Optimally weighted ensembles of surrogate models for sequential parameter optimization

Echtenbruck, M.M.

Optimally weighted ensembles of surrogate models for sequential parameter optimization

Martina Echtenbruck

Optimally weighted ensembles of surrogate models for sequential parameter optimization

Proefschrift

ter verkrijging van
de graad van Doctor aan de Universiteit Leiden,
op gezag van Rector Magnificus prof.mr. C.J.J.M. Stolker,
volgens besluit van het College voor Promoties
te verdedigen op donderdag 2 juli 2020
klokke 11.15 uur

door

Martina Marion Echtenbruck

geboren te Duisburg, Duitsland
in 1977

Promotiecommissie

Promotor:	Prof. Dr. T.H.W. Bäck	
	Prof. Dr. T. Bartz-Beielstein	(TH Köln, Germany)
Co-promotor:	Dr. M.T.M. Emmerich	
Overige leden:	Prof. Dr. A. Plaat	Voorzitter
	Prof. Dr. S. Mostaghim	(University Magdeburg, Germany)
	Prof. Dr. K. Giannakoglou	(National Technical University Athens, Greece)
	Dr. B. van Stein	
	Prof. Dr. M.M. Bonsangue	Secretaris

Contents

Chapter 1

Introduction

A common method to perform an optimization on a function that can not be optimized analytically is to perform a search on this function by evaluating points on the function at strategically chosen positions. However, in real-world optimization tasks, the search is often constrained by time or cost for the function evaluations.

1.1 Optimization of Industrial Problems

A typical example of such optimization problems is a cyclone dust separator [1, 2]. As Slack et al. state, "the cyclone dust separator is perhaps the most widely used separation device to be found in industry. It owes its popularity to the low manufacturing and maintenance costs brought about by its simple design. There are no moving parts in the device itself, which can be constructed from a wide range of materials including refractories for high-temperature operation. Combined with moderate pressure drop and a range of throughputs and efficiencies, these advantages have made the cyclone the most attractive solution to separation in both gas-solid and liquid-solid systems" [3].

Cyclone dust separators exist in a large variety of shapes, but the most common design is the reverse-flow cyclone as depicted in Figure 1.1.
The fluid is induced into the cyclone through the inlet on the upper end of the cyclone. By the position of the inlet and the shape of the cyclone body, the fluid is forced on a circular path. The emerging centrifugal forces, caused through

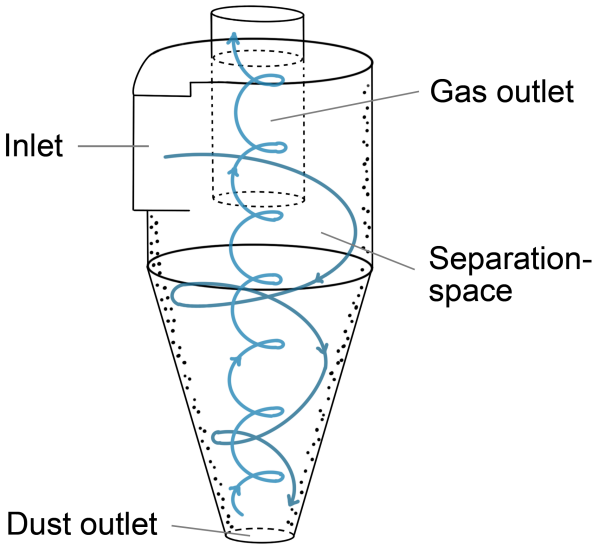


Figure 1.1: Depicted is a schematic representation of a cyclone dust separator with an illustration of the internal flow. The fluid enters the system through a lateral inlet at the upper end of the cyclone and moves then in a downward swirl along the outer areas of the cyclone. When reaching the lower end, the flow changes the direction and turns upward again, in a more narrow swirl, before it leaves the system through the gas outlet at the upper end of the cyclone. The dust particles are separated from the fluid through centrifugal forces, they are moved against the outer wall of the cyclone and then fall through the dust outlet at the lower end of the system.

the circular swirl of the fluid, separate the dust particles from the fluid and fall through the dust outlet at the lower end of the cyclone. However, the fluid, which is now separated from the dust particles, leaves the cyclone through the gas outlet on the upper end of the cyclone.

Main quality indicators for cyclone dust separators are collection efficiency and pressure drop. The collection efficiency, defined as the fraction of dust particles filtered from the fluid, reflects how well the cyclone dust separator performs its primary task. However, the pressure drop has the main impact on operational cost. This is aggravated by the fact, that these two criteria are conflicting, i.e., the settings allowing for the best collection efficiency may not correspond to the setting that enables the lowest pressure drop.

The quality indicators are heavily influenced by the geometry of the cyclone,

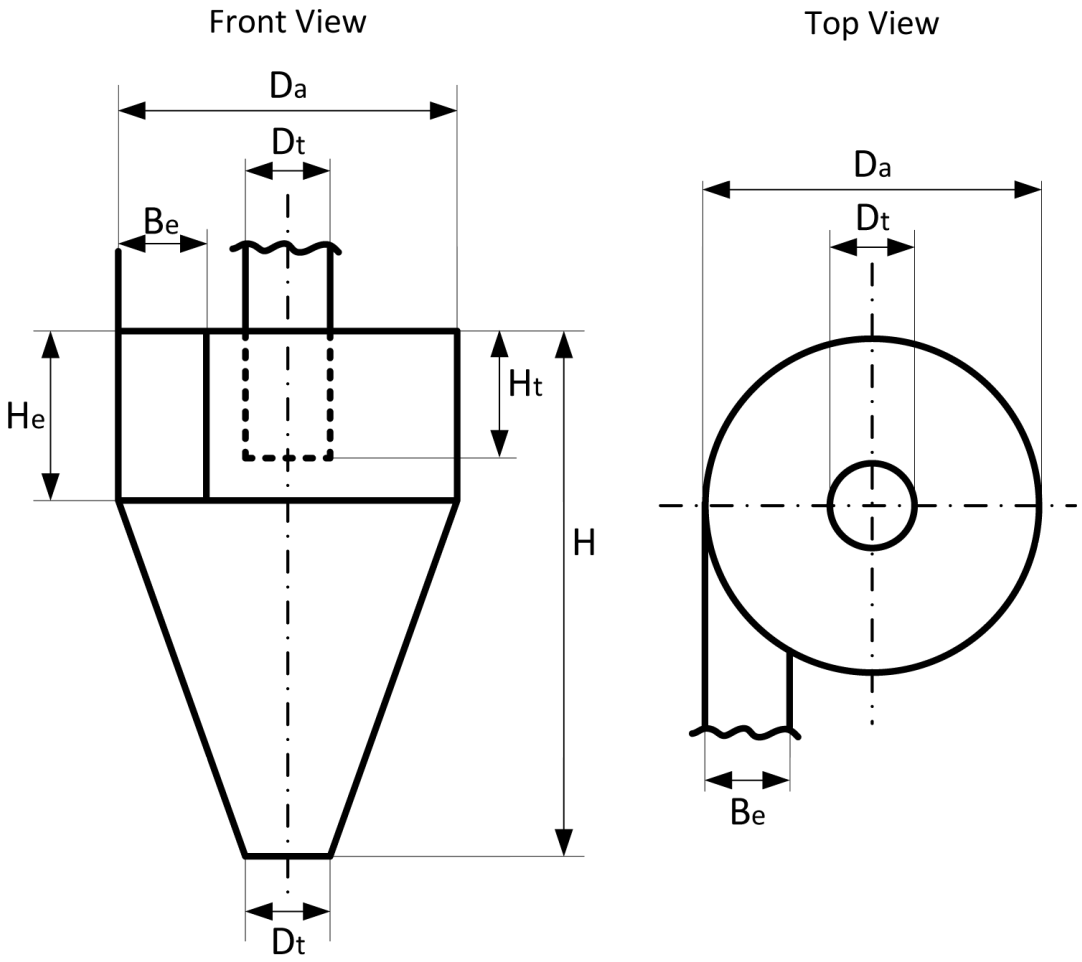


Figure 1.2: Depicted is a schematic representation of a cyclone dust separator (from [1]). Shown is the front view as well as the top view with all critical parameters indicated. These are height H and diameter D_a of the cyclone, diameter D_t and immersion H_t of the outlet pipe and width B_e and height H_e of the inlet.

which is determined by several design parameters, like the height or diameter of the cyclone. All critical parameters are depicted in Figure 1.2. The performance of a newly designed cyclone can be evaluated with Computational Fluid Dynamics (CFD) simulations. Such simulations are, depending on the required accuracy, extremely time-consuming.

All in all, this problem definition specifies an expensive, in terms of calculation time, black-box function whose function values, collection efficiency, and pressure drop, are determined by a set of parameters as depicted in Figure 1.2. In

modern computer-aided design environments, simulation-based optimization is a standard tool for finding optimal parameter settings [4, 5, 6, 7]. However, the performance of such automated-design optimization tools is often challenged by the high demand in terms of computational effort. For instance, physics simulations including fluid dynamics or detailed energy flow computations in real-world structures easily entail several minutes or even hours of runtime for a single design [8]. Optimization requires the repeated execution of such time-consuming simulations, and this requires optimal use of the information gained from each simulation.

Further well-known engineering problems are, for example, the optimization of an airfoil shape for an aircraft wing [9, 10] or aerodynamic shape optimization in the automotive industry [11, 12]. For such problems, a CFD simulation has to be carried out in order to evaluate a specific shape with respect to the different design objectives and other constraints. Jameson et al. (2018) [13] state, that some of these simulations take up to 3 days, what makes a design optimization task impossible since in general lots of simulation evaluations would be needed.

1.2 Motivation and Aim

In global optimization of expensive black-box functions, it is a common technique to learn a surrogate-model, e.g., regression model, of the response function from available evaluations and to use this model to decide on the location of future evaluations. Sequential parameter optimization (SPO) is a well-known approach for solving black-box optimization problems with expensive function evaluations [14, 15]. It combines a sequential experimental design approach and tools for reducing expensive function evaluations on the original model by replacing them partly with fast approximate evaluations on surrogate models. SPO resembles Bayesian Global Optimization [16, 17], but it is less specific in the particular regression model as it does not necessarily make use of uncertainty quantification. SPO packages, such as the SPO Toolbox (SPOT), come with a large variety of surrogate models (base models) from which the user can choose.

Still, the choice of the surrogate model can have a significant influence on the solution quality and performance of the optimizer. Burnham et al. even state that the choice of the right surrogate model is the most critical question in making statistical inferences [18]. However, in order to make meaningful decisions on which surrogate model to select for a given problem, often expert knowledge is needed.

This includes knowledge about the objective function and the characteristics of the surrogate model likewise.

In many situations, preliminary knowledge about the function, or all available models, is not available. To overcome this problem, it would be beneficial if the algorithm could learn all by itself which surrogate model type suits the problem best, based on the given data. This can be done by evaluating different models on available training data and using a statistical model selection approach to select the most promising surrogate model.

But how to handle the situation when there is more than one strong model in the set? In such circumstances, it might be beneficial to combine inference output across several models. Such methods will be referred to as ensemble models. Different approaches to achieve this are known to literature. In Chapter 3 a short overview of previous work regarding ensemble models is given, and a taxonomy of ensemble models is defined. However, so far, only few work has been done on adopting ensemble methods specifically for sequential optimization, which holds its challenges for efficient ensemble modeling.

The overarching goal is to release the user from the burden to select the right surrogate model from a set of heterogeneous surrogate models. From this aim, the main research question of this thesis can be derived:

- Is it possible to create an ensemble building strategy that selects or combines heterogeneous surrogate models to achieve the best possible result and works reliably and as accurately as possible on arbitrary objective functions?

Since many real-world problems are constrained to a comparatively small number of function evaluations, the main focus is laid on regression models and SPO processes that are able to work with smaller numbers of function evaluations.

Black-box optimization refers to a problem definition where an optimum of a function is searched for that cannot be optimized analytically. Since this is, in general, the case for real-world problems, the focus is also laid on black-box optimization. In this thesis, without loss of generality, all optimization tasks are considered as minimization tasks.
Optimization processes that rely on the assistance of surrogate models are referred to as surrogate-based optimization (SBO). The process of training a surrogate model (also referred to as model) on available data is denoted as fitting the model to the data. After the model has been fitted to the data, the model provides an approximation of the underlying function, which is based on the inherent assumptions of the model and the available data. Determining an approximation

of unknown function value with the help of a model is denoted as a prediction.

1.3 Overview of this Thesis

This thesis is structured as follows:

Chapter 2 gives an introduction to all preliminary information needed for the understanding of this work. It starts by introducing the models that are used in this thesis. Then, a brief introduction to SPO as well as to the SPO framework which is used in this thesis is given. Moreover, the objective functions used for the experiments are introduced.

Chapter 3 gives an overview of previous developments in this area, different ensembles approaches and applications, and defines a taxonomy of ensembles. The advantages and disadvantages of the diverse approaches are considered with regard to the general goal of this work. The taxonomy of ensembles of surrogates that is specified in this chapter is an original work of this thesis and has not yet been published.

Chapter 4 regards the findings of Chapter 3 within the premises of the overall goal of this work and draws appropriate conclusions. A new ensemble building approach is then derived from these conclusions, implemented and thoroughly analyzed.

Chapter 5 performs the step from static modeling to SPO. The designed approach is adapted for and applied to SPO. Experiments are carried out, and results are analyzed to allow for further insights into the functioning of the method.

Chapter 6 summarizes the works of this thesis and discusses the methods presented and the results obtained. Also, possible future work and additional open questions in this research area are shown up and discussed.

1.4 Overview of Publications

Substantial parts of this thesis rely on works that have previously been published or are in the process of being published during the writing of this thesis. Some of these works are incorporated only contentwise, and others are in large parts adopted verbatim. For the sake of clarity, the way how each publication has been included in this thesis is outlined at the end of the respective chapters. Of concern are primarily the following publications ordered by the chapter of appearance.

Chapter 4
Martina Friese and Martin Zaefferer. Two challenges in surrogate-modeling: Merging surrogate-models into ensembles and dealing with structured or combinatorial search spaces. Contributed Talk at Surrogate-Assisted Multi-Criteria Optimization (SAMCO) Workshop, Lorentz Center, Leiden, NL, (2016)

Martina Friese, Thomas Bartz-Beielstein, Michael Emmerich, Building ensembles of surrogates by optimal convex combination. In *Proceedings of Bioinspired Optimization Methods and their Applications* , BIOMA 2016, Gregor Papa and Marjan Mernik (editors), Jožef Stefan Insitute, Ljubljana, Slovenia, pg.131-143 (2016)

Chapter 5
Martina Friese, Thomas Bartz-Beielstein, Thomas Bäck, Michael Emmerich, Weighted Ensembles in Model-based Global Optimization. In *AIP Conference Proceedings of LeGO 2018 - Int. Workshop on Global Optimization*, Leiden, The Netherlands, September 18-21, 2018. AIP Web of Science,pg.020003 (2019)

Martina Friese, Thomas Bartz-Beielstein, Thomas Bäck, Michael Emmerich, Optimally Weighted Ensembles of Surrogate Models for Sequential Parameter Optimization, Journal of Global Optimization, Special Issue LeGO Workshop 2019, (submitted for)

Chapter 6
Jörg Stork, Martina Friese, Martin Zaefferer, Thomas Bartz-Beielstein, Andreas Fischbach, Beate Breiderhoff, Tea Tušar, and Boris Naujoks. Open issues in surrogate-assisted optimization. In *High-Performance Simulation-Based Optimization*, Bookchapter, Thomas Bartz-Beielstein, Bogdan Filipič, Peter Korošec,

El-Ghazali Talbi (editors), pg.225-244, Springer (2020)

Chapter 2

Preliminaries

In the following, all tools and basics are introduced that build the fundamentals of this thesis. This chapter is structured as follows: In Section 2.1 the general concept of a surrogate model is introduced as well as the concrete models used in this thesis. In Section 2.2 SPO in general is introduced as well as the SPO framework that was specially developed for the experiments carried out in this work. Finally, in Section 2.3 an overview of the different objective functions is given.

2.1 Surrogate Modeling

The term surrogate model denotes a function $\hat{y} : \mathbb{R}^d \to \mathbb{R}$ which is an approximation of the true objective function[1] $y : \mathbb{R}^d \to \mathbb{R}$, learned from a finite set of evaluations of the objective function. Surrogate models are used, when the objective function is not known, and a complete evaluation is not feasible since they approximate the behavior of the objective function and are therefore cheaper and faster respectively to evaluate.

For the experiments carried out in this thesis a large set of heterogeneous surrogate models is used. To facilitate the choice while at the same time ensuring that the resulting set of surrogate models is most diverse, all models that are available as part of the SPOT package [21] are included. Since this work also

[1] Typically continuous functions are considered, but there are exceptions. [19, 20]

aims at handling unknown models, all models, except for the Kriging based models, are plugged into the system using default settings as provided by the SPOT interfaces. For the Kriging models, the correlation function is specified via the settings. Models that would not run on default settings or fail during experiments are discarded. The surrogate models that are used for the experiments are briefly introduced in the following sections.

2.1.1 Linear Regression Based Models

Linear regression models are the simplest approach to modeling data. A thorough introduction to linear regression models and related topics is given in [22, 23], which the following introduction is also based on.
For a start, we consider an easy example, where the relationship between two variables is to be modeled, i.e., income and years of education. Assuming that the relationship between those variables is of a linear nature, a model of the form

$$y = \beta_0 + \beta_1 x + \varepsilon$$

can be used to model the data. Predictions based on this model will be denoted as $\hat{y} = \beta_0 + \beta_1 x$. Here, y is the response variable, for a given x value, or predictor variable. β_0 and β_1 are model coefficients that represent the intercept and slope of the linear model, and ε denotes the model error. In order to gain the model, these parameters have to be estimated. This can, for example, be done by minimizing the least squares criterion. To do this, we consider the prediction errors $r_i = y_i - \hat{y}_i$, or residuals, between the observed response and the predicted response from the model. Figure 2.1 shows an example fit of a linear regression model and the resulting prediction errors.

Using the least squares criterion, the sum of squared prediction errors has to be minimized:

$$\min_{\beta_0,\beta_1} \sum_{i=1}^{n} (r_i)^2 = \min_{\beta_0,\beta_1} \sum_{i=1}^{n} (\beta_0 + \beta_1 x_i - y_i)^2 \ .$$

This can be calculated as a simple mathematical optimization problem [22, p. 62]. Given that more than one predictor variable has to be considered, the linear regression model would take the form

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon \ ,$$

where n is the number of predictor variables and β_i refers to the i-th predictor variable.

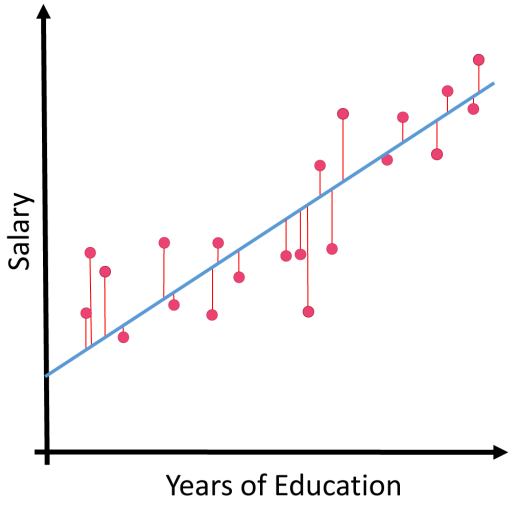


Figure 2.1: Example for a fit of a linear model. The blue line represents the linear model, the red dots the known data and the red lines between them depict the residuals or error made by the prediction for each known point. The model tries to find a linear relationship between the data that minimizes the sum of squared errors and thus obtains a straight line that models the data best under the assumption that the underlying relationship is linear.

However, this model, as specified so far, is based on the strict assumption that the relation between the predictor variables and the response variable is additive and linear [22].

Extensions to the linear model enable a relaxation of these assumptions by allowing interaction terms, i.e.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \varepsilon$$

or quadratic terms, i.e.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \varepsilon,$$

also.

The experiments carried out in this thesis also use a linear model ('LM') with a response-surface component, based on the `rsm` package [24], which again is an extension to the R core functionality `stats::LM`. This model allows for interaction terms and quadratic terms also, if the number of predictor variables is sufficient.

2.1.2 Tree Based Models

Classification and Regression trees were first introduced by Breiman et al. [25]. James et al. also give a thorough and comprehensible introduction to both types of trees [22]. Later, Breiman et al. introduced a method to combine a large number of trees into one more accurate prediction model, known as random forest [26].

Tree-based models, as many other models also, can be used for classification as well as for regression. In the case of classification, a binary tree is learned, where nodes and adjoint labeled branches represent decisions while their terminal nodes, which will be referred to as leaves, represent the class that the data is assigned to. Figure 2.2 shows an intuitive example for such a classification tree, given by

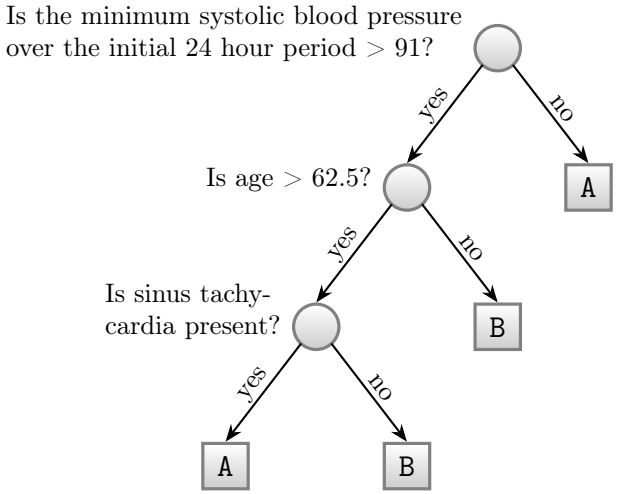
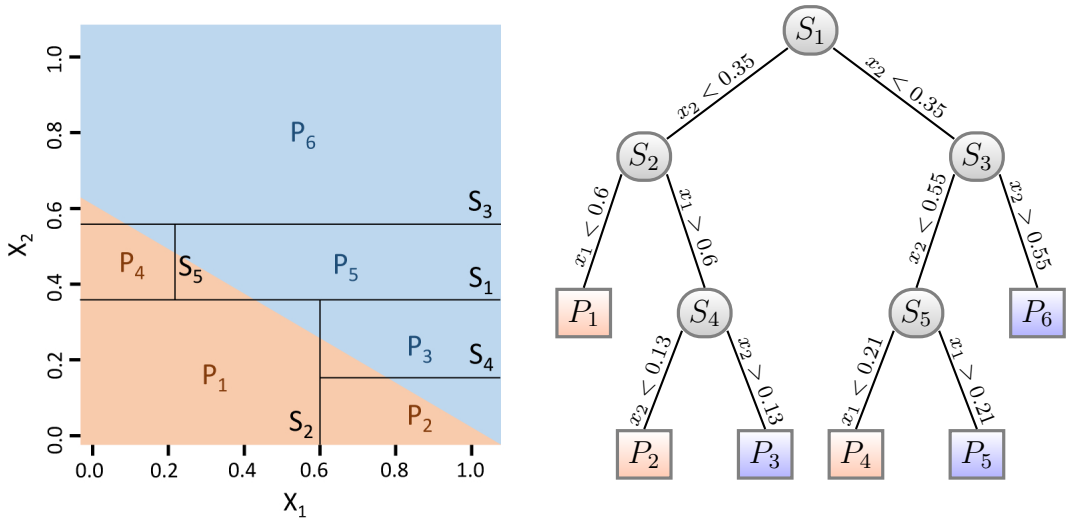


Figure 2.2: Classification tree for the identification of high-risk heart attack patients. Based on [25]. Each internal node marks a split based on the decision to be made for that node, here the related label shows this decision. The terminal nodes, or leaves, represent the class of the observations that fall here. In this example 'A' denotes low-risk patients whereas 'B' represents high-risk patients.

Breiman et al. [25]. The tree shown here was created during a study that analyzed data of a medical study on heart attacks to identify high-risk patients. During this study, 19 variables were measured in the first 24 hours after admission to the medical center. The tree uses only three of the variables to effectively classify low-risk patients 'A' and high-risk patients 'B'.
To generate such a classification tree, for each set of data the one variable is

searched for, that separates the two classes best. In general, the process is repeated for each branch until the part of data on this path is separated or when splitting no longer improves the splitting result and with such, the prediction. However, other rules also may be applied.

In the case of regression also a binary tree is learned, but the tree does not classify the data in that sense but instead partitions the parameter space of the data such, that every partition, represented by a leaf, contains data that is similar or somehow close to each other, and the prediction error is minimized. Here, the nodes and adjoint labeled branches define the partitions of the data.



(a) Depicted are the partitions generated by a regression tree on a mostly flat, two-dimensional function. The colors indicate the underlying function.

(b) An exemplary regression tree that corresponds to the partitions shown in (a). The colors of the leaves indicate the class that the observed data is classified to.

Figure 2.3: The figures show an example of a regression process on a two-dimensional function. Figure (a) shows the partitions constructed by the regression tree, while Figure (b) depicts the related tree.

