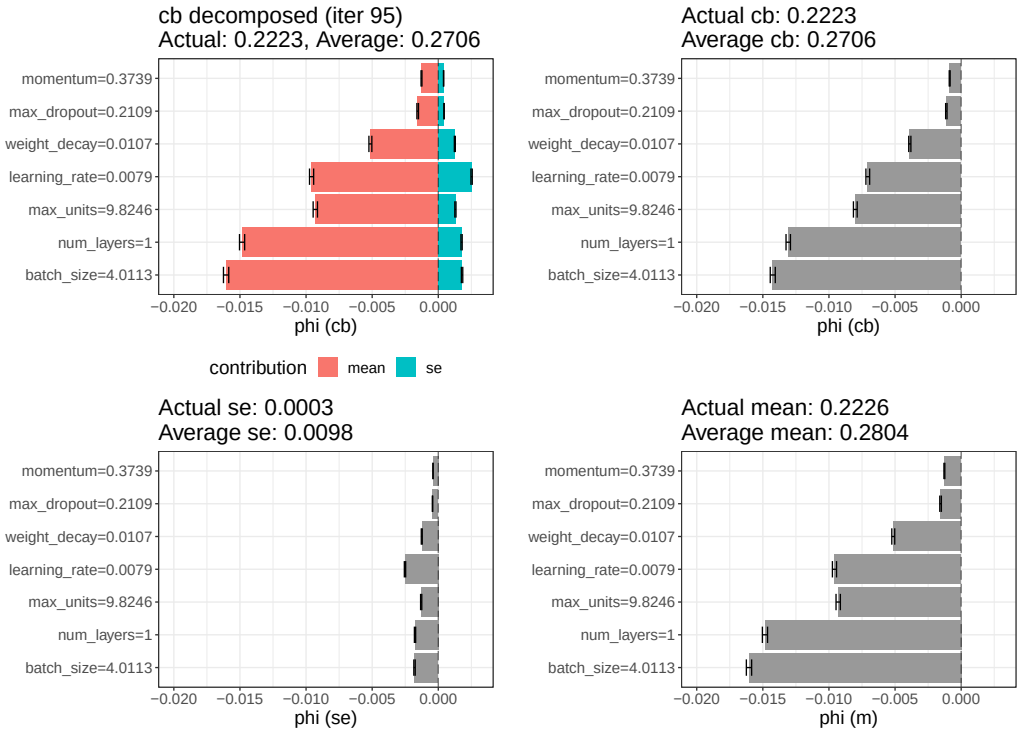

Figure 10: Contributions in iteration 95 of MLP tuning example. On top contributions on the *cb* are displayed: left decomposition plot and right the overall desirability of the parameters. At the bottom, m and se contributions are displayed separately. 95%-Confidence intervals (section 5.1) of the estimates are displayed with error bars. The original se contribution (grey) differs from the one in the decomposition plot (blue) because values in the latter are scaled with $-\lambda$. Above each plot, actual and average prediction of the proposal is returned. On the vertical axis, the configuration values are displayed and rounded to four digits for better visibility. This figure shows how the output of `plotShapleyMBO` for single iterations may look like.