Figure 2.3 gives an example of how such a regression using a regression tree may work. Given a two-dimensional function that mainly defines two plane areas separated by a diagonal. Figure 2.3a shows the regarded function, indicated by the colors. The regression tree may partition the parameter space by splitting at $S_1 : x_2 = 0.35$ first. The predictions for the upper partition may have improved, while the prediction quality for the lower part may not have changed or got even worse. So in the next step, the related partition is split $S_2 : x_1 = 0.6$, and so on.

A few steps later the regression tree may have partitioned the parameter space as shown in Figure 2.3a. The regression tree that was constructed, and relates to the partitions created during this process looks like depicted in Figure 2.3b. Each internal node with adjoint labeled branches represents the Splits S_1 - S_5 specified during the partitioning of the data. The leaves specify the prediction that will be made for observations in the related partition P_i, $i \in \{1, 2, \cdots 6\}$.

In order to build a random forest, the available data is bootstrapped so that a set of distinct training data sets is generated. On each of the training data sets generated this way, a single tree is learned. However, this alone does not ensure a high variance in the trees generated. Assumed that in the available parameters one turns out to be a very strong predictor while several others only are moderately strong, the trees would always tend to use this very strong predictor. To counteract on this behavior, another tweak is added to the tree generating process, that enables the tree to choose from a randomly selected subset of predictors only, to generate the next junction in the tree. These subsets are generated randomly for each junction.
To gain a single prediction from the set of trees generated this way, the predictions of all trees are aggregated using majority voting, in the case of classification, and averaging, in the case of regression, respectively.

For the experiments carried out for this thesis two different kinds of tree-based models are used. These are for one a single tree model ('Tree'), that is based on the `rpart` package [27] and in most details follows Breiman et. al [25] quite closely.
And for another, a random forest model ('RandomForest'), which is based on the `randomForest` package [28], and implements the random forest algorithm by Breiman et al. [29] for regression.

2.1.3 Artificial Neural Networks

Artificial neural networks, also referred to as neural networks, describe information processing systems that are composed of units or nodes respectively, that communicate their information over directed connections between each other. Zell [30] and Anderson et al. [31] give an exhaustive overview of artificial neural networks and its history.
The general idea is derived from the cerebral structure of mammals, rigorously simplified. Different types of bipolar and multipolar cells of mammals are depicted in Figure 2.4. All of them feature the same components of cell body, axon,

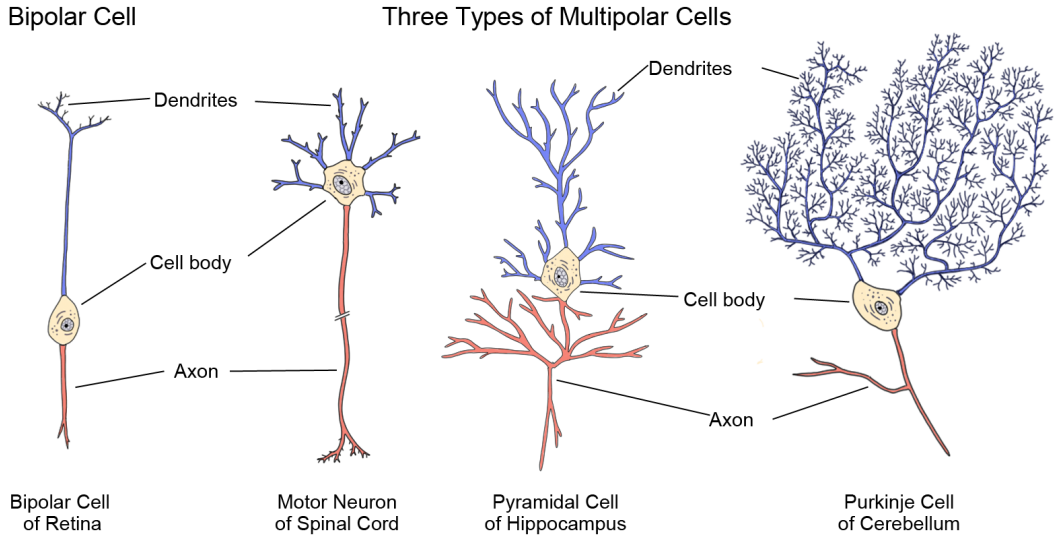


Figure 2.4: The illustration shows examples for different types of neurons (based on [32, 33])

and dendrites. The cell body ensures the energy supply of the cell and has the ability to process and transmit nerve impulses. With the dendrites, incoming impulses are received, with the axon the impulses are transmitted to other cells in the target area.

In order to build an artificial neural network, the neurons are vigorously abstracted. Figure 2.5 shows a schematic representation of two cells of a neural network. The cells i and j both have a network in_i and in_j of incoming connec-

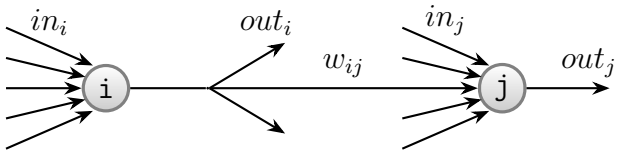


Figure 2.5: Depicted is a schematic representation of two simplified cells of a neural network (based on [30]). The incoming connections *in* represent the dendrites, the outgoing connections *out* the axons. The connection strength between the two cells is specified by a weight w.

tions representing the Dendrites, and a network of outgoing connections out_i and out_j representing the Axons. A weight w_{ij} specifies the transmission strength of the connection from the cell i to the cell j. With such cells, a network is built

that in many cases is arranged in layers with connections heading in the direction of the output neurons only, also referred to as feed-forward networks. Figure 2.6 shows an example of such a network.
This network has three layers of trainable connections, in this particular case every cell of one layer is connected to every cell of the next layer, and four layers of cells with the cells in the lower layer being the input cells and the cells in the upper layer the output cells. The inner layers are referred to as hidden layers.

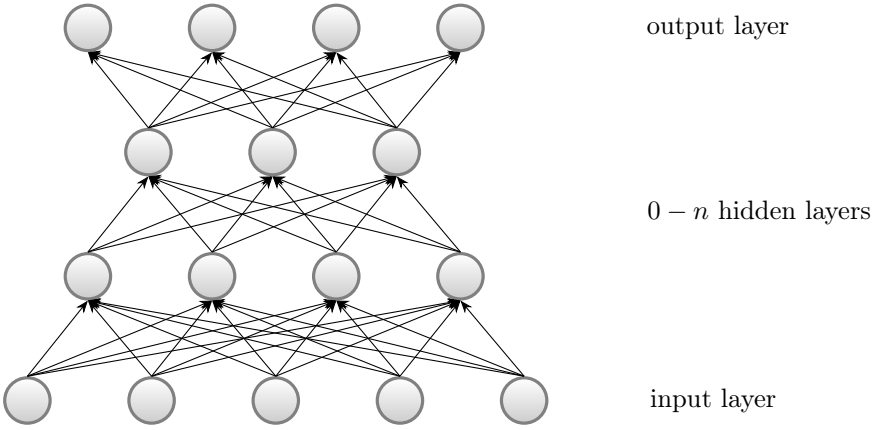


Figure 2.6: Depicted is a schematic feed-forward network with three connection layers and four neuron layers (based on [30]). The four cells in the bottom row represent the input layer; the four neurons in the topmost row the output layer. The inner layers are so-called hidden layers, and their number may strongly vary. Here, every cell of one layer is connected to every cell of the next layer.

For the experiments carried out in this thesis from the neural network type of models three different neural networks are used. These are a neural network ('neuralnet') based on the `neuralnet` package [34]. It is trained using resilient backpropagation without weight backtracking. The model is instantiated with two hidden neurons in each layer and a threshold of 0.001 that corresponds to the defaults that are set by the SPOT interface and thus slightly differs from the default provided by the `neuralnet` package.

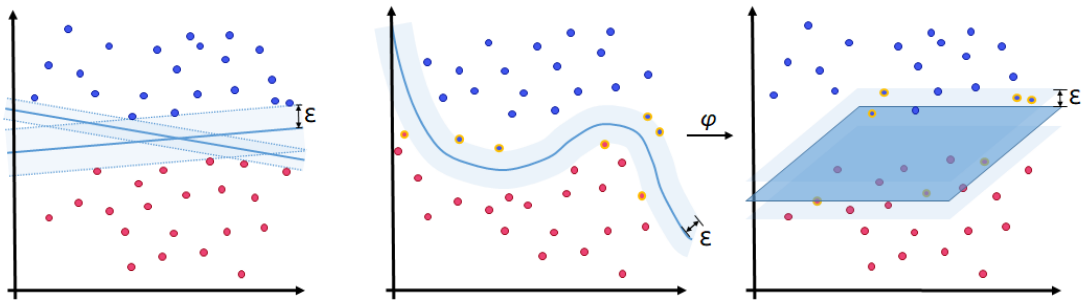
Additionally, an ensemble of twenty multi-layer perceptron neural network models ('MLP') based on the `monmlp` package [35] is used. Each is using two hidden-layers. The implementation applies early stopping together with bootstrap aggregation to control overfitting. The SPOT interface invokes the neural net using two hidden neurons in the first and one in the second layer.

And lastly, a censored quantile regression neural network model ('Qrnn') based on the `qrnn` package [36]. The model is instantiated with a single layer of hidden nodes.

2.1.4 Support Vector Machines for Regression

Support Vector Machines (SVM) implement concepts for regression as well as for classification. They were introduced by Boser et al. in 1992 [37]. A detailed introduction to SVMs can be found in [38], a comprehensive introduction to SVMs for Regression is given in [39, 40].

Here, data points are regarded as vectors in space. In the case of classification, the algorithm constructs a hyperplane that is separating the known vectors optimally. In this case, optimally means, that the distance ε of the input vectors that are closest to this separating hyperplane, also referred to as support vectors, is maximized. Figure 2.7a depicts such a situation where a set of vectors representing two different classes have to be separated. The vectors are directly linearly separable. Two possible separating lines are shown, the line with the broader separation space allows for the largest possible separation distance ε and therefore is the best choice for this example.



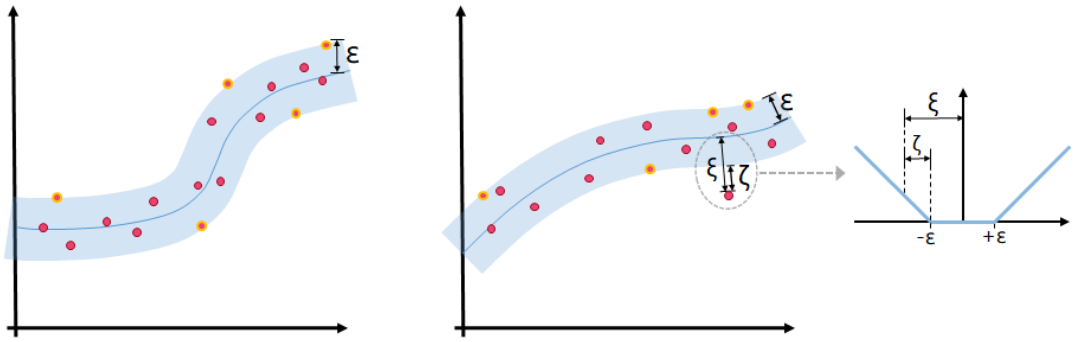
(a) If the data is linearly separable, the SVM fits a hyperplane through the data that separates the two classes with the largest possible distance ε to the nearest points.

(b) The data that has to be classified is not always linearly separable. In such cases, a kernel transformation is applied to the data that maps the data into a higher dimensional feature space, where the data is linearly separable. The points that are marked yellow are the support vectors that lie closest to the separating hyperplane.

Figure 2.7: Classification of two types of data points using a SVM.

However, the input vectors $\mathbf{x}$ are not always linearly separable. If this is the case, they are mapped into a high dimensional feature space Z. This is achieved by using some nonlinear mapping φ, that has been chosen beforehand. In this space, an optimal separating hyperplane is constructed. Figure 2.7b shows such a situation.

In the case of SVMs for regression, a function $f(x)$ is searched for, which approximates the known vectors with the smallest maximum deviation ε for all training vectors from the function $f(x)$ while also being as flat as possible. Here, too, the points that lie on the ε-deviation border are referred to as support vectors. Figure 2.8a illustrates such a function that approximates a set of training vectors with an accepted error of ε.



(a) SVMs for regression aim to find a function that approximates the training data, while minimizing the maximum deviation ε between training data and the function.

(b) Error handling using a soft margin penalty. Points with a deviation smaller than ε don't add to the cost of the function. Points with a larger deviation than ε are penalized linearly with its distance ζ to the ε-border.

Figure 2.8: Modelling data using Support Vector Machines for Regression

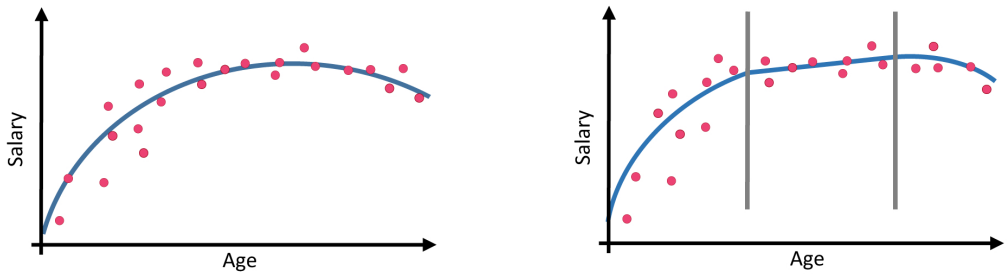
But this procedure is not strict, SVMs also allow for larger deviations in a few points using a soft margin penalty. For that purpose, the trade-off between the flatness of the function and number or cost of errors respectively that is accepted may be adjusted to allow for a few outliers. The outliers are then penalized with a linear cost function that assigns the cost ζ to an outlier that has a deviation of ξ with $\zeta = |\xi| - \varepsilon$. Figure 2.8b displays an example for such a soft margin penalty.

For the experiments carried out in this thesis a regression model based on the R

package `e1071` [41], that builds a support vector machine for regression is also used ('Esvm'). The model is based on the works of Chang et al. [42].

2.1.5 Multivariate Adaptive Regression Spline Models

Multivariate Adaptive Regression Splines (MARS) can be understood as a generalization of the recursive partitioning strategy used by regression trees [43].



(a) The figure shows an example for polynomial regression on salary data using a single polynomial

(b) Depicted is an example of piecewise polynomial regression on salary data using three independent polynomials. The vertical lines mark the so-called 'knots' that define the borders of the partitions.

Figure 2.9: The Figures show two approaches to approximating a function that describes best the relation between age and salary. The red points mark the observed points, and the blue line represents the approximated function.

Supposed that a function is to be learned that estimates the salary of a person with respect to the age of this person best. A simple method would be to fit a polynomial to the data using least squares regression. Figure 2.9a depicts an example of this approach. The red points mark the observed data concerning this relation.

In many cases, this may lead to good solutions, but there will also be cases where the data cannot be adequately approximated using a single polynomial. Then, it may be beneficial to divide the parameter range into disjoint subsets and fit an independent polynomial for each of the ranges. To obtain K ranges, $K-1$ so-called 'knots' have to be specified, that define these ranges. Furthermore, it

has to be ensured, that the resulting piecewise approximation function obtained is continuous or even smooth. For this purpose, additional constraints are introduced for each knot. Figure 2.9b shows an example of such a piecewise regression. The obtained function is continuous at the knots, but not smooth.

In opposition to polynomial regression, which must use polynomials of a higher degree to fit functions of higher complexity, regression splines allow keeping the degree of the polynomials fixed and induce higher flexibility through a higher number of knots. An exhaustive introduction to Multivariate Adaptive Regression Splines is given by Friedman et al. [43].

For the experiments carried out in this thesis also a regression model ('Earth') that uses the model building techniques of Friedman et al. for Multivariate Adaptive Regression Splines [44] and 'Fast MARS' [45] based on the R package `earth` [46] is added to the set of base models.

2.1.6 Kriging Based Models

The Kriging method (also known as Gaussian processes [47]) was first introduced by D.G. Krige [48] to improve mine valuation methods and with it the estimation of the concentration of gold in ore bodies. In 1963 Matheron extended the theory and formalized the technique [49]. Sacks et al. later applied the method to the approximation of computer experiments [50]. Forrester et al. give a very comprehensive introduction to optimization using the Kriging method [40]. The following remarks and equations are also based on these works.
The Kriging method uses a model of the form

$$\hat{y}(\mathbf{x}^*) = \hat{\mu} + \mathbf{k}^T\mathbf{K}^{-1}(y - \mathbf{1}\hat{\mu}). \tag{2.1}$$

as predictor. If μ is an a priori given constant, the Kriging variant is called 'simple' Kriging. If μ is estimated from the data, but then used as a constant, the variant is called 'ordinary' Kriging. Moreover, if μ is estimated from the data and depends on $\mathbf{x}$ then the variant is called 'universal' Kriging. In the following 'ordinary' Kriging is used since it has an excellent prediction quality and not too many parameters.
To derive this model it has to be started from a set of known data, $\mathbf{X} = \{\mathbf{x}^1, \mathbf{x}^2, \cdots \mathbf{x}^n\}^T$, with related known function values $\mathbf{y} = \{\mathbf{y}^1, \mathbf{y}^2, \cdots \mathbf{y}^n\}^T$ and it is searched for an expression to predict a value at an unknown point $\mathbf{x}$. Kriging views the known function values $\mathbf{y}$ as if they are realizations of a stochastic process, with errors ϵ spatially correlated. Between two points that are close

to one another the errors are considered positively correlated. With increasing distance between these points the correlation converges to zero. This correlation between the distance of the points and their errors are modeled using a correlation function, also referred to as kernel. An example for such a kernel is given by Equation (2.2).

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\sum_{i=1}^{m} \theta_i |x_i - x_i'|^{p_i}\right) \tag{2.2}$$

Here, two points x and x' are considered, with $x_i \in \mathbb{R}$ denoting the i-th element of the vector x. This kernel meets the requirements to yield a function value of one if $\mathbf{x} = \mathbf{x}'$, converges to zero for larger distances and is positive semi-definite. Some examples of different kernel functions are shown in Figure 2.10. The kernel function is utilized to create a correlation matrix $\mathbf{K}$ containing all pairwise correlations between errors of all known function values $\mathbf{y}$ at locations $\mathbf{X}$.

$$\mathbf{K} = \begin{pmatrix} k(\mathbf{x}^1, \mathbf{x}^1) & \cdots & k(\mathbf{x}^1, \mathbf{x}^n) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}^n, \mathbf{x}^1) & \cdots & k(\mathbf{x}^n, \mathbf{x}^n) \end{pmatrix}$$

These correlations depend on the absolute distance between the known points $|\mathbf{x} - \mathbf{x}'|$ and the parameters θ_i and p_i. These parameters, θ and p, are in general estimated using Maximum Likelihood Estimation [40, 51], such that for the given model, the known data points have the largest likelihood. The likelihood function of the Kriging model is based on the probability density function of a multivariate normal distribution,

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}|\mathbf{C}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{y} - \mathbf{1}\mu)^T \mathbf{C}^{-1} (\mathbf{y} - \mathbf{1}\mu)\right)$$

with $\mathbf{1}$ being the identity vector and $\mathbf{C}$ the stationary covariance matrix. The covariance matrix $\mathbf{C}$ is related to the correlation matrix $\mathbf{K}$ by $\mathbf{C} = \sigma^2 \mathbf{K}$. With this the Kriging likelihood function can be expressed as

$$f(\epsilon(\mathbf{X})|\mu, \sigma, \theta, p) = \frac{1}{(2\pi\sigma^2)^{n/2}|\mathbf{K}|^{1/2}} \exp\left(-\frac{(\mathbf{y} - \mathbf{1}\mu)^T \mathbf{K}^{-1} (\mathbf{y} - \mathbf{1}\mu)}{2\sigma^2}\right) \tag{2.3}$$

This equation has to be simplified by taking the natural logarithm. The partial derivatives of this function then have to be set to zero in order to obtain the Maximum Likelihood Estimates for μ and σ^2:

$$\hat{\mu} = \frac{\mathbf{1}^T \mathbf{K}^{-1} \mathbf{y}}{\mathbf{1}^T \mathbf{K}^{-1} \mathbf{1}}$$

and

$$\hat{\sigma}^2 = \frac{(\mathbf{y} - \mathbf{1}\hat{\mu})^T \mathbf{K}^{-1} (\mathbf{y} - \mathbf{1}\hat{\mu})}{n}.$$

These terms for $\hat{\mu}$ and $\hat{\sigma}^2$ can now be substituted back into the logarithmized form of Equation 2.3. Removing constant terms yields the so-called concentrated ln-likelihood function:

$$con(ln(L)) = -\frac{n}{2}\ln(\hat{\sigma}^2) - \frac{1}{2}\ln(|\mathbf{K}|). \tag{2.4}$$

Here, the parameter θ and p are still unknown. In order to find values for these parameters that maximize the function, numerical optimization has to be applied, since the function cannot be differentiated. However, as the function is quick to compute, as long as the search space does not get too large, the function can be searched directly, so that a classical global optimization method can be applied.

Introductory it was said that it is searched for an expression to predict a value $\hat{y}$ at an unknown point $\mathbf{x}$. The main idea to do so is to augment the known data $\mathbf{y}$ with the new prediction $\hat{y}$, which yields the vector $\mathbf{y}_{aug} = \mathbf{y}^T, \hat{y}^T$. Treating $\hat{y}$ as a model parameter, the likelihood function is to be maximized with respect to $\hat{y}$, given the already known correlation parameters.
Defining the vector of correlations between the set of known data $\mathbf{X}$ and the new data point x^* as $\mathbf{k} = (k(x^1, x^*), \cdots, k(x^n, x^*))^T$, this vector can be substituted into the prediction function (2.1) together with the already known model parameters to make a prediction.

For a next iteration, the augmented correlation matrix can be defined as $\mathbf{K}_{aug} = \begin{pmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & 1 \end{pmatrix}$. Together with the already known model parameters, $\mathbf{K}_{aug}$ and $\mathbf{y}_{aug}$ are substituted into the likelihood function (2.3). Maximizing the resulting term with respect to $\hat{y}$ yields the predictor (2.1).

In the experiments carried out in this thesis, three Kriging models with different correlation functions are used. These are Kriging with exponential correlation function ('correxp'), gaussian correlation function ('corrgauss'), and spline correlation function ('corrspline'). Figure 2.10 depicts the kernels used. The Kriging implementation is part of the `SPOT` package and follows the implementation of Lophaven et al. as described in [52].

Following the definitions from Lophaven et al., the correlation models can be described as follows. We consider stationary correlations of the form $\mathcal{R}(\theta, x, x') = \prod_{j=1}^{n} \mathcal{R}(\theta_j, x_j - x'_j)$. The first model uses the *exponential* kernel $\mathcal{R}(\theta, x_j, x'_j) =$

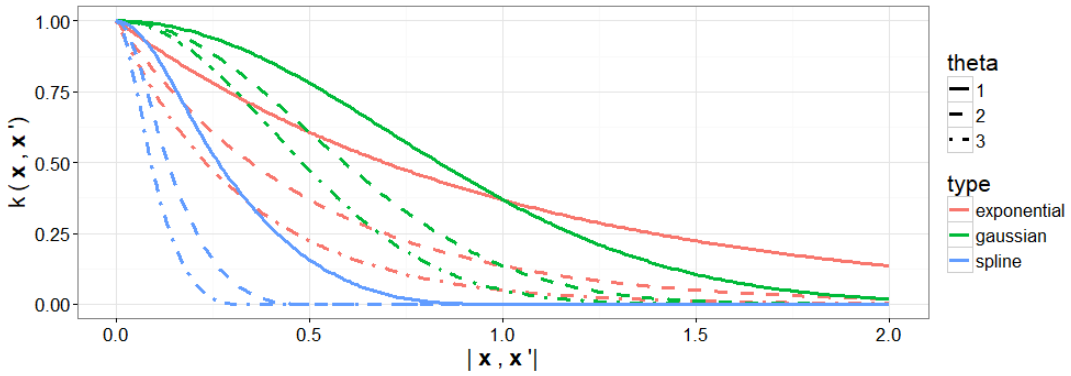


Figure 2.10: The figure shows an example of different correlation functions. The functions 'exponential' and 'gaussian' implement the correlation function specified in Equation 2.2 with $p = 1$ for 'exponential' and $p = 2$ for 'gaussian'. The 'spline' function follows the definitions from [52] and only takes the θ-parameter. The function values then specify the assumed correlation of two known points, based on their distance.

$\exp(-\theta_j|x_j - x'_j|)$ the second model uses a *gaussian* kernel $\mathcal{R}(\theta, x_j, x'_j) = \exp(-\theta_j|x_j - x'_j|^2)$, whereas the third model is based on the *spline correlation* function $\mathcal{R}(\theta, x_j, x'_j) = \zeta(\theta_j|x_j - x'_j|)$ with

$$\zeta(\epsilon_j) = \begin{cases} 1 - 15\epsilon_j^2 + 30\epsilon_j^3 & \text{for} \quad 0 \leq \epsilon_j \leq 0.2 \\ 1.25(1 - \epsilon_j)^3 & \text{for} \quad 0.2 < \epsilon_j < 1 \\ 0 & \text{for} \quad \epsilon_j \geq 1. \end{cases}$$

Here, ϵ and θ are hyperparameters estimated by likelihood maximization.

Additionally, a treed Gaussian process model with jumps to the limiting linear model ('tgp') that is based on the `tgp` package [53] is used in the experiments.

Finally, a simple ensemble model ('Rfmlegp') is also used for the experiments. The ensemble internally builds a random forest model using the `randomForest` package [28] and a Gaussian process model using the `mlegp` package [54]. The mean of these models' predictions is returned as the ensemble prediction.

2.2 Surrogate Model-Based Optimization

In most real-world optimization problems, the number of function evaluations that can be carried out is massively limited by time or cost. At the same time, not seldom the objective function is expensive to evaluate. Direct search methods, in general, require more function evaluations than the number that can actually be spent, which makes such real-world problems a special challenge for global optimization.

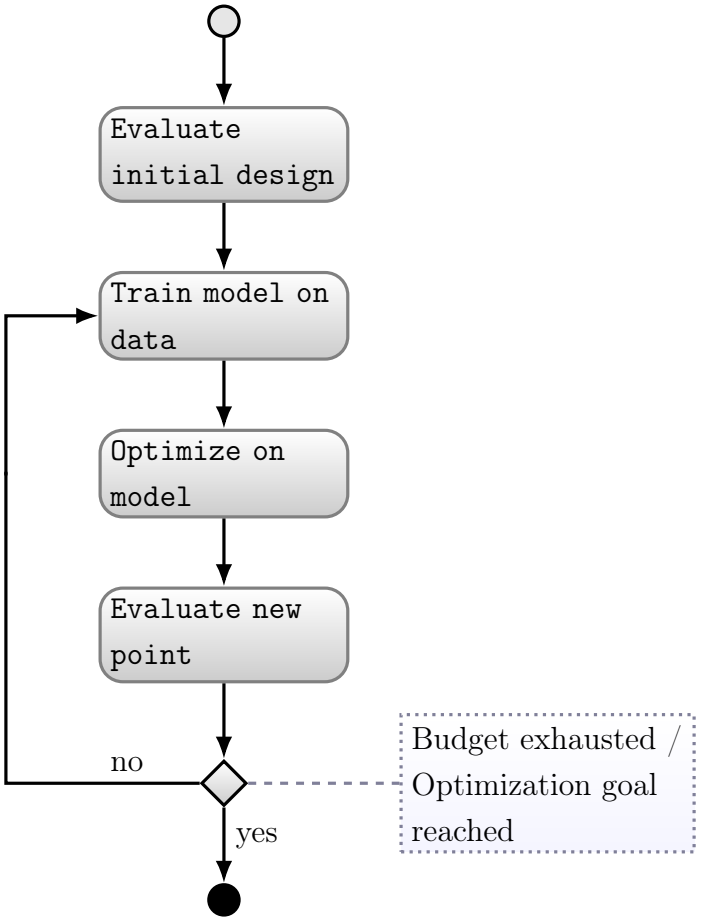


Figure 2.11: Shown is a schematic representation of a surrogate model based optimization process. Initially, a first set of data points is evaluated on the function; in general, these points are chosen using a design of experiment. Then, a surrogate model is fitted to the data. On the fitted model, a separate optimization process is started, and the best points found are evaluated on the function. The process ends when the predefined stopping criterium is reached.

Jones et al. [17] and others [14, 55, 56] addressed this challenge by using surrogate models as a fast and cheap approximation of the real objective function. Instead of performing a direct search on the objective function, in every step a model is trained to the available data, and an optimization process is carried out on this model. The best n solutions found during this intermediate optimization step are then evaluated on the real objective function. Figure 2.11 depicts the general procedure of such an SBO process. Evolutionary surrogate model based optimization was introduced by Emmerich et al. [57].

Like in any optimization process, if initially no information about the function is available, some data points have to be evaluated to gain some base knowledge D about the function. Usually, a design of experiment (DOE) is used to obtain this knowledge.

Then, the main optimization loop is started by fitting the model to the observed data D. On the fitted model that is now approximating the objective function a separate optimization step is carried out.

From the set of points that were evaluated on the model during this optimization, a choice of n points is selected to be evaluated on the objective function. This choice is not limited to merely choosing the point that has the best-predicted value on the model, but can also be based on other criteria like the expected improvement[1] if the model provides confidence intervals for its predictions. The main optimization loop is stopped if the budget of available function evaluations is exhausted or a predefined optimization goal is reached.

Software packages like SPOT provide a complete optimization framework featuring heaps of statistical tweaks to optimize this process even more, as well as parameters for additional fine-tuning [59].

Nonetheless, for the experiments carried out in this work a very basic SPO framework is developed to allow for complete control over the behavior of the sequential optimization process and to gain more insight and interpretable results. Algorithm 1 depicts the main steps of this framework which, up to some minor details, corresponds to standard SPO processes.

In a first step the initial design is evaluated (cf. Algorithm 1 line 2). To allow for reproducible results and equal experimental preconditions across several experimental setups, the framework takes predefined experimental designs so that

[1] Expected improvement is a criterion that helps to balance between exploitation and exploration. It was introduced by Mockus et al. in 1978 [58]. Jones et al. [17] give a comprehensive introduction to the topic.

Algorithm 1: The SPO Framework that was used for the experiments carried out in this thesis

Data: objective function $f : \mathbb{R}^d \to \mathbb{R}$
initial design $\{x_1, \cdots, x_n\}$, $x \in \mathbb{R}^d$,
surrogate model m
sequential step size n_{eval}

Result: Resultset containing all evaluated points

1: **begin**
2: $D \leftarrow$ Evaluate initial design on objective function f;
3: **while** *function evaluations available* **do**
4: $m^* \leftarrow$ Fit model m to known datapoints D ;
5: Evaluate sequential design on model m^*;
6: Choose $\lceil \frac{1}{2} n_{\text{eval}} \rceil$ points x^* according to exploitation;
7: Choose $\lfloor \frac{1}{2} n_{\text{eval}} \rfloor$ points x^{**} according to exploration;
8: Evaluate chosen points x^* and x^{**} on objective function f;
9: Add new information to known data D ;

all experiments use the same initial configuration (cf. Algorithm 1, data input). The resulting dataset $D = \{(x_1, y(x_1)), \ldots, (x_n, y(x_n))\}$ builds the foundation for the subsequent optimization process.

The main optimization steps are carried out in a loop that ends when the number of allowed function evaluations has been reached (cf. Algorithm 1 line 3).

First, the model M is fitted to the observed data D (cf. Algorithm 1 line 4). On the fitted model M^* a sequential design is evaluated (cf. Algorithm 1 line 5). Based on the resulting set of model predictions, points are selected for evaluation on the objective function. For this choice, the available budget of function evaluations per sequential step n_{eval} is shared equally on points x^* that correspond to the criteria of exploitation[1] and on points x^{**} that correspond to the criteria of exploration[2].

[1] Exploitation refers to the strategy of searching a restricted search space in the area of the best-known solution in order to improve this solution. This can also be referred to as local search.

[2] Exploration refers to a strategy of searching the entire region of interest in the search space in order to find new promising solutions. This strategy helps to diversify the search and to

The selection of these points is carried out as follows:
For exploitation from the predictions of the model on the large sequential design, the k best performing points are chosen and added to the candidate set C^*. A typical value of k is 20. Since these points are restricted to points from the design, from each of these points a local search is initiated on the model. The points that are obtained with this search are also added to the candidate set C^*. If these points violate constraints of the search space, they are repaired by setting those violated parameters to the allowed limit. From this set of candidates C^* the $\lceil \frac{1}{2} n_{\text{eval}} \rceil$ points x^* that perform best on the model are chosen for evaluation on the objective function (cf. Algorithm 1 line 6).
For exploration, those k points from the sequential design are chosen and added to the candidate set C^{**}, that have the largest distance to their nearest neighbors. From these candidates those $\lfloor \frac{1}{2} n_{\text{eval}} \rfloor$ points x^{**} that perform best on the model are chosen to be evaluated on the objective function (cf. Algorithm 1 line 7).

The points x^* and x^{**} chosen before are evaluated on the objective function. And the new information gained in this step is added to the dataset D (cf. Algorithm 1 line 8-9).

After finishing the main loop, the dataset D is returned as the result.

The default settings for the experiments performed in this study were chosen as follows: for the sequential design, a size of 200 points and a sequential step size of $fe_{step} = 2$ was chosen.

This optimization is intentionally kept simple to allow for better insight into the performance of the models used for optimization.

2.3 Objective Functions

The experiments carried out for this work are run on a set of objective functions. These objective functions can be roughly divided into three groups.

1. Generic objective functions generated to fit the requirements,
2. standard optimization test problems,
3. and test functions based on physical models.

prevent it from getting trapped in a local optimum. This can also be referred to as global search.

Since most of the black-box real-world problems considered to be difficult are multimodal, the focus for this work is also on multimodal function approximation (cf. [60, 61, 62]). In the following, these functions are introduced in more detail.

2.3.1 Gaussian Landscape Generator

To allow for the flexible generation of test functions that meet the requirements in certain features the *Max-Set of Gaussian Landscape Generator* (GLG) is applied. It computes the upper envelope of m weighted Gaussian process realizations and can be used to generate continuous, bound-constrained optimization problems [63].

Thus, a GLG objective function is defined as

$$G(x) = \max_{i \in 1,2,\ldots,m} (w_i g(x)),$$

where $g : \mathbb{R}^n \to \mathbb{R}$ denotes an n-dimensional Gaussian function

$$g(x) = \left(\frac{\exp\left(-\frac{1}{2}(x-\mu)\Sigma^{-1}(x-\mu)^T\right)}{(2\pi)^{n/2}|\Sigma|^{1/2}} \right)^{1/n},$$

μ is an n-dimensional vector of means, and Σ is an $(n \times n)$ covariance matrix. Implementation details are presented in [64]. For the generation of the objective functions the spotGlgCreate method of the SPOT package is used.

The options used for our experiments are shown in Table 2.1. With the parameter d the dimension of the objective function is specified. The lower and upper bounds (l and u, respectively) specify the region where the peaks are generated. The value max specifies the function value of the global optimum, while the maximum function value of all other peaks is limited by t, the ratio between the global and the local optima.

2.3.2 Optimization Test Problems

Two classical mathematical test problems are chosen. Both, the Ackley function [65] as well as the Rosenbrock function [66] are widely used for testing optimization algorithms.

Table 2.1: Gaussian landscape generator options

Param.	Description	Value
d	Dimension	$1-8$
m	Number of peaks	$10-320$
l	Lower bounds of the area, where peaks are generated	$\{0_1,\ldots,0_d\}$
u	Upper bounds of the area, where peaks are generated	$\{5_1,\ldots,5_d\}$
max	Max function value	100
t	Ratio between global and local optima	0.8

The Ackley function
was proposed by David Ackley in 1987 [65]. Later, in 1993 the function was generalized by Bäck and Schwefel [67]. In its generalized form it is defined by

$$f(x) = -a\,\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$$

with $a = 20$, $b = 0.2$ and $c = 2\pi$.

It is a non-convex function featuring many local optima. In its two-dimensional form the function is almost flat in its outer regions with the global minimum $(f(x^*) = 0, \text{at } x^* = (0,\ldots,0))$ being a large peak at the center. With its highly multi-modal characteristics, it poses a risk to optimization algorithms to get stuck in local minima.

As region of interest we regarded the hypercube $x_i \in [-32.768, 32.768],\ \forall x \in [1, d]$.

The Rosenbrock function
is also known as Valley- or Banana function and has been introduced by Howard H. Rosenbrock in 1960 [66]. It is defined by

$$f(x) = \sum_{i=1}^{d-1}\left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$$

The function is non-convex, but in contrast to the Ackley function, the Rosenbrock function is unimodal. Its global minimum $(f(x^*) = 0)$, lies at $x^* = (1,\ldots,1)$. In its two dimensional form it is located in a narrow, parabolic valley that is following the parabola $x_2 = x_1^2$. Though the valley is easy to find, convergence to the minimum is difficult.

As region of interest we regarded the hypercube $x_i \in [-2.048, 2.048],\ \forall x \in [1, d]$.

2.3.3 Physical Functions

In order to gain a set of test functions that also contains functions with relation to real-world optimization problems, four test functions that model a physical model are added to the set. As a source of these test functions, the 'Virtual Library of Simulation Experiments' [68] is used. The website provides a collection of well-structured test problems. From the choice of emulation/prediction test problems based on physical models, all functions, that are not stochastic and given in the form of a function instead of a dataset, are added to the system. Under this premise we get a set of four real-world objective functions as follows.

The OTL Circuit Function models an output transformerless push-pull circuit [69]. The response variable of the function is V_m, the midpoint voltage, that is affected by six parameters. The parameters, with its units and ranges, are specified in Table 2.2.

Parameter	Description	Value range
R_{b1}	resistance b1 (K-Ohms)	$[50, 150]$
R_{b2}	resistance b2 (K-Ohms)	$[25, 70]$
R_f	resistance f (K-Ohms)	$[0.5, 3]$
R_{c1}	resistance c1 (K-Ohms)	$[1.2, 2.5]$
R_{c2}	resistance c2 (K-Ohms)	$[0.25, 1.2]$
β	current gain (Amperes)	$[50, 300]$

Table 2.2: Search Parameters of the otl-circuit function

The midpoint voltage V_m can be derived from the parameters as follows:

$$V_m(\mathbf{x}) = \frac{(V_{b1} + 0.74)\beta(R_{c2} + 9)}{\beta(R_{c2} + 9) + R_f} + \frac{11.35R_f}{\beta(R_{c2} + 9) + R_f} + \frac{0.74R_f\beta(R_{c2} + 9)}{(\beta(R_{c2} + 9) + R_f)R_{c1}}$$

where

$$V_{b1} = \frac{12R_{b2}}{R_{b1} + R_{b2}} \qquad \text{and} \qquad \mathbf{x} = (R_{b1}, R_{b2}, R_f, R_{c1}, R_{c2}, \beta)$$

The Piston Function is a simulator, that models the movement of a piston within a cylinder. The piston consists of a linear rod that is connected to a disk. By this connection, the linear movement of the rod is transformed into a circular motion. The faster the piston moves within the cylinder the faster rotates also the disk. The performance of the piston is measured by its cycle time, the time it takes to perform one rotation of the disk, in seconds.

The performance of the piston is affected by a set of parameters, given in Table 2.3. These parameters affect the cycle time T_c via a chain of nonlinear equations [69]:

$$T_c(\mathbf{x}) = 2\pi\sqrt{\frac{M}{k + S^2\frac{P_0V_0}{T_0}\frac{T_0}{V^2}}}$$

where

$$V = \frac{S}{2k}\left(\sqrt{A^2 + 4k\frac{P_0V_0}{T_0}T_a} - A\right) \ , \qquad A = P_0S + 19.62M - \frac{kV_0}{S}$$

and

$$\mathbf{x} = (M, S, V_0, k, P_0, T_a, T_0).$$

The function was developed by Kenett and Zacks in 1998 [70].

Parameter	Description	Value range
M	piston weight (kg)	$[30, 60]$,
S	piston surface area (m^2)	$[0.005, 0.020]$
V_0	initial gas volume (m^3)	$[0.002, 0.010]$
k	spring coefficient (N/m)	$[1000, 5000]$
P_0	atmospheric pressure (N/m^2)	$[90000, 110000]$
T_a	ambient temperature (K)	$[290, 296]$
T_0	filling gas temperature (K)	$[340, 360]$

Table 2.3: Search Parameters of the Piston Function

The Robot Arm Function is commonly used in neural network literature. It models a four-segment robot arm with the shoulder of the arm fixed at the origin in the (u, v)-plane [71]. Each segment of the arm has a specific length L_i and an angle θ_i, $i = 1, \ldots, 4$. The angle of the first segment, with respect to the horizontal coordinate axis of the plane, is given by θ_1. The angles θ_2, θ_3, θ_4

describe the rotation of the corresponding arm segment in relation to the previous arm segment.

The response variable of the function D, models the distance of the end of the robot arm to the origin, on the (u, v)-plane. This distance can be directly derived from the eight parameters by:

$$D(\mathbf{x}) = \sqrt{u^2 + v^2}$$

where

$$u = \sum_{i=1}^{4} L_i \cos\left(\sum_{j=1}^{i} \theta_j\right), \quad v = \sum_{i=1}^{4} L_i \sin\left(\sum_{j=1}^{i} \theta_j\right) \text{ and } \quad \mathbf{x} = (L_1, L_2, L_3, L_4, \theta_1, \theta_2, \theta_3, \theta_4).$$

The input parameters, with its ranges, are specified in Table 2.4

Parameter	Description	Value range
$L_i,\ i = 1, \ldots, 4$	length of the i-th arm segment	$[0, 1]$
$\theta_i,\ i = 1, \ldots, 4$	angle of the i-th arm segment	$[0, 2\pi]$

Table 2.4: Search Parameters of the Robot Arm Function

The Wing Weight Function models the weight W of the wing of a light aircraft [40]. The analytical expression is adapted from the work of Raymer (2006) on conceptual aircraft design [72]:

$$W = 0.036\, S_W^{0.758}\, W_{fw}^{0.0035} \left(\frac{A}{\cos^2\Lambda}\right)^{0.6} q^{0.006} \lambda^{0.04} \left(\frac{100tc}{\cos\Lambda}\right)^{-0.3} (N_Z W_{dg})^{0.49} + S_W W_P$$

The ten parameters, that affect the weight W are given in Table 2.5, along with its baseline and range.

While the baseline values roughly represent the values of a Cessna C172 Skyhawk aircraft, the given ranges where specified by Forrester et al. [40].

Parameter	Description	Baseline	Value range
S_w	wing area (ft^2)	174	$[150, 200]$
W_{fw}	weight of fuel in the wing (lb)	252	$[220, 300]$
A	aspect ratio	7.52	$[6, 10]$
Λ	quarter-chord sweep (degrees)	0	$[-10, 10]$
q	dynamic pressure at cruise (lb/ft^2)	34	$[16, 45]$
λ	taper ratio	0.672	$[0.5, 1]$
t_c	aerofoil thickness to chord ratio	0.12	$[0.08, 0.18]$
N_z	ultimate load factor	3.8	$[2.5, 6]$
W_{dg}	flight design gross weight (lb)	2000	$[1700, 2500]$
W_p	paint weight (lb/ft^2)	0.064	$[0.025, 0.08]$

Table 2.5: Search Parameters of the Wing Weight Function

Some passages in this chapter are based on descriptions that were already published in [73, 74, 75].
The composition of base models and the set of objective functions used for the experiments carried out in this thesis were already described in [75]. The specification of the Kriging kernels was already given in [73]. Occasionally, text elements have been adopted verbatim from these publications. However, the text was significantly extended, and adapted to fit the notation of this thesis.
The SPO Framework introduced in this Chapter has already been described in [74, 75]. Some of these descriptions, as well as Algorithm 1, are adopted verbatim or with minor changes to adapt them to the notation and structure of this thesis.

Chapter 3

Taxonomy

In this Chapter, an introduction to the development in the field of surrogate modeling towards ensembles and a more detailed overview of ensemble methods is given. The functionality of the different approaches is analyzed, and a taxonomy of ensemble methods is derived from the observations made. The variety of existing approaches is large and more than a few are specifically designed for one problem or one class of problems. The main focus of this overview is laid on methods that conform to the goal of this work. These are mainly regression models, that can also be applied to Designs of Experiment (DOE) of a smaller size. Additional approaches that do not fit this specification are introduced afterward. Finally, the insights are discussed, and a conclusion is drawn specifying the primary necessities for the envisioned ensemble method.

Models are used when the direct evaluation of the actual fitness function would be too expensive. However, the performance of the available models on different objective functions is strongly varying. All models have their strengths and weaknesses which enable them to model some features better than others. Thus, whenever a model is to be applied, the result is strongly depending on the choice of the model. To choose the right model, not only knowledge about the objective function is needed, but also about the strengths and weaknesses of the available models. This knowledge is not always available, which led to different approaches to facilitate this choice by providing tools and criteria to evaluate or choose a model, methods to automatically select the most appropriate models or even to combine several models into one stronger model. But the different approaches are so diverse that it is not easy to obtain a broad overview.

In the following the different approaches are examined closer, differences and

similarities between the approaches are pointed out, and possible advantages and disadvantages are discussed.

The general question is: Given, that more than only one model is available, how to obtain one prediction from n models.
There are two criteria that allow for the first classification of these approaches. We can differentiate between Model Selection and Model Mixtures or Model combination. Also, we can distinguish by regarding the number of model fitting processes, since there are solutions that only fit a single model to the data and solutions that fit all models to the data.

3.1 Overview of Previous Developments and the State of the Art

As stated before, the choice of the surrogate model can have a significant influence on the solution quality and performance of a surrogate model based optimization process. Burnham et al. even stated that the choice of the right surrogate model is the most crucial question in making statistical inferences [18]. But in order to make meaningful decisions on which surrogate model to select for a given problem, often expert knowledge is needed. This includes knowledge about the objective function and the characteristics of the surrogate model likewise.

However, if there is no preliminary knowledge about the objective function or the available surrogate models, the choice has to be taken nonetheless. This may be done by just choosing the surrogate model that performed well on past optimization tasks. It would even be possible to switch between surrogate models during the optimization process randomly or applying a round robin method to give all available surrogate models their fair share if more predictions are needed sequentially [76]. Other more sophisticated methods might interpret the problem as a multi-armed bandit problem [77]. A well-known strategy is SoftMax, which uses a probability vector where each element represents the probability for a corresponding model to be chosen. The probability vector is updated depending on the reward received for the chosen models [78]. However, these ad hoc rules do not rely on the data to help select the best model and therefore ignore the principle of parsimony[1]. The principle of parsimony [80], also known as Ockham's

[1] Newton wrote in one of his books: "We are to admit no more causes of natural things than such as are both true and sufficient to explain their appearances." [79, p.731]

razor, describes the idea that, when multiple hypotheses are available to explain a thing, one should select the one with the fewest possible assumptions.

To overcome this problem, it would be beneficial if the algorithm could learn all by itself which surrogate model type suits the problem best, based on the given data. This can be done by evaluating different models on available training data and using a statistical model selection approach to select the most promising surrogate model [76].

But how to handle the situation when there is more than one strong model in the set? In such circumstances, it might be beneficial to combine inference output across several models. In statistics and machine learning an *ensemble* is a prediction model from several models, aiming for better accuracy.
Different approaches for building ensembles of surrogates are known. Bagging [29] combines results from randomly generated training data partitions whereas Boosting [81] combines several weak learners to a strong one in a stochastic setting. Weighted averaging approaches combine model predictions by calculating the mean or the median of different predictions [26]. But also operations like calculating the minimum or the maximum over these predictions may be thinkable for some applications [82].

Since imprecise models should not deteriorate the overall result, a weighting scheme is introduced. In [83, 84, 85] every model's result for a single design point is weighted using some criterion, i.e., Akaike's Information Criterion (AIC). The sum over all models yields the final value assigned to the design point. A similar approach is blending or stacking [86], where the weights are chosen in an additional training step. Polikar [87] named further ensemble methods.

3.2 Single Evaluation Model Selection

The first class of solutions to handle a large set of models for generating one prediction only evaluates the single models that have been selected. The diagram in Figure 3.1 displays the general flow of this approach. Whenever a single model comes to action, it has to be selected from the set of available models using some criterion. If the user selects the model by hand, this may be done by guessing (if there is no a priori knowledge available) or the model may be selected by preference (which for instance could be based on good experiences made during previous applications) and so on.

However, also automated model selection methods were proposed as an efficient

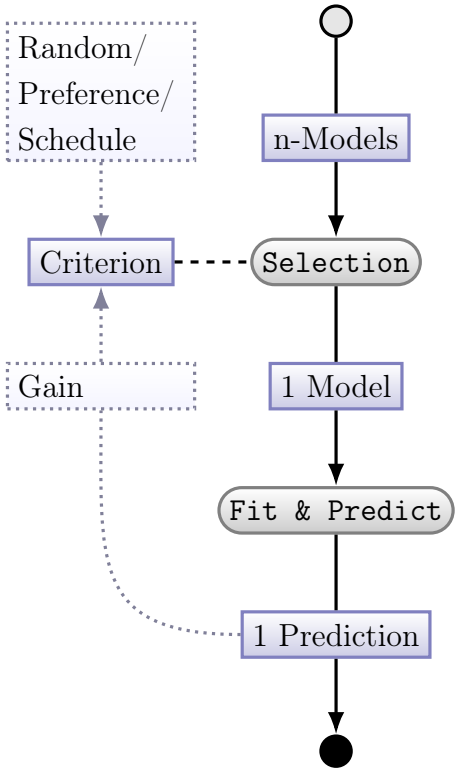


Figure 3.1: The Figure illustrates the general procedure in the case of *Single Evaluation Model Selection*. Given, that more than one model is available, one model has to be chosen with respect to some criterion. This model can then be fitted to the data and used for prediction. Knowledge gained in this step may be used in future decisions.

solution to the model selection problem in SPO as well as to the cold start problem in online modeling applications when there is not yet enough data available [88]. In the latter case, learning the choice of the right model when there is no data available yet corresponds to problems in the context of multi-armed bandits [77]. Methods proposed to approach the model selection problem, in this case, are often based on known solutions to the multi-armed bandit problem or use simple schedules.