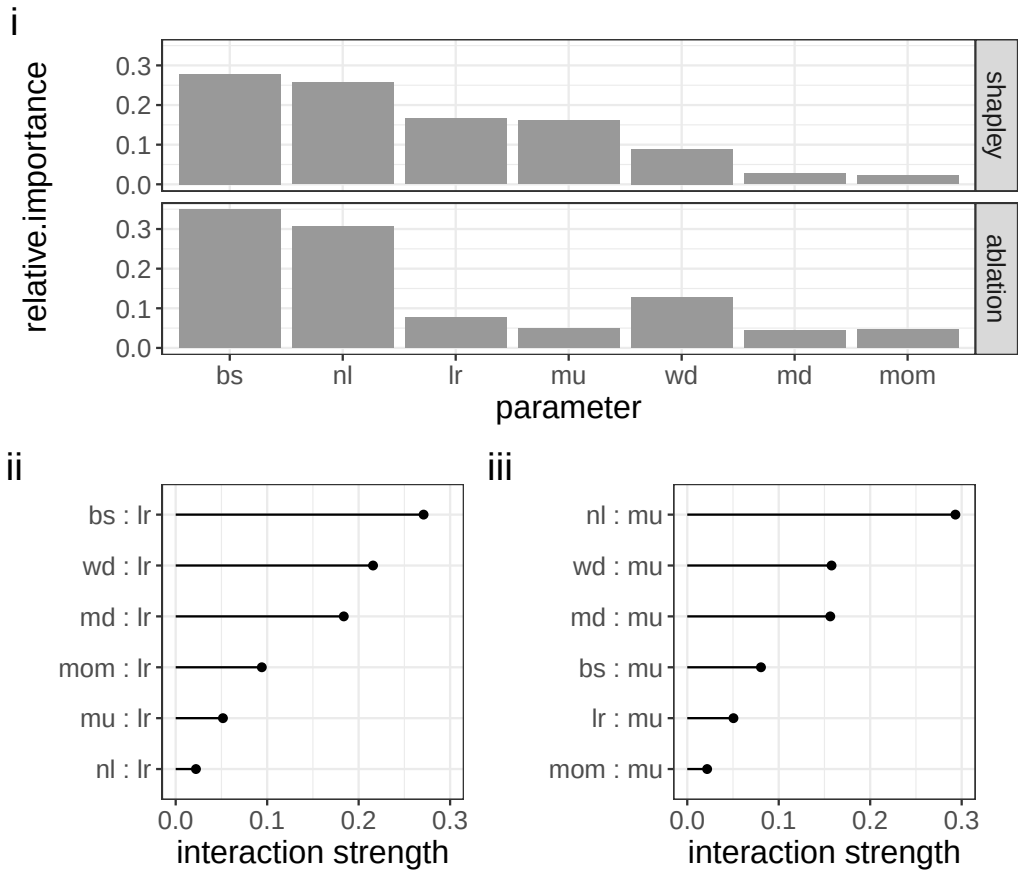

Figure 11: Relative m contribution of the parameters for configuration proposed in iteration 95 is displayed for both methods in plot i. To understand the difference in the results the interaction strength of learning rate (plot ii) and max units (plot iii) with other parameters is computed using Friedman's H-Statistic (interaction between two features).

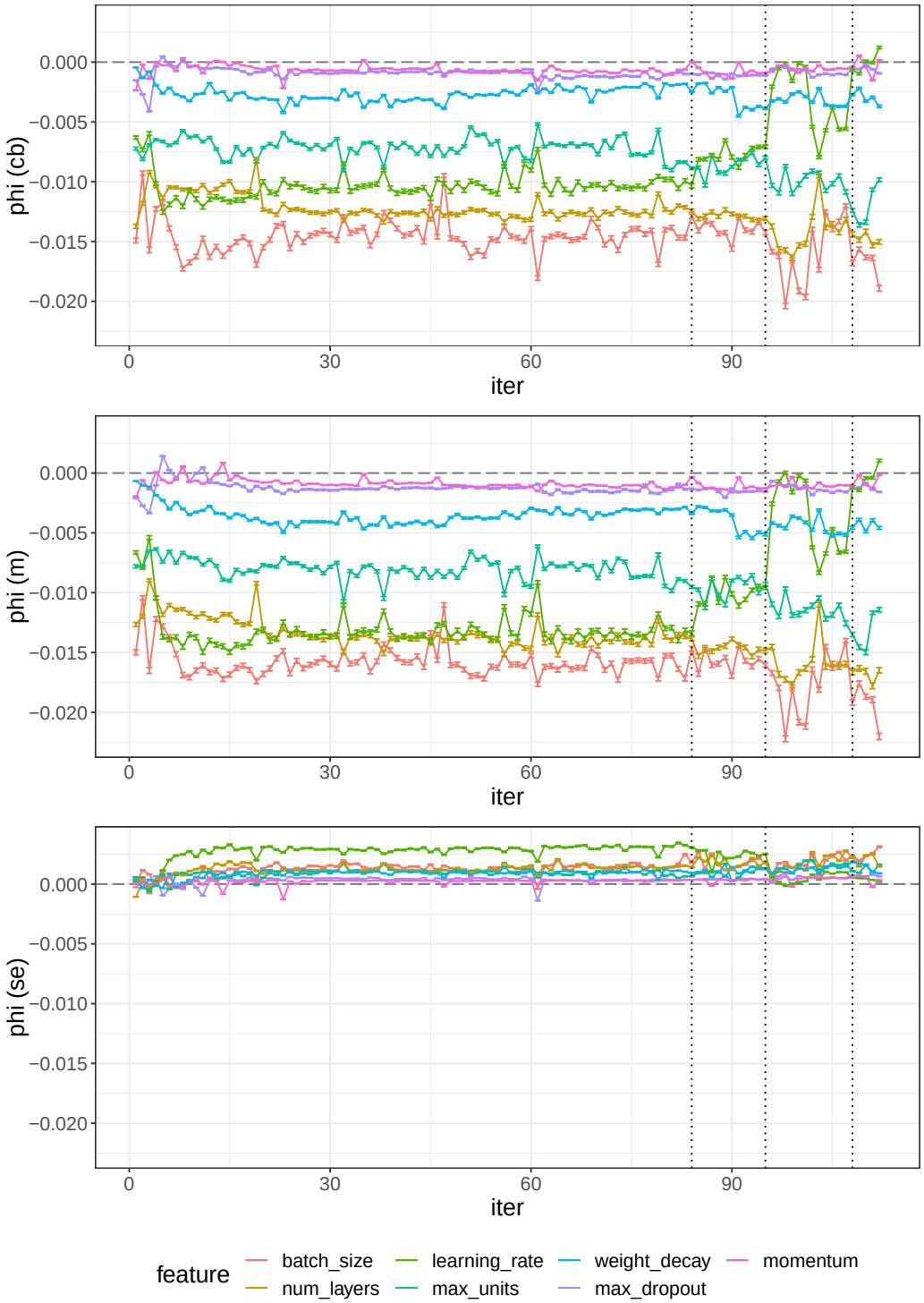

Figure 12: Desirability paths for the MLP optimization. The plot on the top displays *cb* contributions (desirability of parameters), beneath its decomposition in m and se contributions. 95%-Confidence intervals (section 5.1) of the estimates are displayed with error bars. The vertical dotted lines indicate respectively iteration 84 (turning point for lr paths change), 95 (explained in figure 10), 108 (explained in figure 14).