The 'model scheduling' approaches provide a schedule that specifies the order in which to select a model. The most straightforward solution is to use a random order or to apply a round-robin strategy [76]. More advanced solutions to the multi-armed bandit problem also consider results from previous evaluations. ε-Greedy strategies choose the model that provided the best reward so far with a probability of ε [76]. Soft-Max approaches keep probabilities for every model

available, starting with an even distribution of probabilities. After every step, all probabilities are then adjusted according to the success of the last model [76]. To allow for a better distribution of the initial probabilities, an initialization step can be carried out when enough data is available [89]. The Bayesian Learning Automaton keeps track of a variable for each model that affects the probability for the corresponding model to be chosen, but unlike the Soft-Max approach, these variables are adjusted independently. Also, many other statistical criteria are used for model selection; an overview of some of them is given in [90].

All of these approaches benefit from the fact that in every step only a single model has to be fitted to the data. However, these ad hoc rules ignore the rules of parsimony and do not, or only marginally, rely on the data to help select the best model. The use of a schedule, or to chose even randomly, gives the same chance to adequate models and inadequate models likewise. The approaches that utilize the reward of the last model struggle with the same problem. The algorithm may settle down to a single model or a small subset of better performing models after some time. But the chances are that due to unfortunate choices for the reward evaluation and during the first steps, better performing models are put at a disadvantage.

3.3 Multi Evaluation Model Selection

A simple way to approach the drawbacks that come with the model selection methods presented in Section 3.2 is to evaluate all models on the data. This way the decision can be made based upon the models' predictions or their performances. Figure 3.2 displays the flow of such approaches. ε-Greedy approaches choose the model that performed best in the previous step, in terms of prediction error, with a probability of ε, with $\varepsilon \in [0, 1]$. With the complementary probability of $(1 - \varepsilon)$ one of the remaining models is chosen randomly [76].

This choice must not be taken based on knowledge from the last step but can also be based on the predictions itself. For such approaches, all models have to be fit to the data in an initial step. Baxter [91] proposed to choose between the predictions of two neural networks that were trained for different objectives by comparing their predictions to a predefined threshold.

Jacobs et al. [92] proposed to combine a set of neural networks by using one of the networks as a gating network that learns which model is the best choice for a given input.

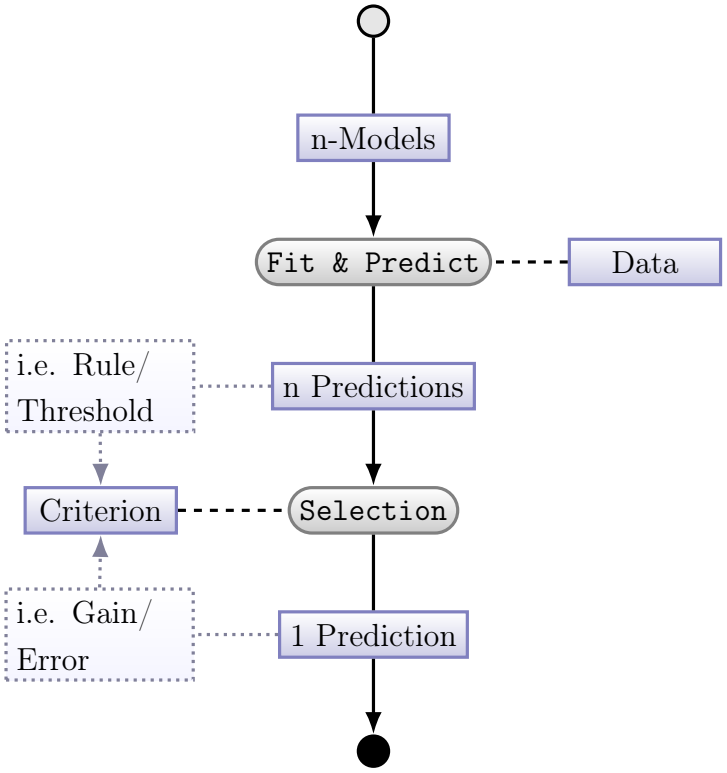


Figure 3.2: The Figure illustrates one approach to *Multi Evaluation Model Selection*, where the choice of the model is based on its prediction. For this purpose, all available models are fitted to the data. The model whose prediction satisfies the predefined criterion is chosen. Knowledge gained in this step may be used in future decisions.

Still, these approaches have limited insight into the performances of the models. Introducing an additional step to evaluate the models on the data allows for a deeper insight into the performances of the models on the data. By applying methods such as cross-validation, the prediction errors of the models can be estimated directly. Figure 3.3 displays the flow of such processes.

Friese et al. [76] proposed a method that does a leave-one-out cross-validation and an additional fitting step on the full data set to gain information about the models underlying uncertainty. This information is then combined with the models' error from the last prediction to obtain an indicator for the selection of the model.

An obvious drawback of such approaches is the number of fitting processes that have to be carried out to evaluate all models the one or the other way. For an

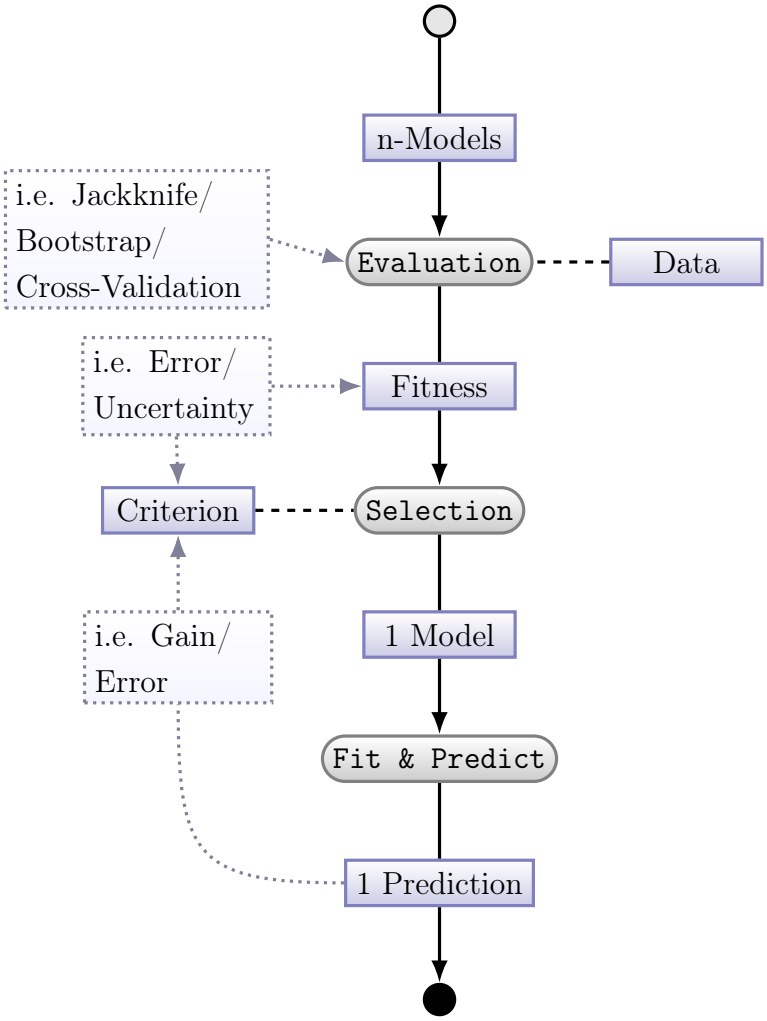


Figure 3.3: The Figure illustrates an approach to *Multi Evaluation Model Selection*, where the choice of the model is based on the general fit of the model on already known data. For this purpose, all available models are evaluated on the available data. The model that scores best during this evaluation is chosen. Knowledge gained in this step may be used in future decisions.

online or sequential process with constraints on the calculation time for the models this step might not be feasible. However, in real-world applications, the actual expenses originate from the evaluation of the objective function. The calculation times may be of small concern.

3.4 Model Combination

Model selection strategies, as introduced before, assign scores to the candidate models to allow for evaluation. In the case of single evaluation model selection, as introduced in Section 3.2, this score is based on performance values of previous steps, whereas in multi evaluation model selection, as introduced in Section 3.3, this score was obtained by a preliminary evaluation of the available models. Such evaluations might yield a clear winner, but they might also lead to the insight that several candidate models perform comparably well. Claeskens et al. [93] state, that in such circumstances, it may be beneficial combining the predictions of these strong models to obtain a more accurate or even better prediction. A straightforward approach to achieve this would be to fit all models to the data and then apply some rule that defines how to combine the various predictions to one.

Figure 3.4 displays the process flow of such methods.

With Bagging [29] multiple versions of a model are generated by fitting them to different partitions of the training data (further referred to as multi-data). Breiman et al. showed, that by averaging the predictions of these models a prediction of higher accuracy can be achieved. Random Forests [26] are another example of this approach where Bagging is used with trees. Their output is combined by majority voting, in the case of classification, or by averaging, in the case of regression.

Boosting [81, 94] is like an add-on to bagging that performs the fitting of the model on the multi-data sequentially. This way data that has not been learned adequately can be considered for the next fitting process with a higher probability. Models generated are combined by weighted averaging, with the weights derived from their prediction errors on the remaining data of the training set.

Bishop [95] combined a set of artificial neural networks to one committee of networks by calculating the weighted sum of their predictions. His method uses the error that the models make at the training points to define the models' weights. The only restriction on the weights is that they have to sum up to one.

Van Stein et al. [96] clustered the training data and learned an independent Kriging model on each data cluster. They introduced a variety of methods to cluster the data and proposed several approaches to retrieve a single prediction from this cluster. The original method [97] considers the Kriging models to be independent, and given this assumption computes an optimally weighted average. One of these approaches is to learn a decision tree on the data with its leaves

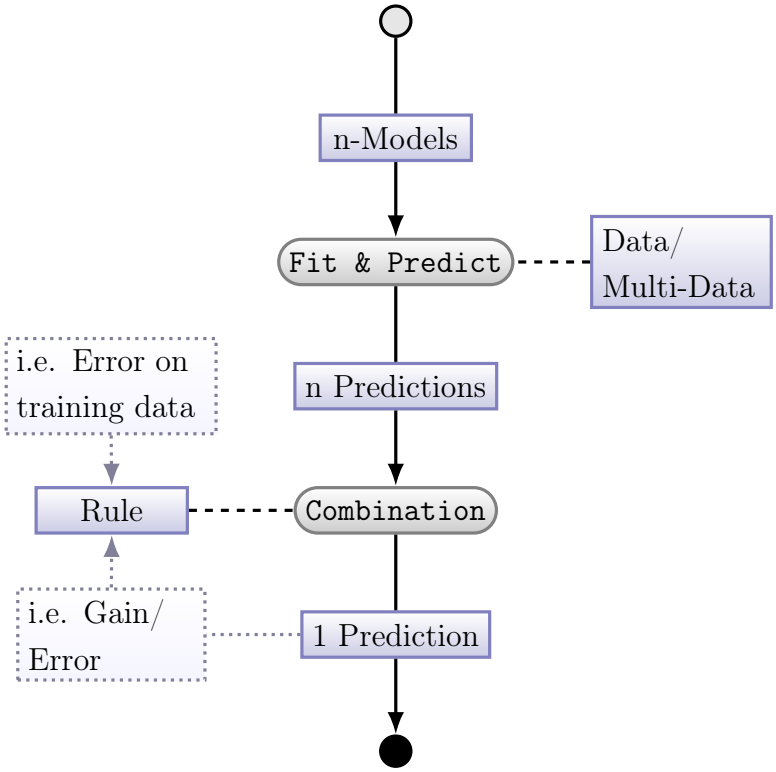


Figure 3.4: The Figure illustrates an approach to *Multi Evaluation Model Combination*, where the predictions of multiple models are combined using a predefined rule. For this purpose, all available models are fitted to the data, or partitions of the data. Knowledge gained from the prediction, or the single predicitons, may be used in future decisions.

being Kriging models. The probability for a data point to be assigned to one leaf of the tree is also used as the weight for the calculation of a weighted sum of the predictions.

Given a set of heterogeneous models is to be used, the models can be fitted to the complete data, variance in their predictions is already given through the heterogeneity of the models, and then be combined by averaging their predictions. However, working with heterogeneous models, this approach may run the risk that one model in the set performs considerably worse, which would bias the outcome of the averaged prediction. Using information (i.e., prediction error) from the last step would minimize this risk [76].

Zerpa et al. [98] proposed to combine heterogeneous models, that are able to provide information about their variance, by calculating a weighted sum of their

predictions. The information about the models' variance is used for the calculation of the models' weights.

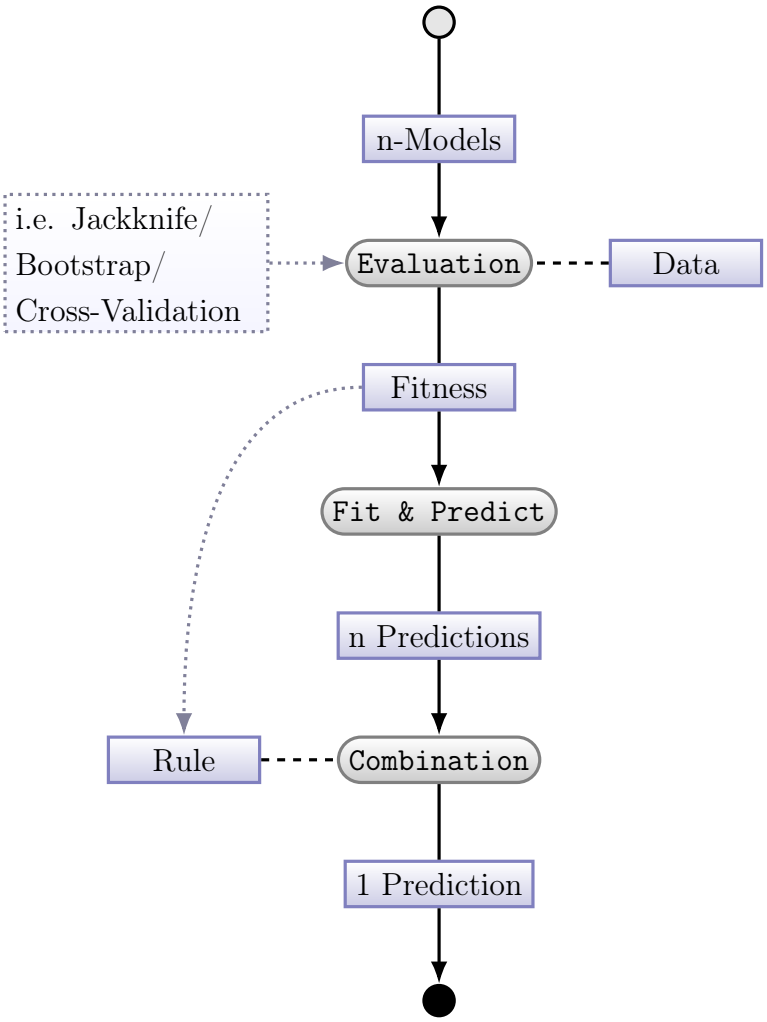


Figure 3.5: The Figure illustrates the most elaborate approach to *Multi Evaluation Model Combination* where the predictions of multiple models are combined according to their fitness on the available data. For this purpose, all available models are evaluated on the data. Each model's score directly affects how the models are combined.

Like with the Model Selection strategies here also an additional step to evaluate the models fit can be introduced (cf. Figure 3.5). Perrone et al. [99] combined a set of ten neural networks by computing a weighted average of the models' predictions. They performed a cross-validation step on the training data to determine

the prediction errors and calculated the optimal weights, in terms of minimal mean squared error[1] (MSE) based on these errors.

Goel et al. [100] combined three heterogeneous models by calculating the weighted sum of the models' predictions. To determine the weights, they used a formula that is based on the generalized mean square cross-validation error of the models. The formula uses two parameters α and β that have to be specified, and control if more trust is laid on the general average of all models predictions (larger α values and smaller negative β values) or if more weight is laid on the single model.

Acar et al. [101], proposed some adaptations of previously defined approaches of weighted sum ensembles for better generalizability and performance. Like in previous works they also required the weights to sum up to one as the only constraint on the weights. Building on the approach of Bishop et al. [95], they proposed to use k-fold cross-validation to allow for an evaluation of models that have per definition no error at the training points. For the works of Goel et al. [100] they suggested to select the parameters α and β to minimize the generalized mean squared error[2](GMSE) of the ensemble. Another approach they propose is a weighted sum ensemble that is evaluated by its root mean squared error[3] (RMSE) with respect to a predefined number N_V of evaluation points ($N_V = 2, 3$ or 5). They optimize the weights using a gradient-based search algorithm starting from the center point, annotating that the search space is not necessarily convex so that the solution possibly only presents a local minimum.

As mentioned before, an obvious drawback of these approaches is the number of fitting processes that have to be carried out. But, unlike in model selection processes, the additional information that might be retrieved from the other models is not necessarily discarded. Previous works show that further enhancement in accuracy can be achieved by combining several models of similar accuracy. However, also with weaker models being part of the set, the combination of models can be beneficial if the strengths and weaknesses are carefully considered.

[1] The MSE calculates the average of the squares of the errors between the observed value and its estimation. It is defined as $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$.

[2] The GMSE refers to the MSE applied in a leave-one-out cross-validation process.

[3] The RMSE calculates the root of the MSE. It is defined as $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$.

3.5 Further Ensemble Generating Strategies

The ideas regarded so far give an overview of the main strategies and ideas on the topic of ensemble building. Of course, this roundup does not claim to be complete. As said in the beginning, the main focus for this overview is laid on methods that conform with the goal of this work. However, there remain approaches to combine models and areas of application that have not been mentioned so far.

Torgo et al. [102] presented a method that uses a regression tree with several alternative independent models in the tree leaves and this way outperformed standard regression tree methods that only average the function value represented by a single leaf.

Dasarathy et al. [103] partitioned the feature space to use two or more classifiers in a pattern recognition system.

Van Stein et al. [104] proposed to improve the performance of Kriging on functions of higher complexity, in terms of dimension or number of known points, by partitioning the data set into smaller disjoint clusters of data, distinguishing between randomly generated clusters and location-based clustering with randomly chosen center points. Their approach trains individual Kriging models on each data cluster. For the prediction, a weighted average of the different models' predictions is calculated using the variance information for each cluster. They showed that using location-based clustering leads to better results than random clustering. The so-called Cluster Kriging generated this way outperforms Ordinary Kriging in terms of computation time as well as in terms of accuracy.

Ginsbourger et al. [105] proposed to use multiple kernels within Kriging. In this case, a mixture of the kernels is defined using a weighted sum. Friese et al. [106] applied the idea of ensemble modeling to time series analysis and forecasting. By adjusting the seasonality of the input data in a preprocessing step they enabled a larger set of models to be used on the data. For each forecasting step, they learned all available models and then used the AIC for the selection of the most promising model.

The possibilities of defining ensembles considering only the different options of combining and evaluating base models, as well as ensembles, seem vast. For evaluation, different methods (i.e., Bootstrapping, Cross-Validation) as also different quality measures or criteria are available (i.e., RMSE, MSE or AIC). Most approaches apply some form of weighted sum for the combination of multiple models. However, for different applications also different combination schemes may be beneficial. Kittler et al. [82] compared the operators product, sum, min,

max, median and majority voting to combine multiple models in a pattern recognition system.

$$
\begin{aligned}
\boxed{\textit{policy}} &= \textit{base policy} \\
&\mid \textit{policy} + \textit{policy} \\
&\mid \textit{policy} \times \textit{policy} \\
&\mid \textit{policy} - \textit{policy} \\
&\mid \text{mean}(\textit{policy}, \textit{policy}) \\
&\mid \min(\textit{policy}, \textit{policy}) \\
&\mid \text{decision}(\textit{point}, \textit{index}, \textit{threshold}, \textit{policy}, \textit{policy}) \\
\textit{base policy} &= \textit{base model}(\textit{point}) \\
&\mid \textit{constant scalar} \in \mathbb{R} \\
\textit{base model} &= \textit{fast base model} \mid \text{KF} \mid \text{MLP} \mid \text{SVM} \\
\textit{fast base model} &= \text{LM} \mid \text{MARS} \mid \text{RF} \\
\textit{point} &= x \mid \textit{constant vector} \\
\textit{index} &= 1 \mid 2 \mid \cdots \mid d \\
\textit{threshold} &= \textit{constant scalar} \in [-1, 1]
\end{aligned}
$$

Figure 3.6: Grammar given in Extended Backus-Naur Form defining the set of valid policy expressions. Terminal symbols are shown in a regular typeface, non-terminal symbols are shown in *italics*. The start symbol of the grammar is marked by a $\boxed{box}$. (based on [107])

Flasch et al. [107] proposed to use genetic programming to automatically build an ensemble, also allowing for multiple operators in an ensemble. For this purpose, a grammar of model ensemble expressions is defined and then searched using genetic programming.

Figure 3.6 shows the grammar that was used for the experiments presented. Using this grammar enables the method to build complex tree structures by combining multiple models using different combination schemes within an ensemble.

3.6 Conclusion

Recapitulating all approaches we considered in this chapter, it can be said that there are many ways to get from a set of models to a single prediction. Moreover, all approaches have their strengths and weaknesses, that have to be considered carefully to choose the most appropriate strategy for a given problem description.

Figure 3.7 combines the approaches considered in the Sections 3.2 through 3.4 in one diagram and gives an overview over the main decisions that have to be taken to specify an ensemble approach. The first of these questions is if a preliminary evaluation of all models on the data should be carried out. This evaluation is typically done by Jackknife, Bootstrap, Cross-Validation or any variations of these. The main characteristic that all of these approaches have in common is that the model is fitted to a part of the data and evaluated on the remaining data, at least once. The more computation time is spent in this step, the more information about the models fit can be gained here.

The next decision that has to be taken is if all models should be fitted to the data or if a single model is chosen for the next prediction. It has to be remembered that the preliminary evaluation step is optional. Thus if the choice of the model is taken now without preliminary evaluation of the models on the data the relation between this choice and the underlying data may be rather sparse or not existing at all. On the other hand, an exhaustive preliminary evaluation of all models allows for a reasonable selection of the best model that is well-founded on the underlying data.

Anyhow, if it is not desired to select a single model at this point, then all models have to be fitted. This step is also not depending on the preliminary evaluation step that might have been performed initially since the models in this step in general only have been trained on parts of the data.

The last decision that has to be taken after training all models to the complete data is if the prediction of one model is chosen or if the predictions of all models are combined into one, presumably more accurate prediction. This decision may depend on whether or not a preliminary fitness evaluation has been carried out or on particular conditions of the problem that has to be modeled. If the preliminary evaluation step has been carried out, it would be an utterly unnecessary step to fit all models to the data and then discard all but the best. Whereas when no preliminary evaluation has been carried out, it may depend on the task if it is best to select one prediction or combine all into one.