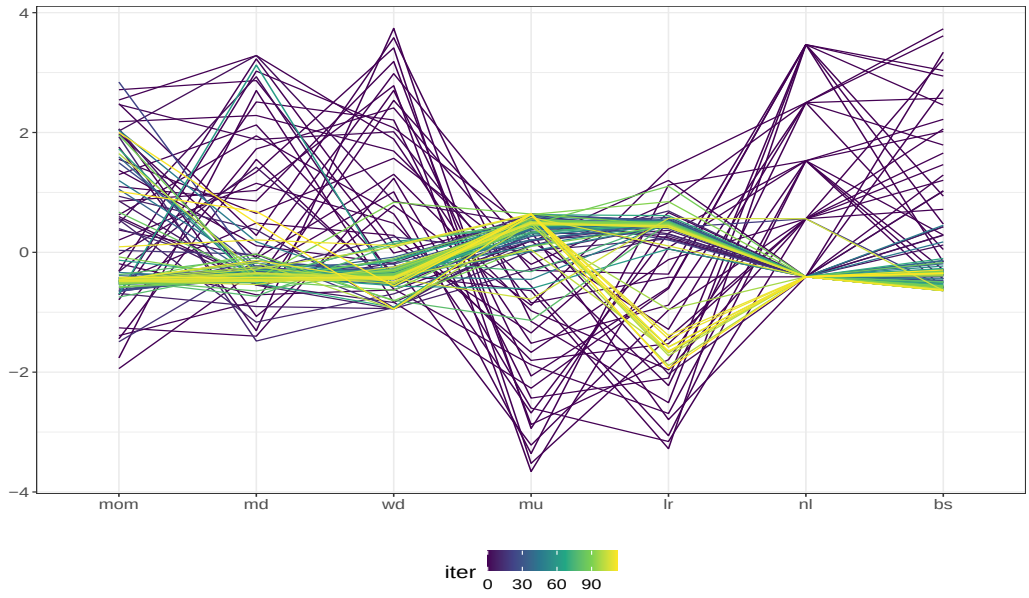

Figure 13: Parallel plot of the configurations visited in the MLP optimization problem (initial design included). For a better visibility, parameter values are univariately standardized by subtracting the mean and dividing by the standard deviation.

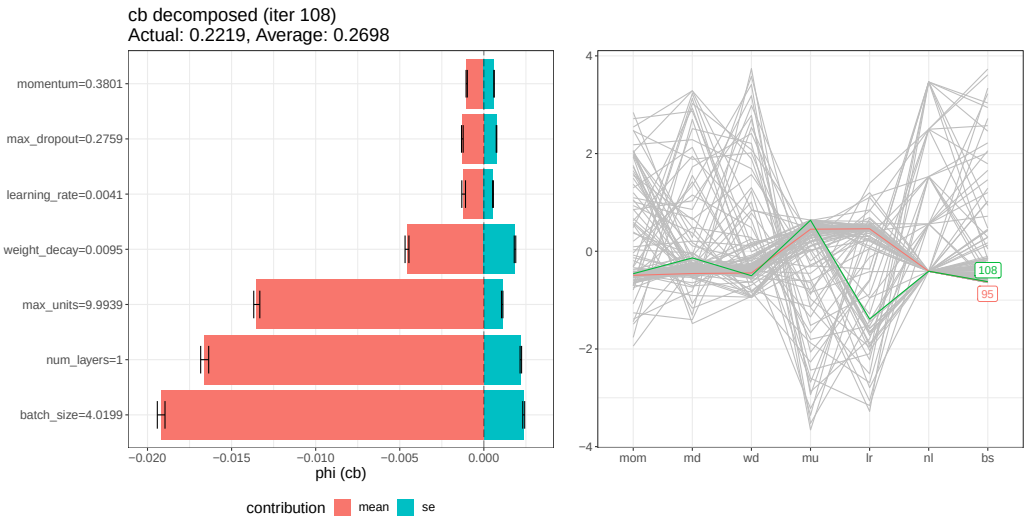

Figure 14: Left: *cb* contributions of configuration proposed in iteration 108 decomposed in m and se contributions including 95%-Confidence intervals of the estimates. Right: same parallel plot as in figure 13 but only proposal in iteration 95 (red) and 108 (green) are highlighted.

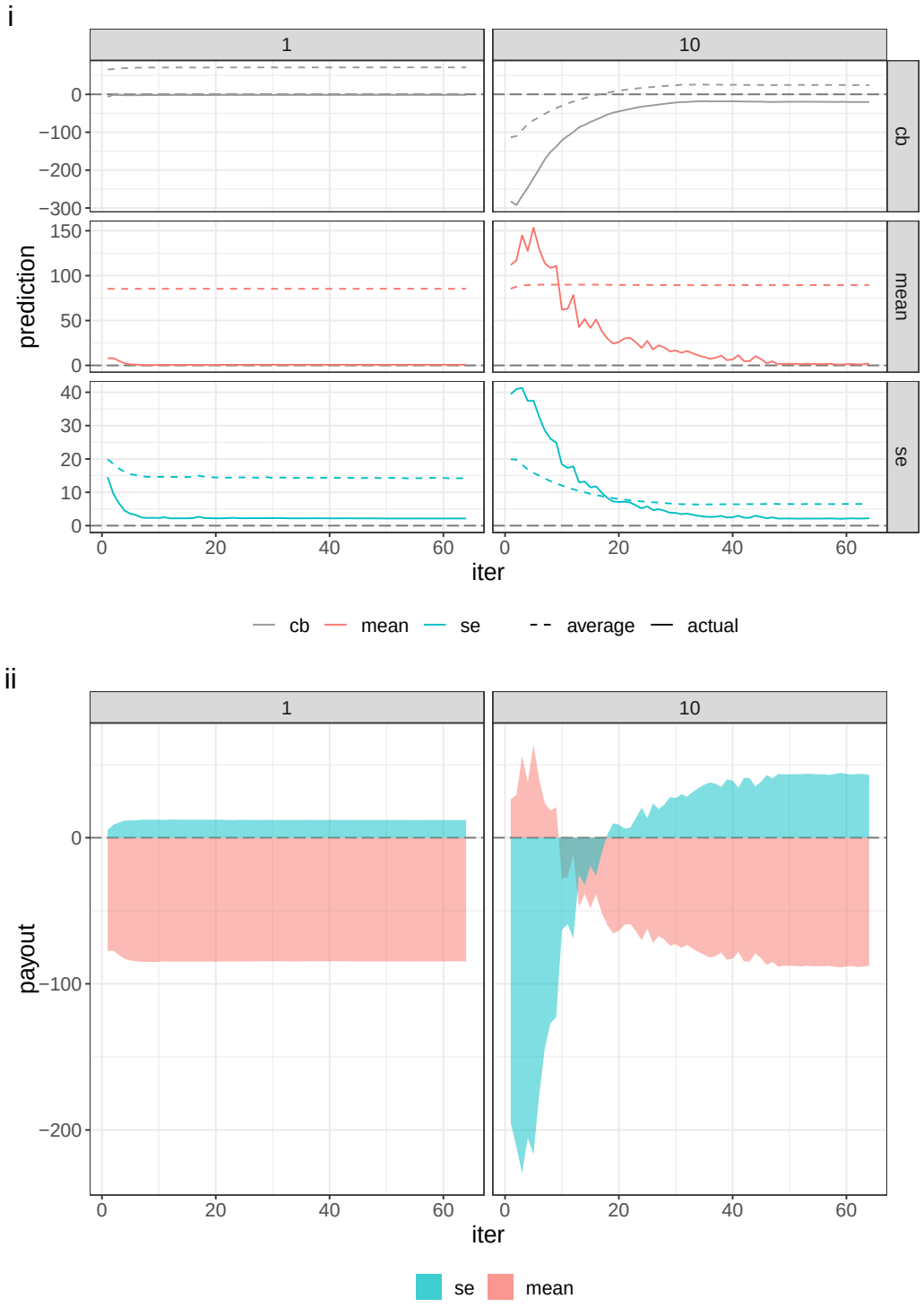

Figure 15: Payout paths for the Hyper-Ellipsoid optimization. Results are averaged over 30 BO runs. In plot i payouts are displayed separately. Scales are different for better visibility of each payout. The payout for each is given by the difference between the solid and the dashed line. Here, *se* predictions are not scaled with $-\lambda$ to better compare both λ values. Notice how the average *se* for $\lambda = 10$ decreases more due to better exploration. In plot ii m and *se* are displayed together, both centered around the respective average prediction (dashed line in plot i). Unlike plot i, here the *se* payout is scaled with $-\lambda$. The sum of both curves gives the *cb* payout.

B CB contribution and the linearity axiom

This section shows how the linearity axiom is used with mean m and uncertainty se to construct the average utility in space $\bar{cb}$ as well as the overall desirability of a parameter $\hat{\phi}_j(cb)$.