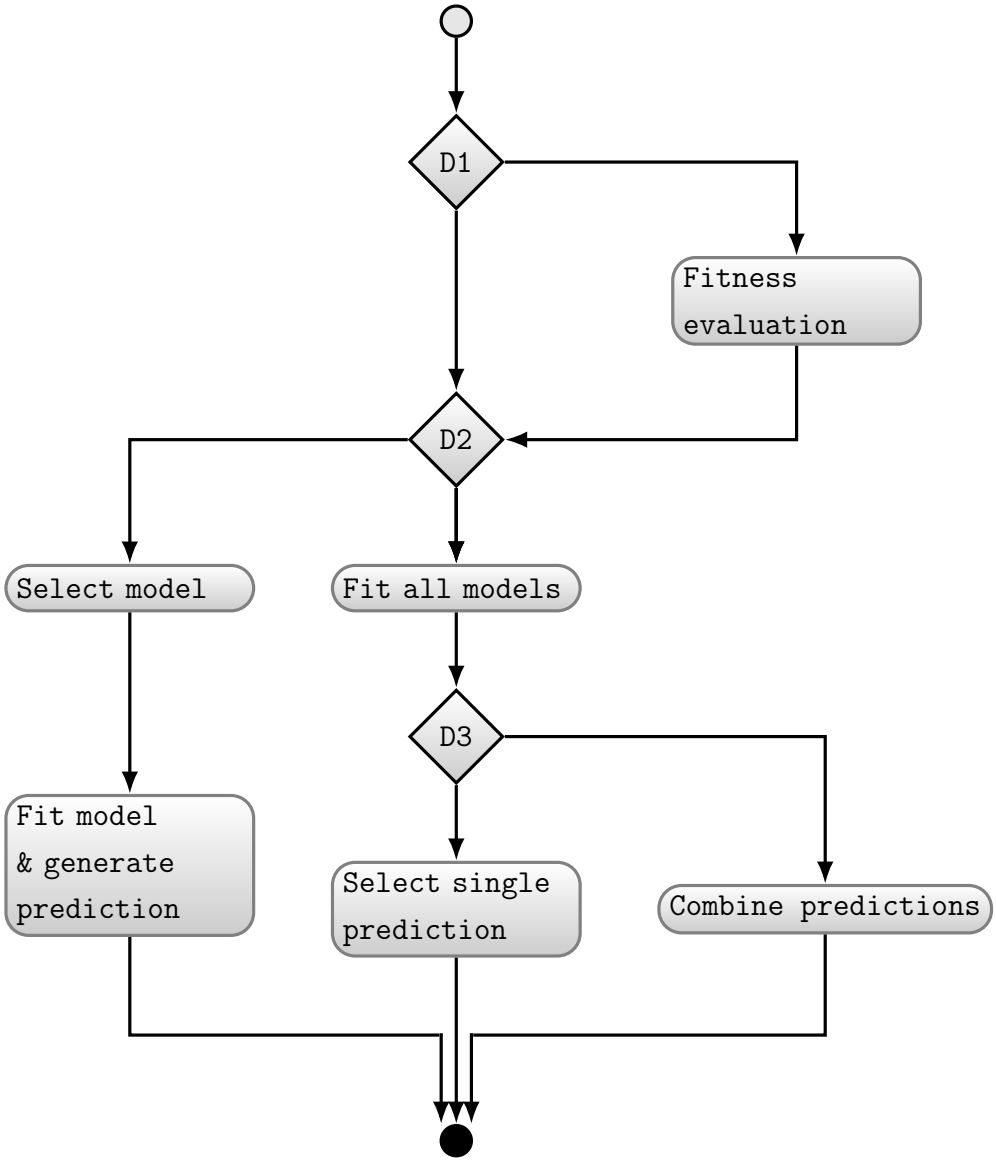


Figure 3.7: Overview over the decisions that have to be taken to define the way how to get a single prediction from a set of models.
D1: Do a fitness evaluation (i.e. cross-validation) on available models?
D2: Select single model or fit all available models?
D3: Select prediction of one model or combine available predictions?

The overall goal of this work is to create a strategy that works reliably and as accurately as possible on arbitrary objective functions well knowingly accepting that this is probably going to happen at the expense of the ensembles compu-

tation time. Thus a method is envisioned, that does an exhaustive preliminary evaluation of all models to gain the best insight into the models' performances, then trains all models on the data to enable the use of the complete knowledge of all models. Still, the method should follow the principle of parsimony and prefer a combination of predictions over a single prediction only if it is clearly beneficial for the overall accuracy. The same applies to the number of models used, it should not be a decision between a single model or a combination of all, but any number of models that seem to be best.

Chapter 4

Building Ensembles Using Convex Linear Combinations

In this chapter, the taxonomy specified in chapter 3 is used to develop a new ensemble method. This approach is studied fundamentally, by first evaluating ensembles of only two surrogate models in detail and then proceeding to ensembles with more surrogate models. Last, experiments are carried out on objective functions based on physical models. The results show to what extent combinations of models can perform better than single surrogate models and provide insights into the scalability and robustness of the approach.

As concluded from the insights gathered in Chapter 3, the preferred method of ensemble building for the considered task of optimizing expensive black box functions is supposed to be a combination of the predictions of several strong surrogates. It appears to be an interesting idea to linearly combine several models into more complex models. A convex linear combination of the models' predictions is both easy to calculate also for several heterogeneous models and comprehensive in terms of meaningfulness.

This Chapter is structured as follows. In Section 4.1 the ensemble building method is defined, thoroughly tested on different settings and analyzed for its strengths and weaknesses. Also, the influence of the RMSE on the behavior of the ensemble is regarded.
In Section 4.2 the method is extended for the use of three base models. To further extend the method for the use of more than three models, the resulting size of the search space has to be taken into account. This is done in Section 4.3.

Additional adjustments have to be made to ensure the functioning of the method with a larger set of models. This is done in Section 4.4. Lastly, in Section 4.5, the method is tested on a set of physical functions using different settings for the previously made adjustments.

4.1 Binary Ensembles

To investigate, whether and why combinations of models by linear convex combination could be beneficial it is reasonable to start with binary combinations of models, to enable an in-depth analysis and a better understanding of the functioning of such ensembles. The main questions are:

- Can such combinations of models compete with, or even improve on the performance of single models?
- Given the answer is positive, how can the observed behavior be explained?

In the following, convex combinations of models (CCM) will be referred to as ensemble model or CCM, while the original models will be referred to as base models.

We focus on positive weights since we do not want to select models that make predictions that are anti-correlated with the results. Given a weight α, where $\alpha \in \{0.0, 0.1, \ldots, 1.0\}$, the ensemble model can be defined as the linear combination of the models a and b as follows:

$$\hat{y} = \alpha \times \hat{y}_a + (1 - \alpha) \times \hat{y}_b \tag{4.1}$$

The prediction $\hat{y}$ of the ensemble model is a weighted sum of the predictions $\hat{y}_a$ and $\hat{y}_b$ of the base models a and b. With this definition of the combination, it is also ensured, that the total weight sums up to one.

To get an impression whether such a combination can be beneficial the two base models and resulting ensemble models are tested and compared against each other in a first simple experimental setup.

The experiment is carried out on an instance of a GLG function, which was introduced in Section 2.3, of dimension=4 featuring 40 Gaussian components (for additional parameters cf. Section 2.3, Table 2.1). A sample of points (design) is evaluated on the objective function. For the sampling of the points, a Latin hypercube design featuring 40 design points is generated.

The two base models are Kriging with exponential correlation function (referred to as a) and Gaussian correlation function (referred to as b). The models are evaluated by calculating the RMSE of the predictions made during a leave-one-out cross-validation on the 40 design points. Both base models are fitted to the data and then used to predict the left out point. The predictions $\hat{y}$ of the ensemble models are calculated as convex combinations of the predictions of the base models as specified in (4.1).

Since randomness has been induced into the experiment by using the Latin hypercube design, the evaluation process has been repeated 50 times in order to receive a meaningful result.

To get a first quick insight into the result data, for each repetition the rankings of the RMSEs have been calculated. The models with $\alpha = 0.6$, $\alpha = 0.8$ and $\alpha = 0.9$ dominate this comparison, each performing best 8 times out of 50. The base models, a and b, performed best only in four respectively two cases out of 50. Figure 4.1 shows the distribution of the ranks of each model.

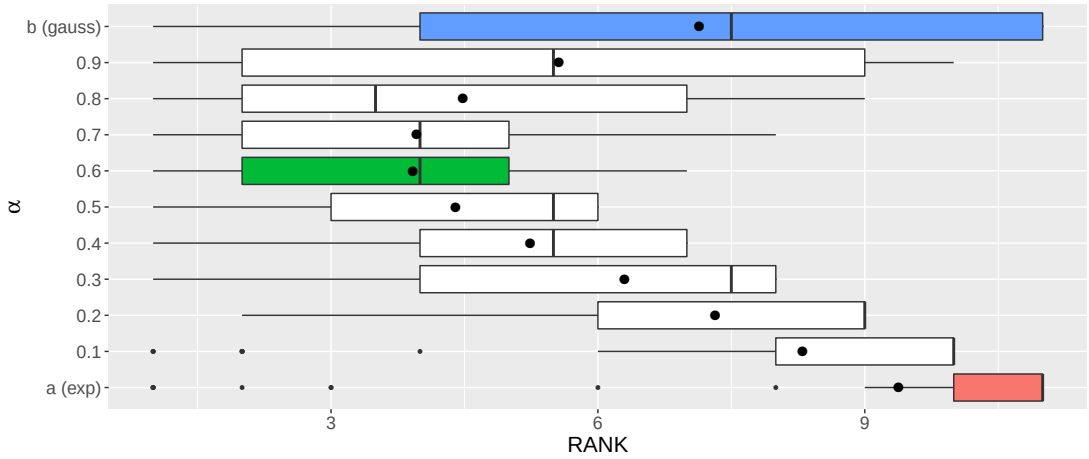


Figure 4.1: Boxplot over the repetition wise ranks of all models. The models are defined by an α-weighted linear combination of the two base models. The results of the base models are depicted on the outer rows and colored red (exponential kernel), and blue (gaussian kernel) respectively. The model combination chosen as best with $\alpha = 0.6$ is colored green. The mean value of each result bar is marked by a dot.

Model a (exponential) performed best in four of the cases but worst in 36. Model b (gaussian) shows a larger variation in its performance. It has been the best

performing model in two cases and performed worst in 14. In none of the cases an ensemble model was performing worst. Overall a parabolic tendency can be seen in the performance. This indicates that linear combinations of the models are indeed beneficial.

4.1.1 Detailed Analysis on Transparent Test Cases

It can clearly be stated that for this first experiment setup the combination of two models is beneficial for the overall prediction. In this section we will have a closer look at possible explanations for the successful result and will address the following questions:

- Are there problem features that encourage using ensembles?
- And is this result generalizable?

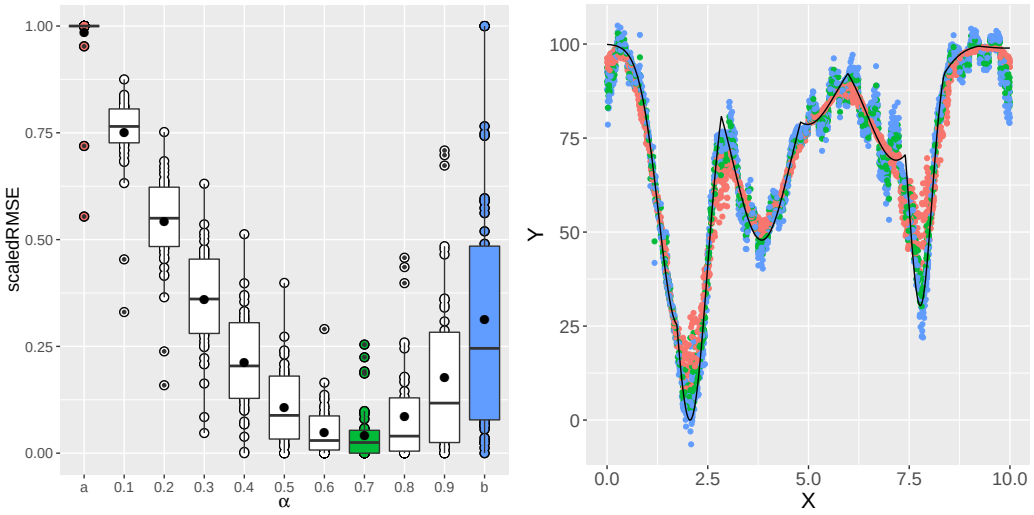
A larger number of experiments is carried out to analyze the performance and benefits of the ensembles.

As a consistent method for evaluating the performance and automatically choosing the best model, the following approach is proposed: Model-wise mean-, median- and 3rd quartile-values are calculated. The resulting values are ranked and then summed up to one final ranking. The model that achieved the lowest value is recommended as the best choice. In Figure 4.1 the model recommended as the best choice by this method is colored green. Applying this method, it might as well be more than just one model returned as best ensemble choice. In these cases, a decision has to be made about which model should be favored. Here, giving more importance to the median or 3rd quantile value means favoring a smaller distribution but taking the risk of larger outliers, while giving more importance to the mean value means having smaller outliers, but allowing for an overall wider distribution. Other criteria also might be taken into account.

For a better understanding of the underlying process and the strengths and weaknesses of the method, experiments are carried out on one-dimensional objective functions. The previous experimental setup has been preserved, except for the change of the objective functions' dimensionality.

Figure 4.2a shows exemplary results from these experiments. To allow for an easier visual comparison, the RMSE's of the models have been repetition-wise scaled to values from zero to one. The plot on the left hand side depicts these values. Applying the rule defined in Section 4.1.1 the model obtained by a linear

(a) Results on a GLG objective function



(b) Results on a piecewise assembled objective function.

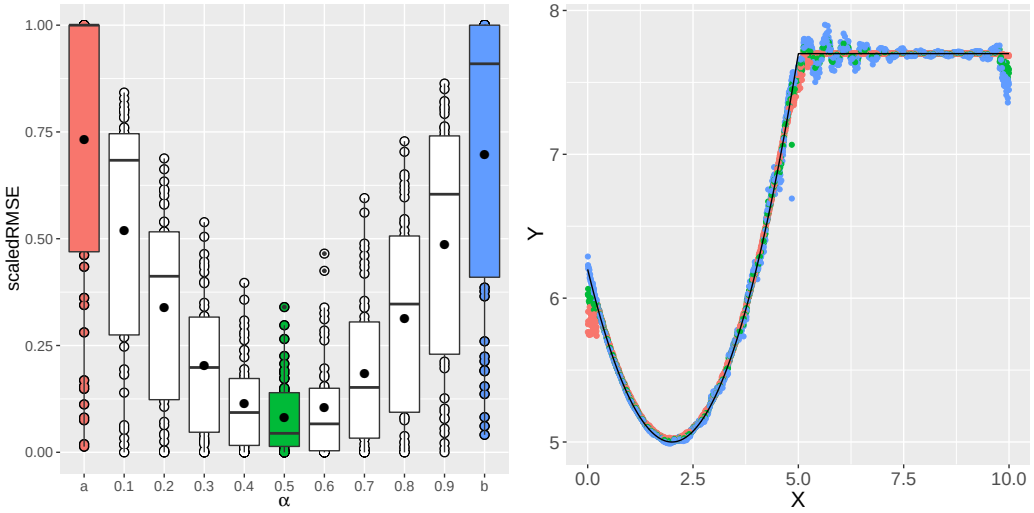


Figure 4.2: The plots show the results on two different one-dimensional objective functions. Each plot on the left shows the repetition-wise scaled RMSE's for each model. The α value defines the weight for the linear combination, the model that has been chosen as best is colored green. On the right hand side all predictions done during the leave-one-out cross validation for the base models and the best model are plotted against the objective function.

combination with $\alpha = 0.7$ is marked as best choice.

The plot on the right-hand side shows the predictions of the models versus the objective function. In this plot, only the two base models and the best ensemble are regarded. Each dot marks a single prediction made during the leave-one-out cross-validation. In every repetition, each model makes one prediction for every point of the design. This results in a total of 2000 prediction values for each model (40 design points $\times$ 50 repetitions).

As can be seen in the plot, the predictions of the model a (exponential kernel function), marked by red dots, seem to smooth the objective function: straight segments are well met while curved segments are smoothed out.

The predictions of the model b (Gaussian kernel function), marked by the blue dots, show signs of overfitting. Again, straight segments are well met, but when approaching local extrema, the predictions start to oscillate. So the linear combination of both predictions averages positive as well as negative outliers of base models. This seems to provide some benefit to the overall experiment outcome.

To confirm this assumption two additional experiments are carried out. For these experiments two objective functions are specified featuring corners that are not continuous differentiable. For one experiment a triangle objective function is used while the other features a piecewise assembled objective function. Figure 4.2b depicts the results on this function. A parabolic graph is joined together with a straight line graph. Special focus is laid on the joint of both function parts and on the straight part. Both base models succeed in describing the parabolic part of the function, but while base model a (red) fails at the borders and at the junction point, base model b (blue) is not able to describe the straight part of the function. The ensemble (green) benefits from this difficulties. We again find a strong parabolic tendency in the boxplot. Both base models have a rather large variance in their performance. The ensemble model marked as best choice has a smaller variance and performed better than the base models in nearly all cases.

The results on the triangle objective function happened to show a clear tendency towards base model b, which clearly outperformed base model a and thus was chosen best. This also is a good result, since it is desirable that the method only chooses an ensemble over a base model if it is actually better.

4.1.2 Detailed Analysis on Single Predictions

To better understand what is happening, we take a closer look at convex linear combination itself and how it influences the fitness of the resulting model. The main questions are:

- In what circumstances can the prediction of a CCM be better than the prediction of the base models?
- Is it possible that the prediction of a CCM is worse than the predictions of both base models?

To answer these questions, we consider two base models and related CCMs making a single prediction. For reasons of simplicity, we assume, that the real function value is zero and that the base models are making different combinations of prediction errors.

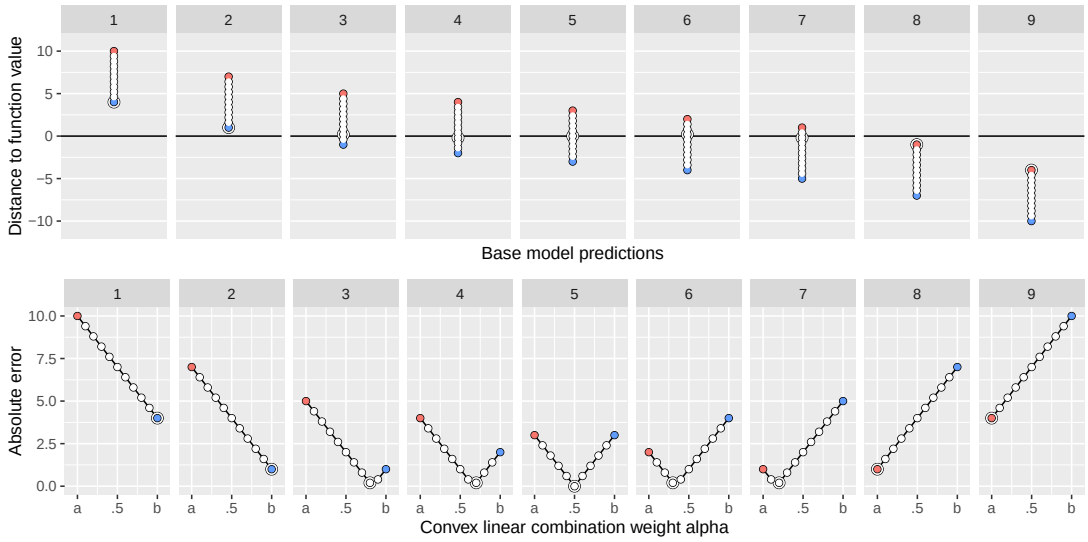


Figure 4.3: The upper row plots show the predictions of two base models and its related CCMs for a single point. The true function value of zero is marked with a line. In the lower row the related RMSEs are given. The best prediction and the related smallest prediction error are marked with an additional circle. It can be seen that in some cases the CCM performs better than both base models, but it never performs worse than both.

Figure 4.3 displays such a situation. In the upper row, the colored points mark the predictions of the base models, while the white points mark the predictions

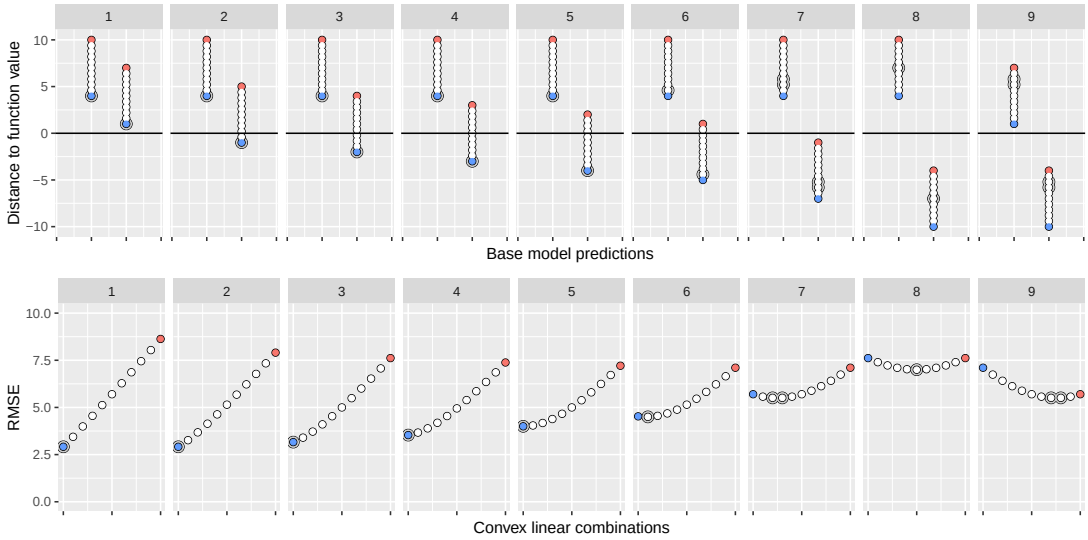
obtained through convex linear combination. The true function value is set to zero, displayed by the horizontal line.
The lower row depicts the related resulting RMSEs. The best prediction and its related error are marked with an additional circle.
Since we are only considering a single prediction, the resulting RMSE equals the single prediction error. Thus, as assumed before, if the base models make opposing prediction errors, the prediction of the CCM will be better than the predictions of the base models (cf. Figure 4.3, Columns 3-7).
Whereas, when both base models make the same kind of prediction error, the prediction, as well as the prediciton error, of the CCM lies inbetween the predicions and the errors of the base models. (cf. Figure 4.3, Columns 1,2,8 and 9).
All other possible relations between prediction errors and resulting RMSE can be easily derived from this. So would a change of the distance between the base models predictions result in a steepened slope, while a larger distance between the two predictions and the real function value would result in a larger absolute error. The most important information we can read from these plots is that for a single prediction the convex linear combination of the predictions cannot be worse than the prediction of the weaker base model, independent of the weight chosen for the combination. Moreover, for some values of α the combined prediction can be better than the predictions of both base models given, that one model overestimates the actual function value while the other model underestimates it. In the experiment (cf. Section 4.1) this happened in 649 out of 2000 cases.

However, when evaluating the ensemble built by the proposed method of building convex linear combinations, this is not done based on a single prediction but on a set of predictions. Also, calculating the RMSE of a single prediction only yields the simple prediction error. Questions that also have to be considered are:

- Do these findings made for single predictions still apply for multiple predictions?
- How does the RMSE influence the evaluation of the models' overall performance?

To answer these additional questions, the analysis is extended to two predictions. Figure 4.4 displays several possible combinations of predictions for such a setup. Figure 4.4a shows a set of predictions where the one model always predicts a larger value than the other model, while in Figure 4.4b one model always returns a larger prediction for one point and a smaller prediction for the other point. In both plots, like before, the upper row shows the predictions, base model predictions are colored blue or red, and predictions of the linear combinations are colored

(a) One base model always predicts a smaller value than the other



(b) Each model predicts larger in one and smaller in the other case

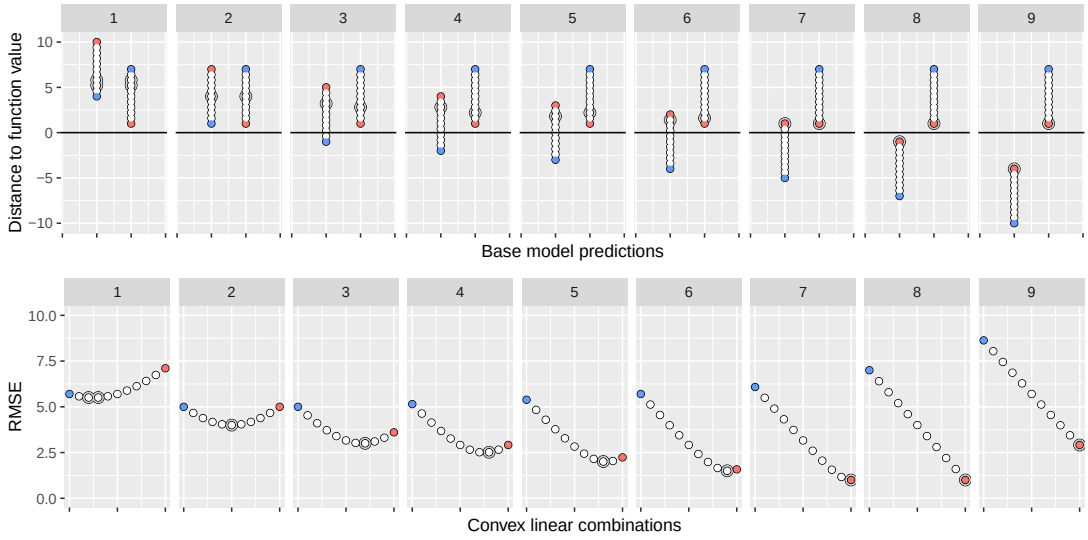


Figure 4.4: Both plots show a set of combinations of two predictions made by two base models (red and blue) along with the resulting predictions made generated through convex linear combining using a fixed step width of 0.1. The second row of each plot displays the related RMSE values resulting from the combination of the two predictions.

white. The lower row displays the relating RMSE values resulting from the two predictions.

We can see from these plots, that it does not suffice for the ensemble prediction to be better than the base models, when in one of the two cases the prediction spans the actual function value, while the combined predictions of one of the base models are better in terms of RMSE (cf. Fig. 4.4a items 2-5, 4.4b items 6-7). The combined prediction error, in terms of RMSE, rather benefits from opposing errors in general (cf. 4.4a items 6-9, 4.4b items 1-5).