1) $\bar{cb}$: let $\{(\boldsymbol{\theta}^{(i)}, \hat{cb}(\boldsymbol{\theta}^{(i)})\}_{i=1}^{n}$ be the sampling population for the SV computation. Then the average desirability or utility in space is

$$\begin{aligned}
\bar{cb} &= \frac{1}{n}\sum_{i=1}^{n} \hat{cb}(\boldsymbol{\theta}^{(i)}) \\
&= \frac{1}{n}\sum_{i=1}^{n} \hat{m}(\boldsymbol{\theta}^{(i)}) - \lambda \hat{se}(\boldsymbol{\theta}^{(i)}) \\
&= \frac{1}{n}\sum_{i=1}^{n} \hat{m}(\boldsymbol{\theta}^{(i)}) - \lambda \frac{1}{n}\sum_{i=1}^{n} \hat{se}(\boldsymbol{\theta}^{(i)}) \\
&= \bar{m} - \lambda \bar{se}.
\end{aligned}$$

2) $\hat{\phi}_j(cb)$: this computation is carried out by `computePhiCb` (see GitHub repository). Refer to algorithm 2 for the notation. Let $\tilde{\boldsymbol{\theta}}$ be the proposal or explicand and K the number of Monte Carlo samples. The desirability of a parameter is

$$\begin{aligned}
\hat{\phi}_j(cb) &= \frac{1}{K}\sum_{k=1}^{K} \hat{cb}(\tilde{\boldsymbol{\theta}}_{+j}^{(k)}) - \hat{cb}(\tilde{\boldsymbol{\theta}}_{-j}^{(k)}) \\
&= \frac{1}{K}\sum_{k=1}^{K} \left[\hat{m}(\tilde{\boldsymbol{\theta}}_{+j}^{(k)}) - \lambda \hat{se}(\tilde{\boldsymbol{\theta}}_{+j}^{(k)})\right] - \left[\hat{m}(\tilde{\boldsymbol{\theta}}_{-j}^{(k)}) - \lambda \hat{se}(\tilde{\boldsymbol{\theta}}_{-j}^{(k)})\right] \\
&= \frac{1}{K}\sum_{k=1}^{K} \Delta\hat{cb}^{(k)}.
\end{aligned}$$

Finally the variance of the estimate is

$$\hat{\mathbb{V}}(\hat{\phi}_j(cb)) = \frac{1}{K-1}\sum_{k=1}^{K} (\,\Delta\hat{cb}^{(k)} - \overline{\Delta cb}\,)^2.$$

C checkSampleSize

This method offers a possible way to find a sufficient sample size for the SV estimation. Formally, let $\mathcal{P}(v) = \hat{f}^v(\tilde{\boldsymbol{\theta}}) - \frac{1}{n}\sum_i \hat{f}^v(\boldsymbol{\theta}^{(i)})$ be the payout to distribute among the parameters of explicand $\tilde{\boldsymbol{\theta}}$, where $\hat{f}^v$ is the model that predicts $v = \{cb, m, se\}$ and $\{\boldsymbol{\theta}^{(i)}\}_{i=1}^n$ is a set of instances. According to the **efficiency axiom** it should hold $\sum_j \phi_j(v) = \mathcal{P}(v)$, though because the true contributions are approximated, the computation comes with an *efficiency error*, which can be defined as $\Delta_{eff}^K(v) = \sum_j \hat{\phi}_j^K(v) - \mathcal{P}(v)$ with K being the number of Monte Carlo samples. The higher the sample size the smaller $\Delta_{eff}^K(v)$ should get according to the asymptotic distribution of $\hat{\phi}_j(v)$ (see section 5.1). When the error is greater (smaller) than zero too much (less) payout has been distributed and the quantity should be subtracted from (added to) the contributions. It can be especially problematic if, after adjusting for it, the ranking of the SVs changes. Although it is unknown which contributions exactly must be modified (unknown where the error(s) exactly happens), arguably the most unfair correction would be to entirely assign or detract $\Delta_{eff}^K(v)$ from one parameter only. Notice, that we are not actually interested in correcting the results, but instead in assessing if K was set high enough. Further, notice that adding the error to one parameter is equivalent to subtracting it from another for the scope of the task. Hence, to facilitate the computation we define $\Delta_{eff}^K(v)$ as $|\sum_j \hat{\phi}_j^K(v) - \mathcal{P}(v)|$. Finally, let $\delta^K(v) = min\{\boldsymbol{d_1}(\hat{\phi}_j^K(v), \hat{\phi}_l^K(v))_{j\neq l}\}$ be the smallest absolute difference (L_1 distance) between the Shapely values of two different parameters. If condition $\Delta_{eff}^K(v) < \delta^K(v)$ holds, then K is sufficiently high, because adding the efficiency error to the weaker parameter between j and l or equivalently removing the error from the stronger one will not change the importance ranking in the results. If the condition does not hold, then K should be increased. Starting from a lower size, a sufficient sample size can then be found trough greedy forward search. In the figure below an intuitive example of the method is shown and in algorithm 4 the main steps are summarized.

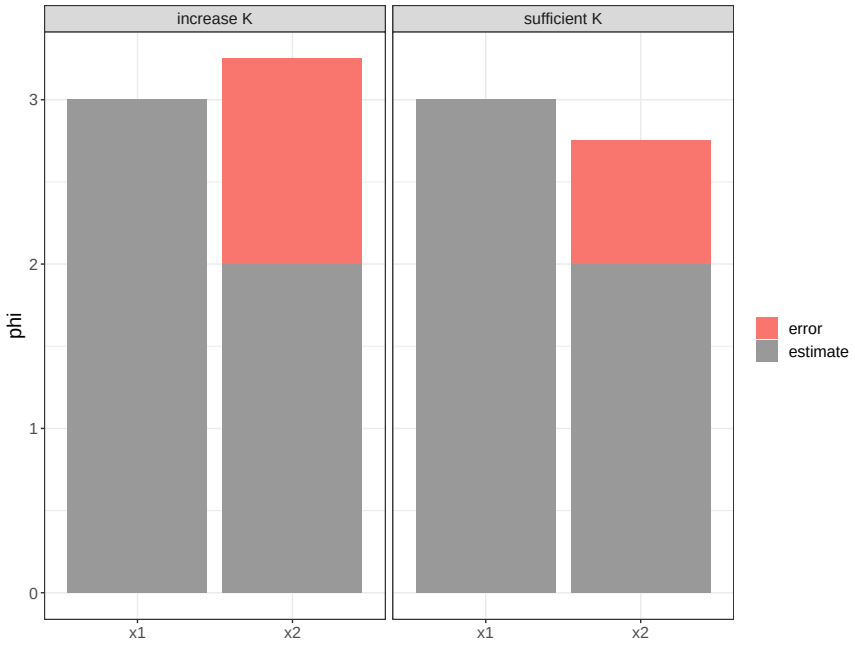

Figure 16: Example of the idea behind `checkSampleSize`. Estimated SVs of parameters x_1 and x_2 in grey are purely indicative and give threshold $\delta^K = 3 - 2 = 1$. The efficiency error in red varies for different K. Suppose $\Delta^K_{eff}(v) < 0$ and the error is hence added entirely to x_2. In the left plot the sample size K is not sufficient because redistributing the resulting error would make x_2 more important than x_1. Instead in the right plot K would be sufficient. In case $\Delta^K_{eff}(v) > 0$ removing the error entirely from x_1 would lead to the same conclusion for K.

D Algorithms

Algorithm 1 Sequential Model-Based Optimization basic procedure

create an initial design $\mathcal{D} = \{(\boldsymbol{\theta}^{(i)}, \Psi^{(i)})\}_{i=1}^{n_{init}}$
while termination criterion is not fulfilled **do**
    fit a surrogate model $\hat{f}$ on design $\mathcal{D}$
    propose $\boldsymbol{\theta}^{new} = \arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} u(\boldsymbol{\theta} | \mathcal{D})$
    evaluate Ψ on $\boldsymbol{\theta}^{new}$ and **update** $\mathcal{D} \leftarrow \mathcal{D} \cup (\boldsymbol{\theta}^{new}, \Psi(\boldsymbol{\theta}^{new}))$
end while