As mentioned before, the CCM can make a better prediction on a single point only if the predictions of the base models span the true function value.
However, regarding more than one prediction, this evaluation is influenced by the characteristics of the RMSE. The overall fit of the CCM, in terms of RMSE, can be better than both base models also, when none of the base models predictions is spanning the true function value. For such an evaluation it is sufficient when one model returns a larger prediction than the other model for one point, and a smaller prediction for the other point.

Noticeable is also, that in contrast to the plots shown in Figure 4.3, the points marking the RMSE values in most of the cases lie on a parabolic curve. This might indicate that the search space, generated by the RMSE could also be convex. Interpreting the RMSE as a function describing a form of mean squared distance would support this claim. However, this assumption is based on observations only and would require further analysis.

4.1.3 Conclusion

The insights gained so far are promising and seem to make convex linear combinations an ideal choice for combining models. So far recognized advantages are:

- Due to the convex linear combination that is used for combination, the ensemble cannot perform worse than the weakest base model.
- The ensemble can perform better than the base models when compensating opposing prediction errors.
- A CCM is favored over a base model only if the overall fit of the ensemble model is actually better (in terms of RMSE), than the overall fit of both base models.

- The nature of the combination is intuitive and interpretable.
- The linear convex combination of predictions for a given set of weights is easy to compute.

4.2 Ternary Ensembles

Up to this point, the first experiments have been carried out, and the method showed promising results. However, the question arises if this method is scalable and generalizable. In this section the experiments are extended to a larger scale: The dimensionality of the objective functions is increased, and the method is adapted to enable the combination of three models. As before, Kriging models with different kernels are used, but now a third model using the spline correlation function is added.

For the linear combination of three base models also three weights are needed, that sum up to one as specified in (4.2).

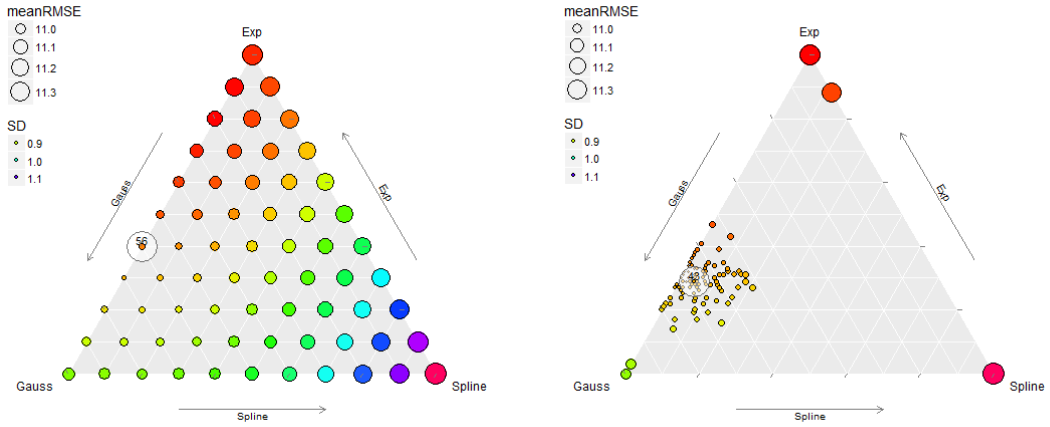
$$\alpha, \beta, \gamma \in \{0.0, 0.1, \ldots, 1.0\}, \quad \alpha + \beta + \gamma = 1 \tag{4.2}$$

Retaining the step size of 0.1 for the linear combinations results in 66 ensemble models.

As a first step towards problems of higher complexity, the dimensionality d of the objective function is set to 4. However, this change alone is not sufficient to gain a larger complexity, since without adjusting the number of Gaussian components used for generating the objective function, it rather gets less complex. Thus the number of Gaussian process trajectories is adjusted to forty times the dimension.

With increasing the complexity of the objective function also the size of the design should be adjusted to gain reasonable results. Desirable is a design that is preferably small yet allows for a proper fit of the models. To achieve this an experiment with varying design sizes is run beforehand. For each size, the fit of the base models to the objective function, in terms of RMSE, is compared to a simple mean predictor. The design size chosen for the experiment is that with the smallest design size, such that all base models are performing significantly better than the baseline predictor.
For the experiment presented in this Section, this occurred at a design size of 160.

Figure 4.5a shows an exemplary experiment result using three base models. The depiction of the result corresponds to the previously shown boxplots. The ternary plot also shows parts of these data but allows for a more intuitive interpretation than a boxplot would do.



(a) The optimal linear combination has been found by a complete evaluations of all linear combinations using a fixed step size of 0.1.

(b) The optimal linear combination has been searched with a simple (1+1)-Evolution Strategy with 1/5th success rule (cf. [108]).

Figure 4.5: The plots show the results of the experiment set up with three base models. Each circle depicts the performance results for one model. The three base models are located on the corners of the triangle, models gained by linear combinations of only two models are located on the outer border. Circles on the inner region of the area show the results for models that were gained by linear combinations of all three base models. The size of the circles denotes the mean RMSE value, the color the standard deviation. The model proposed as best choice is marked by an additional white circle.

Each circle in the plot displays the performance of a single model. The base models are positioned on the corners of the triangle, models on the outer border are built from a linear combination of only two of the three base models. All circles on the inner area of the triangle are linear combinations of all three base models; its proximity to the corner relates to the share this model has to the linear combination. The size of the circle corresponds to the mean RMSE that the model

has achieved, the color to the standard deviation of the RMSE values. The best choice model, as defined in Section 4.1.1, is marked by a white circle.

In most of the experiments carried out, the model that is automatically chosen as the best performing model is located on the inner area of the triangle. For some of the experiments, the best model is located on the outer border of the triangle. From these results, it can be deduced that the method not only meets the demands but also performs better than a single base model only.

4.3 From Exhaustive Search to an Evolutionary Strategy

The approach proposed so far proved its functionality using a small set of homogeneous base models. Still, the underlying goal is to evolve a system that can handle a large set of heterogeneous base models. In this section, preparations are taken to enable the method to also handle a considerably larger search space. To achieve this, the method is adapted to use an evolution strategy instead of performing an exhaustive search of the complete search space.

When increasing the number of available base models, also the number of possible discretized convex combinations between these base models grows exponentially. The number of resulting convex combinations can be calculated using formula 4.3. The formula recursively counts the possible combinations of weight distributions under the premise that the available weight units have to be completely distributed among the available models, but also allowing that a model receives no weight.

$$f(s,r) = \sum_{i=0}^{r} f(s-1, r-i), \quad f(s,0) = 1, \quad f(1,r) = 1 \tag{4.3}$$

The relation between the number of models, the step size for the discretized convex combinations and the resulting number of linear combinations can be expressed as a function of s, the number of models, and r, the reciprocal of the step size or the number of available weight units respectively. Recursion stops if either only one model or no distributable weight unit remains. Since weights have to sum up to one, the remaining model has to take the remaining weight. Alternatively, if no weight remains, the remaining models receive no weight. In both cases, only one possible option remains.
Using three base models and a step size of 0.1 as defined in (4.2) results in

$f(3, 10) = 66$ linear combinations. Now thinking of combinations of 10 base models already results in $f(10, 10) = 92,378$ linear combinations.

The complexity of the search space, when increasing the number of models, quickly gets too large to do a complete evaluation of all possible convex combinations with a fixed step size of 0.1. Moreover, the restriction to a fixed step size of 0.1 might be too rough.

Looking at previous findings, the function that describes the performance of the models built by convex combinations up to this point only showed convex, unimodal characteristics. This seems to be expectable due to the nature of convex combinations and the use of the RMSE as the fitness function. We expect the function to show this characteristic also when combining a larger number of models.

Thus, instead of a complete evaluation of all linear combinations, an optimization step is introduced to find the best combination instead of performing an exhaustive search on a fixed grid. The allowed weights are extended to a precision of two decimal places, allowing for $f(3, 100) = 5,151$ possible ensemble combinations. Since the area around the optimum tends to build a plateau smaller differences in the weights distribution seems to be negotiable. This restricts the possible search space to a reasonable size without losing the possible best solution.

For the optimization step a (1+1)-evolutionary strategy (ES) with 1/5 success rate for step width adaption is introduced [109, 110]. Building on the assumption that the regarded search space is supposedly but not surely unimodal and convex this would be a reliable and robust search algorithm that performs well on a search landscape as assumed but can also handle more difficult search spaces. However, other search strategies like Particle Swarm Optimization ([111]) would also fulfill these criteria.

Each individual of the population, representing an ensemble, is distinctly defined by its weights vector $\mathbf{v} = (\alpha, \beta, \gamma)^T$. For the mutation of the parent individual a vector of three random samples of a normal distribution function with standard deviation according to the actual step width σ is added to the related weights vector. Here, σ is initialized with $\sigma_{init} = 0.16$ and adjusted by the factor 0.9 according to the 1/5 success rate rule.

To ensure that the offspring individual still meets the requirements needed for a valid weight vector, the resulting vector $\mathbf{v}$ is adjusted in three steps:

1) If the smallest of the weights is smaller than zero ($\min(\alpha, \beta, \gamma) < 0$), this value is subtracted from the vector $\mathbf{v}$ ($\mathbf{v} \leftarrow \mathbf{v} - \min(\alpha, \beta, \gamma)$) to ensure that

all weights are positive.

2) In a next step, the vector $\mathbf{v}$ is scaled such, that the weights sum up to one ($\mathbf{v} \leftarrow \mathbf{v}/(\alpha + \beta + \gamma)$).

3) Finally, the values α, β, γ are rounded to two decimal places such, that the premise $\alpha + \beta + \gamma = 1$ is not violated.

The modified ensemble building algorithm is run on the same experimental setup as already used in Section 4.2, to allow for a comparison of this method to the previously applied method using exhaustive search. Figure 4.5b displays the result of the optimization.

For this experiment we allowed a maximum of 100 individuals to be evaluated. Within these bounds already the 43rd evaluated individual has been the best individual found in this run, this individual is marked by a white circle. Table 4.1

Model	**Ensemble weights**			**mean RMSE**	**SD**
	Gauss	Exp	Spline		
Gauss	1	0	0	11.036	0.92
Exp	0	1	0	11.349	0.82
Spline	0	0	1	11.396	1.19
by exhaustive search	0.6	0.4	0	10.95	0.85
by optimization	0.67	0.29	0.04	**10.945**	0.87

Table 4.1: Comparison of the Model results in more detail. The left column names the considered models, in the first group the base models alone, in the second group the ensembles found by exhaustive search versus the ensemble found by optimization. The second column gives the weights on the base models. Of course the base models only use one model, marked by a 1, the other models have no weight. Whereas the ensembles combine two or three models respectively. The last columns give the mean and standard deviation of the RMSE over all repetitions. The result achieved by optimzation has the best result.

shows the precise performance values for the ensembles found in comparison to the base models used. From this, it can be said, that the ensemble found by optimization is not only able to compete with the result received from exhaustive search but also is marginally better. Thus, the optimization algorithm not only can receive comparable results in comparable time but also benefits from the ability to handle a smaller step size.

4.4 N-ary Ensembles

With Section 4.3 the basis is provided for setting up a large system of heterogeneous models. In the following, the set of base models is extended to the intended size and experiments are carried out to ensure that the performance of the method does not suffer from this change. The leading questions are:

- Can the ensemble method handle the heavily extended search space for the best model combination and still find a beneficial combination?
- Are further adjustments of the ensemble building method needed due to the extension of the model set?

To answer these questions the set of base models is extended, and the method is tested. To ensure, that the method can handle the increased number of base models it is also compared to the performance of the algorithm with only three models in the set. If the method can handle the higher number of base models, a solution should be found that is at least as good as the solution found when using only three models.

4.4.1 Extending the Base Model Set

To obtain a large set of homogeneous base models all models from the SPOT package are added to the system, using their default options. This results in a system containing 13 heterogeneous models:

- Gauss, Exp, and Spline refer to the same Kriging models that are already used in the previous experiments.
- Tree, Earth, LM, RandomForest, Tgp, Esvm, MLP, neuralnet, Qrnn, and RFMlegp are now newly added to the system (cf. Chapter 2).

The set of models here is rather diverse, chosen without preselecting appropriate models and the models are not tuned but used with default settings. Thus, their performances also have considerable variances. Due to these variances in their performances the design size of the experiment is enlarged to 200 points. Herewith, the previously formulated demand, that all models should perform considerably better than the baseline predictor is relaxed such, that at least about half of the base models fulfill the demand.

Figure 4.6 shows the performances of the models on the objective function, in comparison to a simple mean predictor. It can be seen, that from the set of

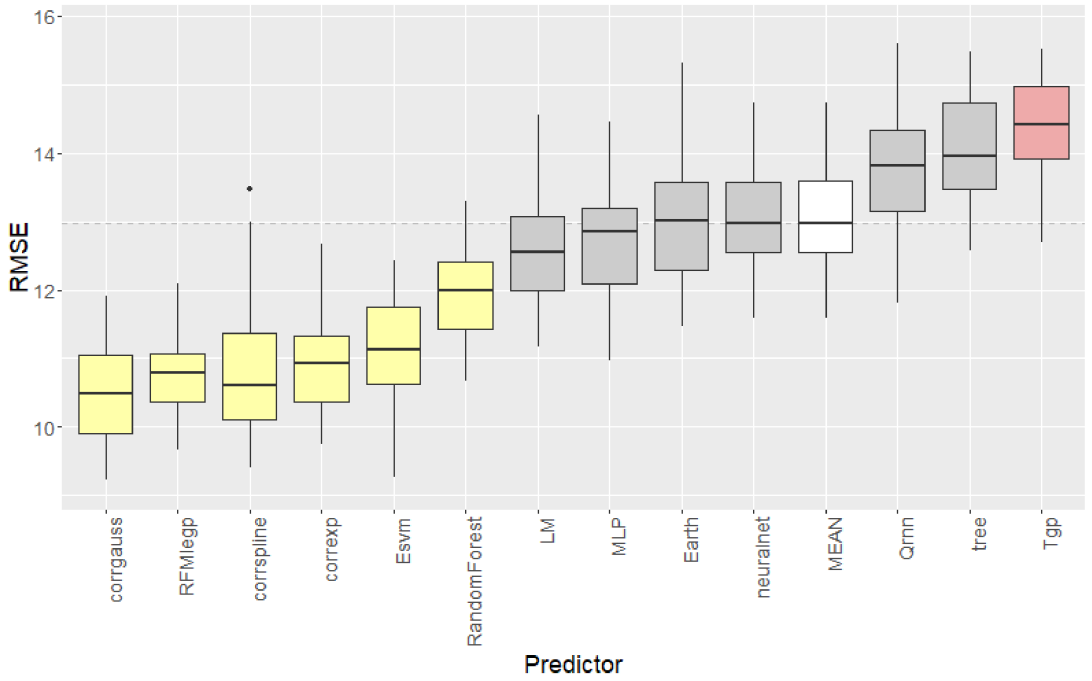


Figure 4.6: Comparison of the performance of the base models chosen for this experiment setup. Models are ordered by performance in terms of RMSE. Six base models perform significantly better than the mean predictor, these are shown in light yellow. All other models perform comparable (gray) or even worse (light red) than the mean predictor.

13 base models only six perform significantly better than the mean predictor, these models are colored light yellow. The remaining models perform comparable (gray), or even worse than the mean predictor (light red).
However, the size of the base model set and this diversity in the performances makes it harder for the search algorithm to find a good or even better solution. This fact makes this experimental setup an even more interesting setup to ensure that the search algorithm, also for larger model sets, is capable of finding a better solution if one exists.

4.4.2 Adaptation of the Evaluation Method

With this size of design, especially when evaluating more than three base models, the use of leave-one-out cross-validation is no longer practicable. Particularly with regard to problems of higher complexity an evaluation method that requires a

fixed number of model fitting processes would be preferable. Thus, for evaluation of the base models 10-fold cross-validation is applied.

Though these changes seem to be petty, they may implicate a change in the weights of the best ensemble solution. To serve as a reference, the previous experimental setup (cf. Section 4.2 and 4.3) is repeated using the larger design size of 200 points per repetition. Experiments are carried out once performing a leave-one-out cross-validation and once a 10-fold cross-validation for the evaluation of the base models. In both cases, the same set of designs is used, so that any differences in the results from the complete evaluation of these experiments solely originate from the change in the cross-validation.

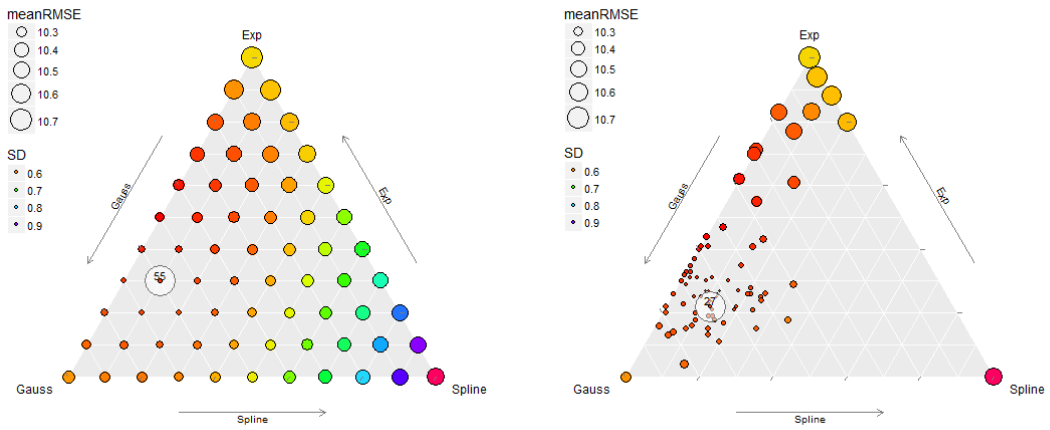


Figure 4.7: Results of the experiment carried out on a GLG objective function with a design size of 200. Base Model evaluation is done by leave one out cross-validation

The results for the experimental setup using leave-one-out cross-validation are shown in Figure 4.7. This experiment differs to the experiment presented in Section 4.2 Figure 4.5 only in the increased size of the design. In comparison to this experiment result, the best ensemble solution moved a little towards the inner area of the triangle.

Figure 4.8 shows the results for the experimental setup using 10-fold cross-validation. Here again, the best solution moved a little farther towards the inner area of the triangle.

From these results, it can be assumed that both, the increased design size as well

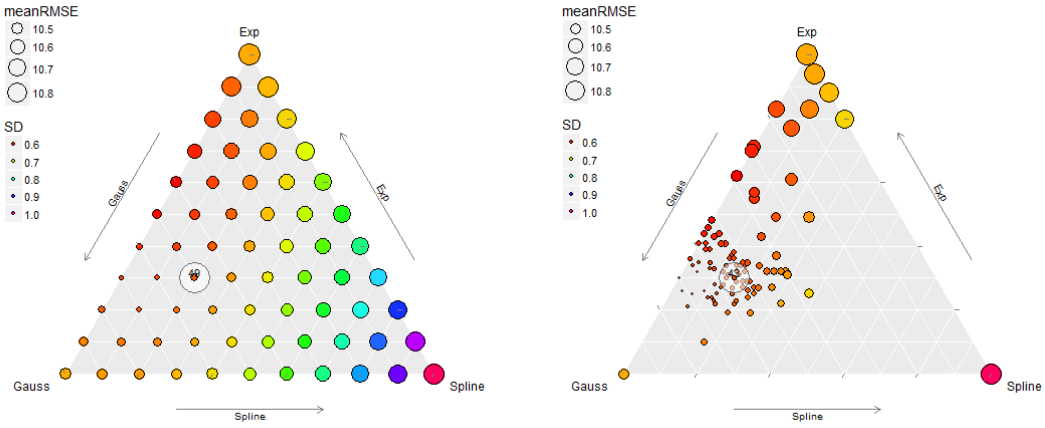


Figure 4.8: Results of the experiment carried out on a GLG objective function with a design size of 200. Base Model evaluation is done by 10-fold cross-validation

as the adjustment of the evaluation method, has an impact on the result of the experiments. Still, all experiments agree that indeed an ensemble solution exists that performs better than the base models solely. The result depicted in Figure 4.8 also serves as a reference for the following experiments using a large set of base models.

4.4.3 Performance Test Using a Large Base Model Set

With the preliminary experiment, it is already shown, that for the experiment setup carried out in Section 4.4.2, using the three kriging base models only, an ensemble combination exists, that performs better than these three models alone (cf. Figure 4.8).
Thus repeating this same experiment, only extending the set of base models while knowing that none of the new models performs better than the best of those three (cf. Figure 4.6), the search algorithm should be capable to at least reproduce a comparable solution or even find a better one. However, the ensemble algorithm as presented so far is not able to find neither a comparable, nor any ensemble solution that uses at least two of the base models. It presents Gauss alone as the best choice.

Figure 4.9 shows the progression of weight combinations that are evaluated during the search. Each line represents a single ensemble combination; the color of the

line corresponds to the optimization step it has been generated in. Each of the base models has been evaluated preceding the optimization. Therefor thirteen light red lines are shown, each featuring only one peak giving full weight to this base model. The remaining lines, representing the progress of the distribution of the weights during optimization, and therefore the different ensemble combinations tested, are entirely arbitrary. None of the random mutations seems to yield an improvement to the known best solution. The best solution that was found during the optimization gives all weight to the Gauss model. The bold white line presents this solution. The bold black line represents the best combination that has been found using the three kriging models only (cf. Figure 4.8), suggesting a mixture of these three base models, which the algorithm was not able to find.

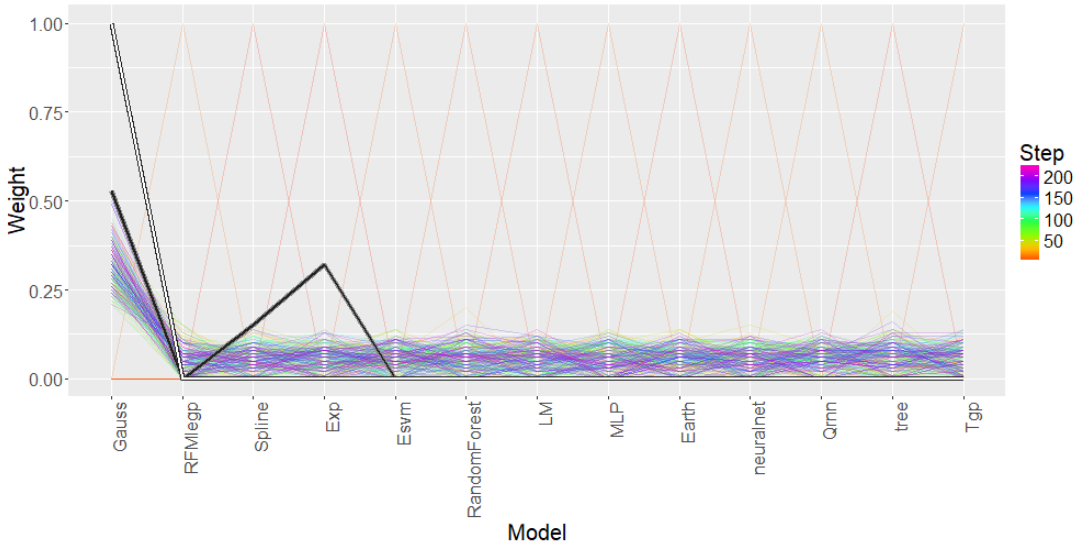


Figure 4.9: Developement of the ensemble weights during optimization. The black line marks the best ensemble found when only three models were part of the set. The white line marks the best solution found in this experiment run. The algorithm is not able to find the already known, better solution.

A possible explanation for this behavior is the number and performance of the base models added to the system. So far not only was the search space smaller but also the performance of the base models used was comparably strong. Now 13 heterogeneous base models are part of the system, only six of them are performing significantly better than the mean predictor.

This is aggravated by the fact that the search strategy performs a mutation of

the best solution by adding a random vector to the weights vector, which allows for small changes of all weights in every step (cf. 4.3). But with seven out of 13 models performing comparatively bad, the probability is high that with every mutation weights are adopted that worsen the overall performance of the new individual. Hence, the new individual generated has only very slim chances to perform better than the parent.

Possible solutions to overcome these problems might be:

1 *Restrict the number of genes that may be changed in one mutation step.* In a case where many of the base models are not beneficial for the system the chances of finding a beneficial mutation are getting larger.

2 *Minimize the overall search space by preselecting the most promising base models.* Before starting the search for the best weights, the set of models is reduced to the best performing of the models by a predefined rule.

3 *Start with a search space of a smaller dimension and enlarge during the search.* Starting the search, only using a small number of base models and then adding further base models as the search goes along, does not imply too much a priori intervention.

In the following subsections, these possible solutions are introduced in more detail and tested on the previously specified experiment setup. Possible advantages and disadvantages of the methods are discussed, and finally, all approaches are compared against each other.

4.4.4 Restriction of the Mutation

Based on the assumption that the method is not able to find a better offspring because of the rather large number of poorly performing base models in the set, the mutation is adapted to allow only a smaller number of weights to be changed in every mutation step. Doing so, it is more probable that less or none weight is given to the weaker models and with this, the probability of doing a beneficial mutation is increased.

To achieve this, a weights vector $\vec{v} = (v_1, v_2, \ldots, v_n)^T$ of n random samples of a normal distribution function with standard deviation of σ is drawn as before. However, to meet the demands of only a restricted number of weights to be changed, the smallest of these values are reset to zero, before being added to the parents' weights vector, such that only the required number of genes is changed.

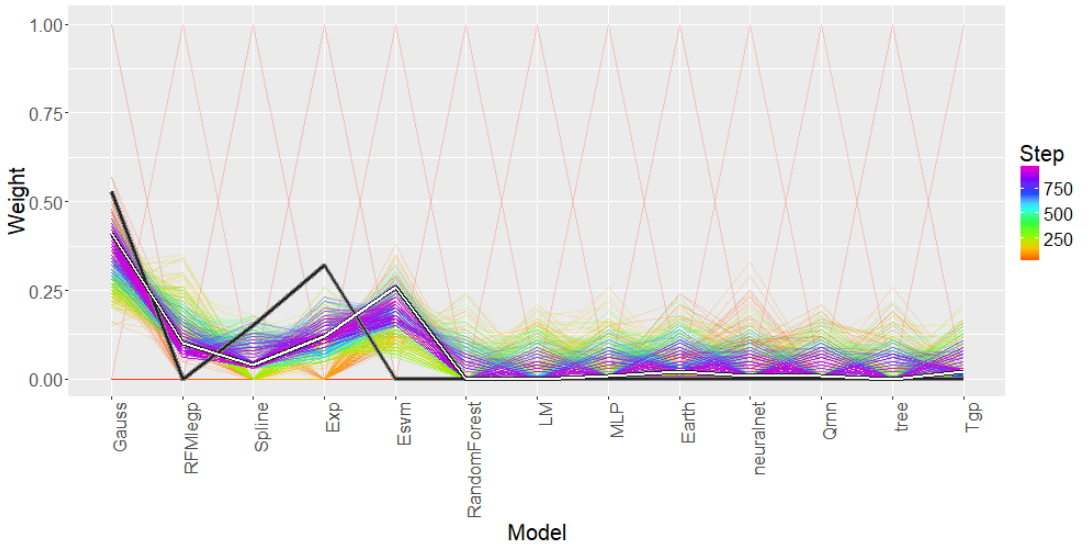


Figure 4.10: Developement of the ensemble weights during optimization using a constrained mutation step. Again, the black line marks the ensemble already found previously with only three base models available. The white line marks the best solution found during this optimization.

The resulting weights vector is then adjusted to meet the demand, to sum up to one as before (cf. Section 4.3). The approach is tested allowing three weights to be changed in each mutation step.

Fig. 4.10 shows the development of the weights distribution during the optimization process applying this approach. The lines show much more structure than before. So can be read from the column for the Gauss model that for this model a broad range of possible weights (about 20-50%) is tested in the earlier steps of the optimization and then slowly settled down to weights around 38% in the later steps of the optimization. Whereas for example for RandomForest a range of weights is tested, but none of these is an improvement to the known best, such that this model ends up with no weight. Other than the basic ES used so far, this method can find combinations that are better than the single best base model.

4.4.5 Preselection of Models

The main idea of this approach is to reduce the search space by choosing a reasonable set of base models prior to the actual optimization. Models are selected

based on their single performance during the initial evaluation of the base models. With this preselection, the search space, in terms of the number of base models, may be reduced to a smaller number of base models excluding weaker models according to a predefined exclusion criterion.

This criterion might heavily depend on the objective function and has to be chosen carefully. Choosing too restrictive a criterion might a priori exclude models that would have been beneficial contributors to the ensemble while choosing too lenient a criterion might hamper the optimization algorithm. Assuming that the algorithms inability to find a better solution arises from the large number of models performing worse than the mean predictor leads to the conclusion that the comparison against the mean predictor might be a good indicator.

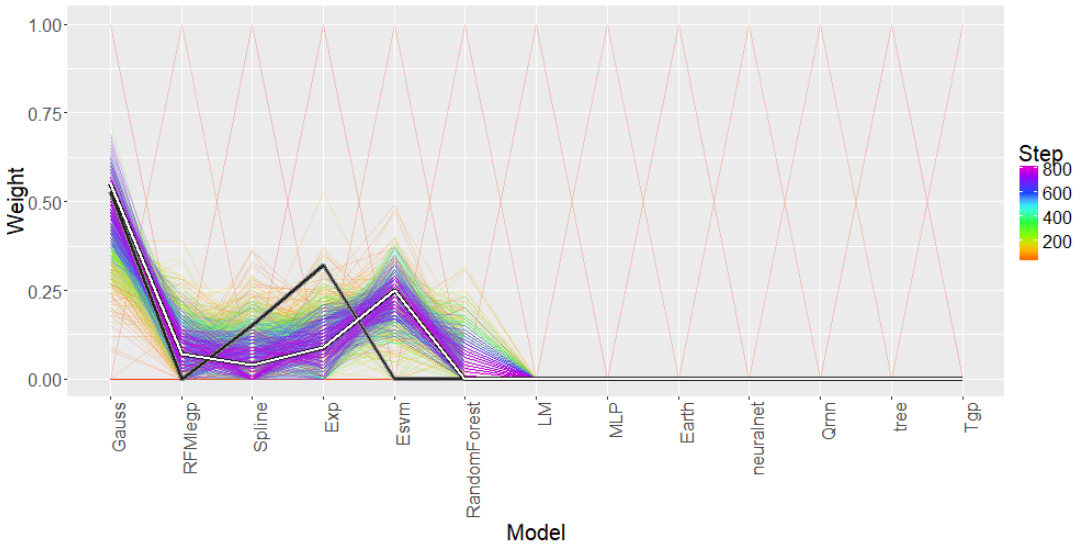


Figure 4.11: Developement of the ensemble weights during optimization using a preselected set of models only. The black line marks the ensemble already found previously with only three base models available. The white line marks the best solution found during this optimization.

This adaptation of the original ES is realized such, that models are chosen by their performance in comparison to the mean-predictor. Models with a third quantile performance, in terms of RMSE, worse than the mean predictor's first quantile performance are excluded. For the experiment setup regarded here, the base models Gauss, Exp, Spline, RandomForest, Esvm, and RFMlegp are preselected. Figure 4.11 shows the development of the weights distribution for this method. The search is restricted to the preselected models only, and the weights of these

models slowly and steadily settle down on the final solution.

4.4.6 Sequential Addition of Base Models

The previously introduced methods are realizable with small effort but also come with features that might as well be weaknesses. So is the restriction of the mutation as introduced entirely determined by the random choice of weights that is changed in each step. Whereas the preselection of models requires a selection rule to be specified and hence runs the risk to also exclude models that might have made a positive contribution to the ensemble otherwise.

This approach tries to combine the strengths of both ideas while doing without their weaknesses. The idea of this approach is to restrict the mutation to a small set of preselected models and then extending the set of models as the search proceeds. Before starting the optimization process, the base models are ranked by their performance, using the known fitness evaluation method (cf. Section 4.1). The selection of the models is based on this ranking. A small number of models is selected initially, and the remaining models are then added one by one following the ranking. Each time when adding another model to the search space, the search parameters of the (1+1)-ES are reset, and the search is started anew. In the following, the search steps between two consecutive resets are referred to as optimization tier.

Still, the question remains when the next model should be added to the system and how to proceed when the optimization process stagnates. For the experiments carried out in this section two adaptations of this idea are realized. Both start restricting the search to the three best base models and attribute a fixed number of minimum search evaluations to each tier. However, they follow different policies for the adaptation of the restriction of models and also for the handling of stagnation of the optimization process.

Sequential Addition with Stop on Stagnation

This adaptation of the approach adds the next best base model not before it succeeded in finding an improving distribution of the weights after adding the last model to the search space. With every addition of a model, the maximum number of search steps for the search is extended due to the number of active models. If no improving combination is found during the predefined minimum search interval, the search is continued until a combination is found or the search is finished.

Since the models are added to the system in order of their ranked performance, it might be more beneficial to extend the search on the actual tier than stepping on to the next. However, like the model selection approach, this may exclude lower ranked models, without ensuring if they would contribute to the system. Also, the problem of finding an improving distribution for the weights again may get harder with adding more models. But in opposition to the method of preselecting a set of base models for this optimization step no a priori decisions have to be made.

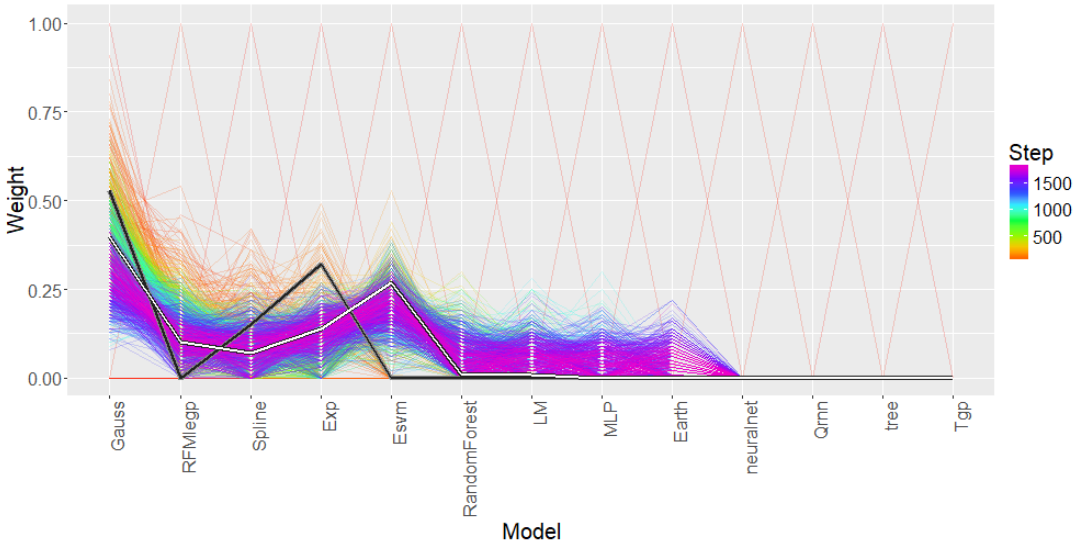


Figure 4.12: Developement of the ensemble weights during optimization while sequentially adding additional models. The black line marks the ensemble already found previously with only three base models available. The white line marks the best solution found during this optimization.

Figure 4.12 shows the development of the weights distribution for this method. The optimization process starts using only three base models: Gauss, Spline and RFMlegp. During the optimization process Exp, Esvm, RandomForest, LM, MLP, and Earth are consecutively added to the search space. After gaining no further improvement with the addition of Earth, the algorithm adds no further base models to the search space and eventually stops.

An interesting result is the performance of Esvm. The model was added to the search as the fifth model only, yet gained second most weight during optimization, while the three models added before gained less weight. Since the models are added to the system in order of their ranked performances in terms of RMSE,

this proves that a model ranked lower than another still may make a better contribution to the ensemble.

Sequential addition without Stop on Stagnation
Based on the idea, that models might be able to contribute to an ensemble though lower ranked than a model that doesn't, this method is adopted such, that every model of the set gets a chance to receive weight during the optimization.

In opposition to only extending the search space after finding an improving mutation, this adaptation of the algorithm drops the last model that was added after a fixed number of iterations, when the addition was not beneficial. Instead, it continues with the next best model. The algorithm stops when all models were at least for some time part of the search space, and after the last addition the search is stagnating or the maximal iteration count is reached.

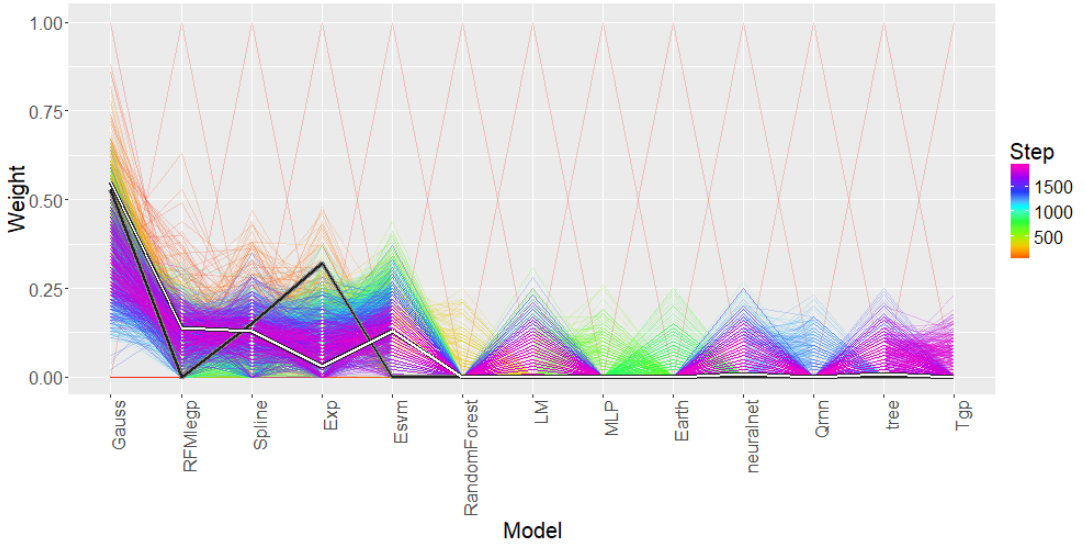


Figure 4.13: Development of the ensemble weights during optimization while sequentially adding additional models, and dropping non-beneficial models again. The black line marks the ensemble already found previously with only three base models available. The white line marks the best solution found during this optimization.

As before the algorithm starts only using Gauss, Spline, and RFMlegp. Then Exp and Esvm are added. RandomForest is added next but dropped again before adding LM. Next MLP and Earth are added but also dropped again. After that,

neuralnet is added and kept, then Qrnn but dropped again. At last tree and tgp are added and kept until the optimization finishes.
During the optimization, all base models were part of the search space for some time. Figure 4.13 shows the development of the weights during the optimization It can be seen, that some models were part of the system for some time during optimization and then dropped again. These models only have peaks in a small color range (i.e., RandomForest is peaked only by some yellow lines. Yellow is attributed to the earlier steps in the optimization).
The course of the optimization shows that indeed a model may make a beneficial contribution to the ensemble although a better-ranked model earlier was not able to do so.

4.4.7 Comparison of the Different ES-Adaptations

Results so far illustrate the behavior of the different approaches and show that all are able to overcome the initial inability to find an ensemble combination.

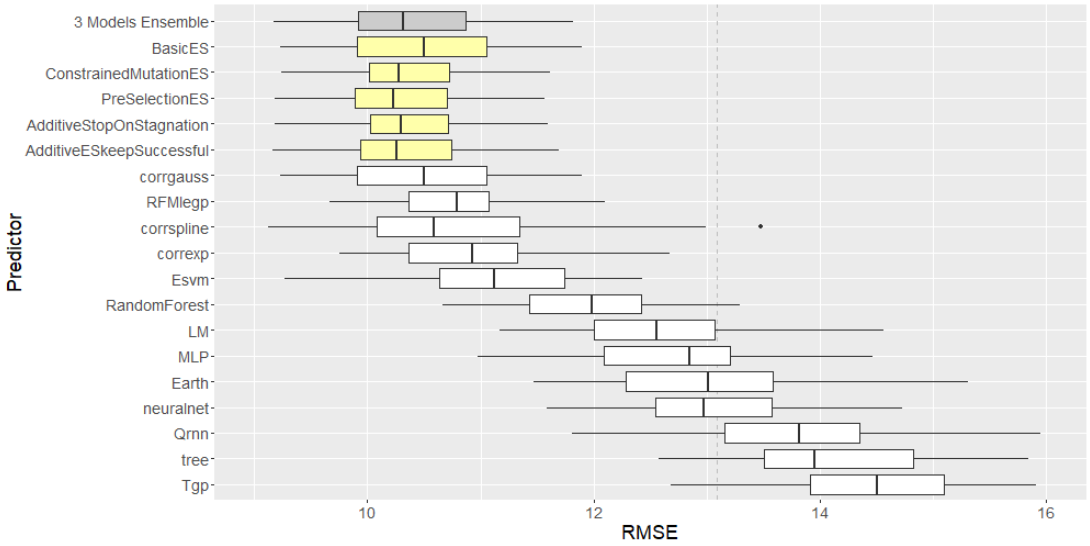


Figure 4.14: Comparison of the results found by the different optimizers with the base models and the ensemble built when only 3 base models were available.

Figure 4.14 shows a comparison of the performances of the base models (white), the ensemble model found with only three base models available (gray), and the ensembles found with the adapted search strategies using 13 heterogeneous models (yellow). A dashed line marks the mean performance of the baseline predictor.

The basic ES without any adaptations was not able to find an ensemble combination; therefore its result corresponds to the result of the Gauss model. It can be

Model	**Ensemble weights**													**mean RMSE**	**SD**
	Gauss	Exp	Spline	tree	Earth	LM	RandomForest	Tgp	Esvm	MLP	neuralnet	Qrnn	RFMkgp		
Gauss	1	0	0	-	-	-	-	-	-	-	-	-	-	**11.216**	0.87
Exp	0	1	0	-	-	-	-	-	-	-	-	-	-	11.621	0.71
Spline	0	0	1	-	-	-	-	-	-	-	-	-	-	11.489	1.06
exhaustive	0.5	0.3	0.2	-	-	-	-	-	-	-	-	-	-	10.449	0.62
optimization	0.53	0.32	0.15	-	-	-	-	-	-	-	-	-	-	**10.443**	0.62
constr. mut.	0.41	0.12	0.04	0	0.02	0	0	0.02	0.26	0.01	0.01	0.01	0.1	10.369	0.59
preselection	0.55	0.09	0.04	0	0	0	0	0	0.25	0	0	0	0.07	**10.334**	0.61
additive	0.4	0.14	0.07	0	0	0.01	0.01	0	0.27	0	0	0	0.1	10.355	0.60
keep succ.	0.55	0.03	0.13	0.01	0	0	0	0	0.13	0	0.01	0	0.14	10.376	0.61

Table 4.2: The table compares the results for the best base model and the different ensemble approaches. The left column names the considered models, in the first group the best performing base model only, in the second group the ensembles found by exhaustive search versus the ensemble found by optimization. The third group gives the results for the ensembles using the different adaptation strategies for the ES. The second column gives the weights on the base models. Since in the previous experiment only three base models were available, the others are crossed out. The last columns give the mean and standard deviation of the RMSE over all repetitions. The group ensembles using the adaptation strategies show the best results and the smallest standard deviations.

seen, that the ensembles found by the adapted search strategies can compete with the previous best solution (gray). All solutions found are performing comparably good.

The precise results for the ensemble solutions found using the adapted ES in comparison to the ensemble solution found with only three base models available (cf. Figure 4.8) and also to the best performing base model are displayed in Table 4.2.

From these results, it can be said that the different adaptations of the algorithms can handle the large search space, but it cannot be determined if one of these solutions is remarkably better than the others.

4.5 N-ary Ensembles on Higher Dimensional Physical Functions

So far a reliable method for ensemble building has been introduced and realized in different adaptations. On the experiment conducted all adaptations show comparable performance. To allow for a better comparison of the different adaptations of the method, in this section, the performance of the different adaptations are compared against each other and against the performance of the best performing base models alone.

Moreover, the initial step size σ_{init} of the (1+1)-ES is reconsidered. It is assumed, that with an increase of the dimension of the search space also the initial setting for the step width should be increased.

Therefore, the leading questions of this section are:

- Is one of the adaptations of the ensemble method preferable to the others?
- Moreover, should the initial step size σ_{init} adapted according to the dimensionality of the search space?

To answer these questions, additional experiments are carried out on a set of physical objective functions of higher dimensions (cf. Section 2.3.3).
For these experiments, the experimental setup has, in general, been left unchanged. Of course, for each experiment, the initial design size has been adjusted to 110, due to the higher dimensional functions. For the wing weight function, the initial design size has been set to 280, to ensure, that at least half of the base models perform better than the mean predictor.
Additionally, the experiments are used to get some insights into the effect of the initial step width σ_{init} on the optimization result. Therefore, each experiment is carried out twice with different values for σ_{init}. Here $\sigma_{init} = 0.16$, as used in earlier experiments is compared to $\sigma_{init} = 0.37$.

Figure 4.15 gives an overview of the performances of the different adaptations of the ensemble building method on the four physical functions. Each of the plots additionally shows the performance of the best performing base models. Further plots depicting the full comparison of all base models are also available in the Appendix (cf. Appendix A, Figures A.1-A.2).
From these plots, it can be read that each of the ensemble method adaptations can compete with the best performing base model. The Additive methods seem to have a small advantage over the remaining methods on the otl-circuit function and

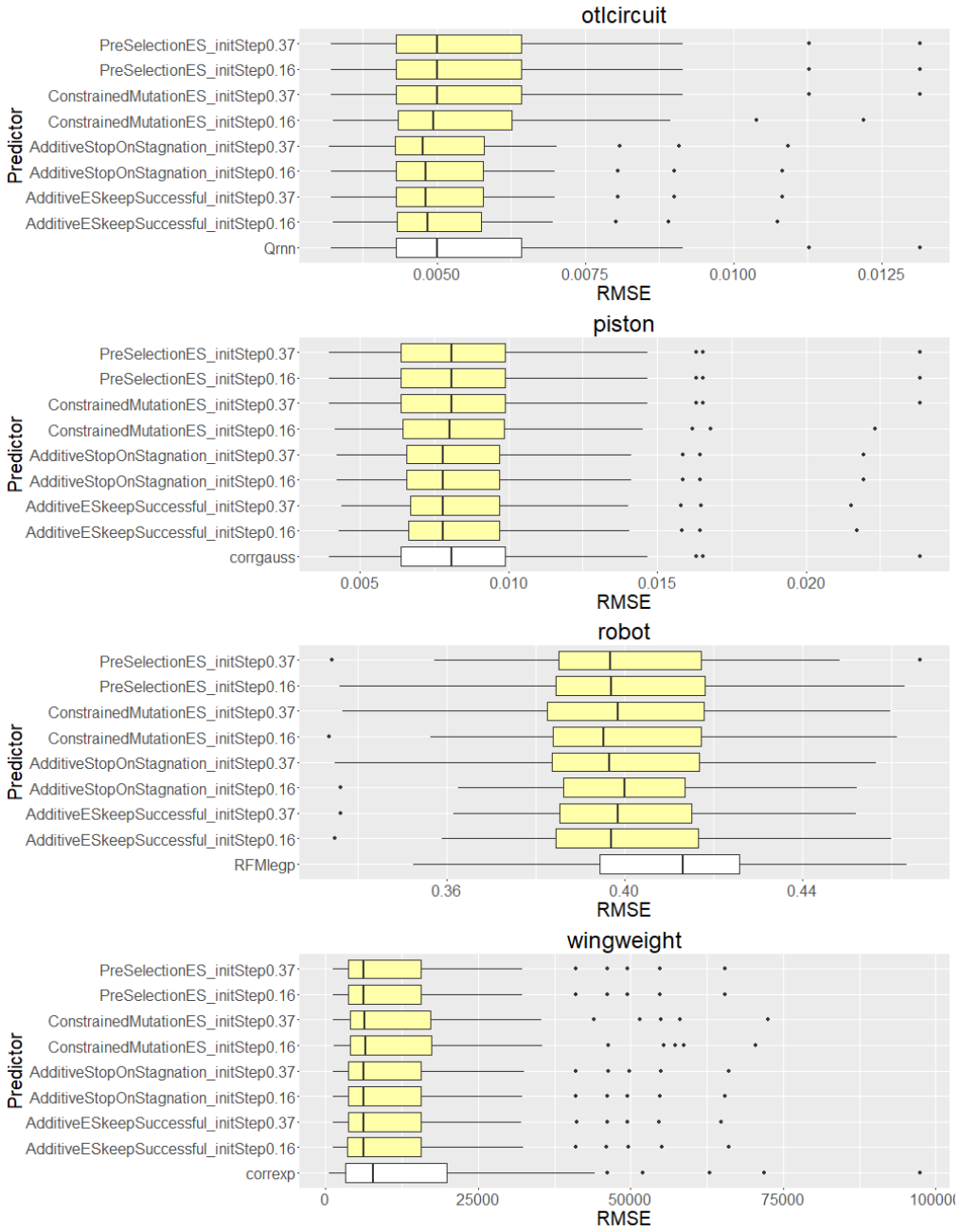


Figure 4.15: Comparison of the performances, in terms of RMSE, of the different adaptations of the ensemble building method and the best base model on the four physical functions. Ensemble results are colored yellow, the base model result is shown in white.