Algorithm 2 Estimation of the Conditional Expectation Shapley value

Require: explicand $\tilde{\boldsymbol{\theta}}$, feature index j, model $\hat{f}$ and sample size K
for $k = 1 \rightarrow K$ **do**
    sample (at random and with replacement) an instance $\boldsymbol{z} \in \boldsymbol{\Theta}$
    sample (at random and with replacement) an order $\pi \in \Pi(P)$
    order $\tilde{\boldsymbol{\theta}}$ and $\boldsymbol{z}$ according to π
        $\tilde{\boldsymbol{\theta}}_\pi = (\tilde{\theta}_{(1)}, \ldots, \tilde{\theta}_{(p)})$
        $\boldsymbol{z}_\pi = (z_{(1)}, \ldots, z_{(p)})$
    construct two new instances
        $\tilde{\boldsymbol{\theta}}_{+j} = (\tilde{\theta}_{(1)}, \ldots, \tilde{\theta}_{(j-1)}, \tilde{\theta}_{(j)}, z_{(j+1)}, \ldots, z_{(p)})$
        $\tilde{\boldsymbol{\theta}}_{-j} = (\tilde{\theta}_{(1)}, \ldots, \tilde{\theta}_{(j-1)}, z_{(j)}, z_{(j+1)}, \ldots, z_{(p)})$
    $\hat{\phi}_j^k(v) = \hat{f}(\tilde{\boldsymbol{\theta}}_{+j}) - \hat{f}(\tilde{\boldsymbol{\theta}}_{-j})$
end for
$\hat{\phi}_j(v) = \frac{1}{K} \sum_{k=1}^{K} \hat{\phi}_j^k(v)$

Algorithm 3 `ShapleyMBO`

Require: SMBO result object *mbo*, iteration of interest t, sample size K
get explicand from *mbo*: $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}_t^{new}$
sample $1000 \cdot p$ points $\boldsymbol{Z}$ from $\boldsymbol{\Theta}$ to approximate the space
compute $\hat{\boldsymbol{\phi}}(m) = (\hat{\phi}_1(m), \ldots, \hat{\phi}_p(m))$:
    get SM from *mbo*: $\hat{f}_m = \hat{f}_t^{mean}$
    explain $\tilde{\boldsymbol{\theta}}$ with `iml::Shapley()` using $\boldsymbol{Z}$, $\hat{f}_m$ and K
compute $\hat{\boldsymbol{\phi}}(se) = (\hat{\phi}_1(se), \ldots, \hat{\phi}_p(se))$:
    get SM from *mbo*: $\hat{f}_{se} = \hat{f}_t^{uncertainty}$
    explain $\tilde{\boldsymbol{\theta}}$ with `iml::Shapley()` using $\boldsymbol{Z}$, $\hat{f}_{se}$ and K
compute $\hat{\boldsymbol{\phi}}(cb)$ with linearity axiom: $\hat{\boldsymbol{\phi}}(cb) = \hat{\boldsymbol{\phi}}(m) - \lambda \hat{\boldsymbol{\phi}}(se)$

Algorithm 4 `checkSampleSize`

Require: `ShapleyMBO` results for $\tilde{\boldsymbol{\theta}}_t$ with size K, models $\hat{f}^v$ with $v = \{cb, m, se\}$

for w in v **do**
    compute payout $\mathcal{P}(w)$, error $\Delta^K_{eff}(w)$ and threshold $\delta^K(w)$
    if $\Delta^K_{eff}(w) < \delta^K(w)$ **then**
        $K_w = \texttt{T}$
    end if
    if $\Delta^K_{eff}(w) \geq \delta^K(w)$ **then**
        $K_w = \texttt{F}$
    end if
end for
if $(K_{cb}, K_m, K_{se}) = (\texttt{T}, \texttt{T}, \texttt{T})$ **then**
    K is high enough
else if $(K_{cb}, K_m, K_{se}) \neq (\texttt{T}, \texttt{T}, \texttt{T})$ **then**
    K should be increased

Eidestattliche Erkärung

Ich versichere hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst habe und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ort, Datum

.....

(Federico Croppi)