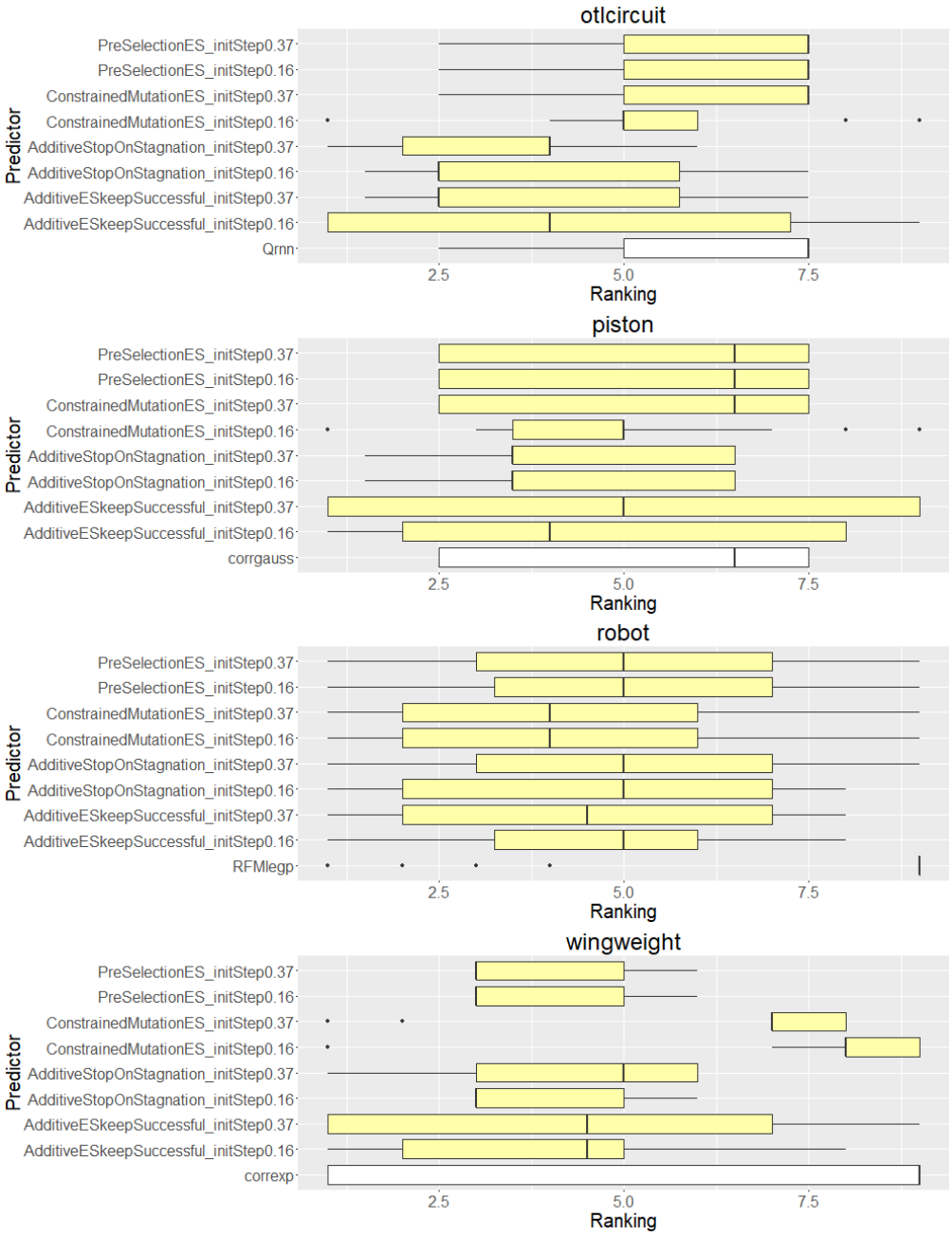


Figure 4.16: Comparison of the performances of the different adaptations of the ensemble building method and the best base model on the four physical functions. Results are repetition wise ranked. Ensemble results are colored yellow, the base model result is shown in white.

a smaller standard deviation on the piston function. On the robot arm function and the wing weight function the ensemble adaptations even show slightly better results, than the best base model.

To allow for a better comparison, Figure 4.16 shows the results of these experiments with the RMSEs achieved repetition wise ranked. From these results, it can be read, that the additive methods, in terms of ranking, on the otl-circuit function and the wing weight function perform better than the methods using pre-selection of the models or a constrained mutation.

FUN	**AdditiveESkeepSuccessful**		**AdditiveStopOnStagnation**	
	$\sigma_{init} = 0.16$	$\sigma_{init} = 0.37$	$\sigma_{init} = 0.16$	$\sigma_{init} = 0.37$
otl-circuit	0.0051659 (± 0.00142)	**0.0051656** (± 0.00144)	**0.0051656** (± 0.00144)	0.0051669 (± 0.00146)
piston	0.0086486 (± 0.00323)	0.0086670 (± 0.00319)	**0.0086355** (± 0.00326)	**0.0086355** (± 0.00326)
robot	0.4007355 (± 0.02708)	0.4009263 (± 0.02558)	0.4009612 (± 0.02580)	0.4011576 (± 0.02687)
wingweight	13492.530 (± 15246.0)	**13457.750** (± 15117.3)	13474.070 (± 15180.1)	13505.770 (± 15265.1)
Sum RankMN	13	**10.5**	11	17.5
Sum RankMD	14	19.5	16	**5.5**

FUN	**ConstrainedMutationES**		**PreSelectionES**	
	$\sigma_{init} = 0.16$	$\sigma_{init} = 0.37$	$\sigma_{init} = 0.16$	$\sigma_{init} = 0.37$
otl-circuit	0.0054833 (± 0.00170)	0.0056067 (± 0.00192)	0.0056067 (± 0.00192)	0.0056100 (± 0.00192)
piston	0.0086966 (± 0.00338)	0.0087677 (± 0.00354)	0.0087677 (± 0.00354)	0.0087677 (± 0.00354)
robot	0.4007446 (± 0.02794)	**0.4003984** (± 0.02682)	0.4015957 (± 0.02743)	0.4012452 (± 0.02815)
wingweight	14847.550 (± 16888.8)	14398.630 (± 16628.1)	13474.070 (± 15180.1)	13474.070 (± 15180.1)
Sum RankMN	21	22	25	24
Sum RankMD	19	27	22	21

Table 4.3: The Table gives the results of the different adaptations of the ensemble building method on the four physical functions and for two different settings of σ_{init}. Shown is the mean RMSE with standard deviation (in brackets). In the two last rows, the sums over the ranks of mean and median RMSE are given.

Table 4.3 lists all results in more detail. For each combination of ensemble building method adaptation with a given σ_{init} and a function, the mean RMSE with standard deviation (in brackets) is given. These mean RMSE values are function-wise ranked and summed up to obtain the sum of ranks over the mean RMSE values that are given in the second last row. Median RMSE values are considered accordingly. The sum of ranks over the median RMSE values is given in the last row.
For example, `AdditiveESkeepSuccessful` with an initial step width σ_{init} of 0.37 is ranked first on the Wing weight function (1) and shares the first place on the otl-circuit function (1.5), while on the Piston and the Robot Arm function it is ranked fourth place each (4; 4). In total it achieved a sum of ranks value over the

mean RMSE of 10.5, which is also the best value here.
Best evaluation in terms of sum of ranks over the median RMSE is achieved by `AdditiveStopOnStagnation` with an initial step width σ_{init} of 0.37.

The results suggest that an additive approach with an initial step width σ_{init} of 0.37 might be the best choice for experiments of this dimension. However, for more distinct results further experiments have to be carried out.

Additional plots for the development of the weights during optimization for the different methods using an initial step width σ_{init} of 0.37 on the otl-circuit function can be found in the Appendix (cf. Appendix A, Figures A.3 and A.4.

4.6 Conclusion

The primary goal of this chapter was to create an ensemble building strategy that works reliably and as accurately as possible on arbitrary objective functions. A method was aimed for, that can compete with the best performing base model for each considered function. In the Sections 4.1 and 4.2, based on the Chapter 3, a strategy was developed, which builds ensembles using convex linear combinations of the models' predictions. The method was thoroughly analyzed, and convex linear combinations showed to be an ideal choice for combining models.
The most important insights made and advantages recognized are:

- Due to the convex linear combination that is used for combination, in terms of RMSE, the ensemble cannot perform worse than the weakest base model.
- The ensemble can perform better than the base models when compensating opposing prediction errors.
- A CCM is favored over a base model only if the overall fit of the ensemble model is actually better (in terms of RMSE), than the overall fit of both base models.
- The nature of the combination is intuitive and interpretable.
- The linear convex combination of predictions for a given set of weights is easy to compute.

In Section 4.3, with the step from an exhaustive search to optimization using a (1+1)-ES, the basis was provided for setting up a large system of heterogeneous models. Then, in Section 4.4, further adaptations are made, and the set of base models is extended to the intended size. First experiments showed that the

method was not able to handle the extended search space, which lead to further adaptations of the ensemble building method. Four different approaches were introduced and tested, and it was shown that the additional approaches are at an advantage in some cases. Another insight gained in these experiments is that indeed a model can make a beneficial contribution to the ensemble although a better-ranked model was not able to do so earlier.

In Section 4.5 additional experiments on physical functions were performed to allow for further comparison of the different approaches on functions of a higher dimension and with relation to real-world applications. Moreover, the question should be answered if, with an increasement of the search-space, also a larger value for the initial step width of σ_{init} would be recommendable. Still, the question which adaptation of the ensemble method is the best and if the initial step width σ_{init} should be adapted to the dimension of the search-space could not be answered satisfyingly and would require further analysis.

To a great extent, this chapter (Sections 4.1 through 4.3) is based on the article *"Building Ensembles of Surrogates by Optimal Convex Combination"* by Friese et al. [73]. Major parts from the original article were adopted verbatim. Of course, the text was adapted to fit the structure and notation of this thesis.

Chapter 5

Automated Model Selection in SPO

In Chapter 4 a method was developed that automatically builds an ensemble from a large set of heterogeneous models by computing an optimally weighted convex linear combination. It was shown that regarding regression tasks, none of the possible ensembles could perform worse than the weakest of the available base models. Moreover, it is possible that an ensemble combination exists that performs even better, in terms of RMSE, when fitting the regarded data, than any of the base models. Additionally, a method was supplied to automatically determine the best performing model from the possible ensemble combinations. In this chapter, the CCM building method is adapted for the use in SPO. Since the overarching goal of this thesis is to release the user from the burden to select the right surrogate model, also and especially in time-consuming optimization tasks, the driving questions are:

- Can the CCM building method adapted such that it works reliably and accurately in SPO?
- Can optimized (and dynamically adapted) CCM compete with fixed base models in SPO?
- How does the CCM approach compete with approaches that dynamically update the surrogate model selection or apply the same ensemble throughout the entire optimization process?
- Is it possible to adapt the method such that it performs feasible, in terms

of calculation time, despite a large number of models involved?

The CCM already showed good results on regression tasks. A central focus of this chapter lies on the characteristics of the SPO that come with the sequential step. Some of these characteristics may be taken advantage of to improve the algorithm further. Other characteristics make it harder for the CCM to perform reliably. Possible solutions to approach these difficulties are introduced, analyzed and incorporated into the ensemble building method. Additionally, the algorithm is further adjusted such that it functions reliably even if one or more models of the set fails during the optimization process.

This chapter is structured as follows. In Section 5.1 the CCM method is introduced in detail, and further adaptations needed for the application in SPO are discussed and implemented. In Section 5.2 the dynamically adapted CCM method is then thoroughly tested for its performance during sequential parameter optimization on a large set of diverse objective functions. Also, it is compared to the base models and two strong ensemble competitors. Finally, the results of the experiments are presented and discussed.

5.1 Adapting to the Sequential Step

The experiments carried out in Chapter 4 were all regression tasks of a static nature. A given objective function had to be fitted via a given set of points generated by a space-filling design. For the evaluation of the fit of the model to the function, the RMSE was calculated. For a reliable result, the experiment has been repeated several times without any changes in the experiment setup besides the positioning of the points of the design that are to a given extent chosen randomly.

With the step from this static experiment to a sequential experimental setup, some alterations of the method have to be done to make it work and to ensure it functions efficiently and reliably.

The most noticeable change that has to be addressed is the fact that in each sequential step only one set of data is available for evaluating the models and finding the best ensemble combination. So instead of running repetitions, the evaluation is now done in a single 10-fold cross-validation step. The single fitness value resulting from this evaluation is crucial for the choice of the best ensemble.

The main steps of our proposed CCM building method are presented in Algo-

rithm 2 which, up to this degree of detail, is self-explanatory. The approach takes a set of n points that have been evaluated with an expensive black box evaluation function; they are denoted with $(x_1, y(x_1)), \ldots, (x_n, y(x_n))$. Then, it minimizes the cross-validation error over the set of CCM, thereby performing a parameter optimization of the model weights over the simplex $\{\alpha \in \mathbb{R} | \sum_{i=1}^{n} \alpha_i = 1, \alpha_i \geq 0\}$.

Algorithm 2: CCM Ensemble-building using weighted 10-fold cross-validation

Data: Previously evaluated n data-points $(x_1, y(x_1)), \ldots, (x_n, y(x_n))$;

Result: CCM function $\hat{y} : \mathbb{R}^d \to \mathbb{R}$

1: **begin**

2: For all base models, compute cross-validation on previously evaluated n data-points;

3: Search for best weights using box-constrained optimization and data generated in Step 2;

4: Generate ensemble-predictor function $\hat{y}$ using best weights;

To apply these models in sequential optimization requires several adaptations, e.g., to adjust the models continuously in the presence of dynamically changing and non-uniform data sets. Several adaptations will be applied in order to integrate the CCM approach in sequential parameter optimization. These are:

- Periodically rebuilding of models and temporarily suspending models, to take dynamical updates into account.
- Local density weighted cross-validation, to deal with non-uniform point distributions.
- Adaptation of (1+1)-ES used for weight optimization, to deal with large ensemble sets.

Next, these adaptations will be introduced one-by-one before in the next section further experimental justification of their usefulness is added.

5.1.1 Building Intervals and Suspension of Models

The characteristic of the SPO that is of highest interest for the adaptation of the presented approach is the steadily expanding dataset. With each step of the

optimization additional points are added to the set of known data points D. With the growth of the set of known points also the knowledge about the underlying objective function grows. Characteristics about the function that can be read from the known points may be changing over time, and the ensemble should adapt to that change. Therefore, it should be beneficial to update the weight combination over time to adopt the model to features of the objective function that were not known before.
An additional parameter τ is introduced to the method specifying a fixed number of steps, such that the proposed CCM building method updates the ensemble combination in every τ-th step of the optimization.
This parameter controls the ability of the ensemble to adjust itself, in terms of giving more weight to a more appropriate model during optimization. We suppose this ability, in general, to be beneficial for the performance of the ensemble but the choice of the parameter has to be taken carefully. While choosing too large a value for the parameter reduces the ensembles ability to adapt to changes, it is not given that an adaptation of the ensemble in every step of the optimization is still beneficial. Also, the ensemble building process is rather expensive in terms of computation time. Though in real-world optimization the objective function is the expensive part, a reasonable calculation time for the model may still be desired and thus also might be considered when specifying the rebuilding interval.

The computation time of an SPO process when using a CCM not only depends on the frequency with that the ensemble is rebuilt but also on the number of models that are part of the set. While a large set of heterogeneous models is in general desirable, the evaluation of a large set of base models also takes its time. Moreover, not all models might make a beneficial contribution to the ensemble at any time of the optimization. We expect only a smaller subset of the models to be a beneficial contribution to the ensemble, and with the set of known data steadily growing, this choice of beneficial models may steadily shift. Based on this assumption the method is adapted to allow for temporary suspension of models that do not contribute to the ensemble. A possible suspension of a base model is checked after every CCM building process, whereas re-inclusion of these models is done every λ-th step of the optimization.

It is obvious that the model building interval τ has a large influence on the computation time since the majority of the time is needed for the cross-validation. However, the influence of the suspension interval λ on the calculation time may vary since it depends on the number of models that are suspended.

These considerations embed the actual CCM building part described in Algorithm 2. In case of the actual step being a λ's step, all suspended models are

added to the set again. The CCM building is then started if this is a τ's step of the optimization and at least two models are not suspended. If all but one model are suspended, this model is the new CCM response.
Potential suspension of models is considered only after completion of the CCM building process. Models that do not contribute to the ensemble, and therefore gained no weight in the CCM building process, are suspended.

5.1.2 Local Density Weighted Cross-Validation

The main step of the CCM building is the cross-evaluation that is carried out on the previously evaluated n data-points to evaluate the fit of the base models (cf. Algorithm 2, Line 2). This step runs the risk to be the most time-consuming step of the CCM building method, but it is also a crucial step of the Algorithm since the whole CCM building process depends on the evaluation of the models that is done in this step.

Methods to gain control over the computation time were already introduced in Section 5.1.1. However, additional precautions to ensure a reliable performance are also introduced in this step. Since the overall goal is to allow for the incorporation of a large set of heterogeneous models with characteristics that may not be known to the user, these models may also show unreasonable calculation times or do not perform reliably. To encounter such problems models may be excluded from the set during the optimization process. Models that exceed a predefined time limit, fail or return defective predictions (i.e., NAN[1] values) during the fitting process are immediately excluded from the system.

However, the more critical aspect is the appropriate evaluation and weighting of the models since also the performance of the CCM depends on it. For the experiments carried out in Chapter 4 the use of the RMSE lead to good results. Nonetheless, it has to be considered that for these experiments the experimental setup was chosen such, that it enables the best circumstances for evaluating the fit of the models to the objective function consistently over the entire region of interest by choosing a space filling design for the automatic generation of the experimental data.

In general, SPO starts the optimization process by evaluating an initial set of points which are derived from a space filling design. However, it is expected

[1] NAN is the abbreviation for 'Not a Number' which is, in general, return when division by zero is attempted.

that during the progress of an SPO evaluated points will cluster at local optima. Without taking this into account during cross-validation, we risk putting too much emphasis on prediction errors close to those local optima, which leads to over-fitting in these areas.

A possible solution to this problem would be to exclude points from the data set before evaluation, to ensure an even distribution of the data for cross-validation. However, for this approach, it would have to be specified how many and which points have to be excluded from the data set for cross-validation also risking to exclude important data from the set.
Instead, to overcome the problem, for the evaluation of the overall fit of the model, we propose to reduce the importance of points that are located in areas with a very dense neighborhood by weighting their squared errors. Weights are depending on the density of the close neighborhood, and therefore on the position of the regarded point. Hence weighting occurs evenly and without harsh steps in weights between points that are close to each other since they also share parts of the same neighborhood.

The resulting weighted Root Mean Square Error (wRMSE), which is implemented as quality indicator, is calculated as follows:

$$wRMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\beta_i(y_i - \hat{y}_i)^2}$$

The weights $\beta_i \in [0,1]$ that are applied to the prediction errors $(y_i - \hat{y}_i)$ at the points $\mathbf{x}_i \in \mathbb{R}^d,\ \ i = 1,\ldots,n$, are derived from the density of the point's direct neighborhood. For the calculation of this density, the k closest neighbors of each point are utilized.

The density ρ_i of a point i is then calculated as the median distance to these neighbours:

$$\rho_i = \text{median}\left\{ \sqrt{\sum_{j=1}^{d}(\mathbf{x}_i^j - \mathbf{x}_l^j)^2} \mid l = 1,\ldots,k \right\}$$

Since only points are to be weighted that are located in areas with a dense neighborhood, density values that are exceeding the overall mean density are truncated to the mean value. This is done to ensure that all points with a neighbourhood of mean density, or sparser, get full weights in the cross-validation.

In order to obtain the weights $\beta_i \in [0,1]$, the determined density values ρ_i are normalized to the $[0,1]$ range, by computing $\beta_i = \rho_i / max\{\rho_0,\ldots,\rho_n\}$. The lower

bound has not to be taken in account here, since the density values ρ_j have to be positive per definition because of the distances used for calculation and it is not intended to force zero weight on points with the highest density.

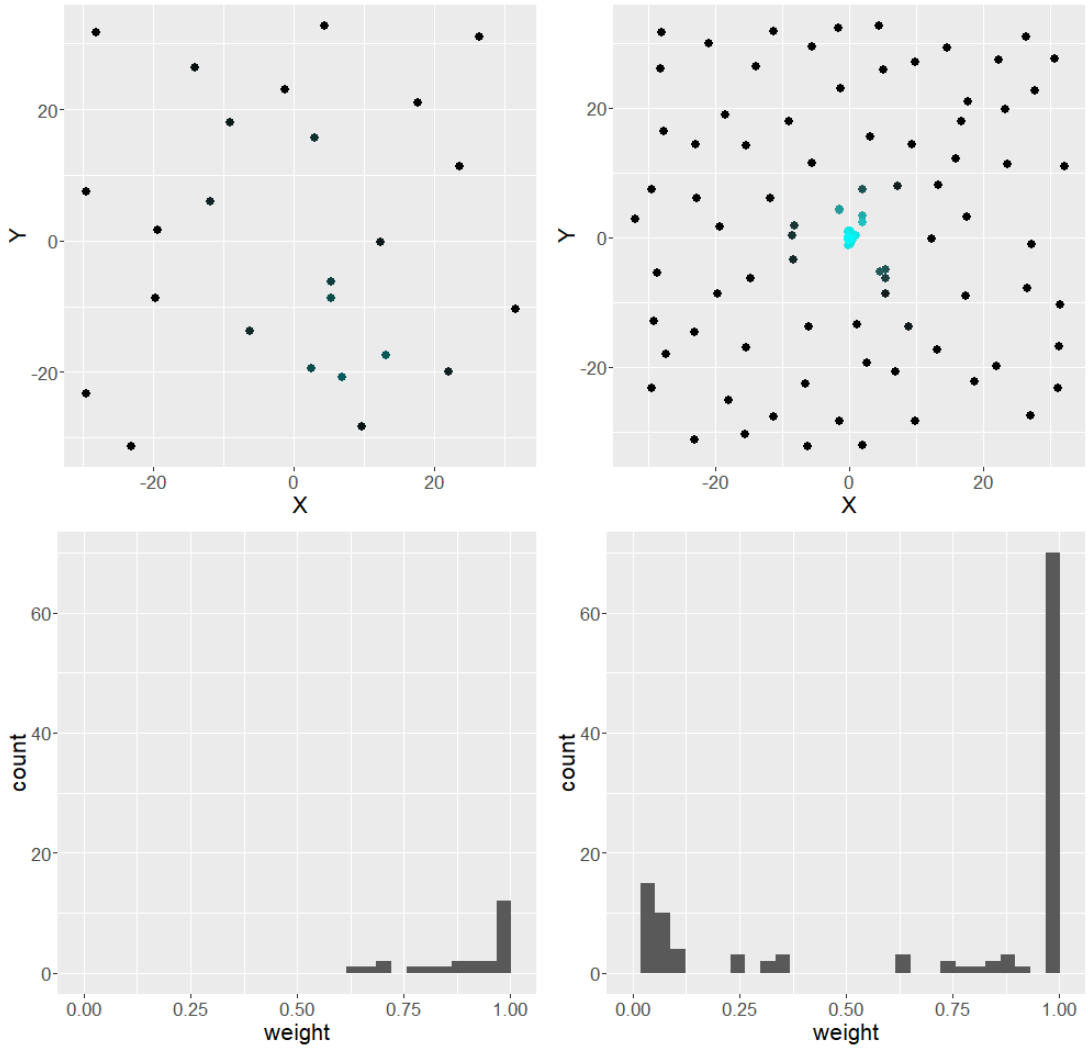


Figure 5.1: The plots show the impact of the weighting procedure on the points at the beginning and the end of an optimization process on a 2D Ackley function. Points that are colored black are fully taken into account, and lighter blue points are weighted. The lower row shows the related distribution of weights used. It can be seen that points near the cluster are rigorously weighted.

Applying these weights to the squared errors of the RMSE during cross-validation ensures that all points with a neighborhood not denser than the mean-density are

considered with full weight and only points located in denser neighborhoods are weighted according to the density of their neighborhoods.

At the beginning of the SPO, this approach has only a small impact on the weighting of the predictions during cross-validation since initial designs of experiments preferably are space-filling and as such avoid clustering of points. Only with the optimization process proceeding points will eventually cluster at local optima. The denser this clustering will be the higher gets the impact of the weighting during cross-validation on points located in or near clusters.

Figure 5.1 shows the effect of the weights applied to data points during an sequential optimization on a two dimensional Ackley function. The plots belong to two different steps of the same optimization process. In the left column, the situation after the first sequential steps of the optimization is shown, only five additional points were evaluated. In the right column, the situation after the optimization process has stopped is shown.
In the upper row, the points are depicted; the color indicates the applied weights. Here a point colored black means no weighting or a full weight of one respectively, while a blue colored point means that this point was weighted. In the lower row, the distribution of the weights is shown.
It can be seen that in the early steps of the SPO the weighting has nearly no impact, some points are only slightly weighted while the majority of the points get full weight. The histogram in the lower row also confirms this.
In the last step of the SPO altogether 120 points were evaluated and clustering has taken place in the center of the regarded area. The points that are densely clustered also are severely weighted, while points that are not located near a cluster get full recognition for the evaluation. Again, this can be read from the histogram, which shows that the majority of the points gets full weight, but a smaller subset of the points is rigorously weighted.

5.1.3 Optimization of the Ensemble Weights

After finishing the cross-validation of the base models, we use a (1+1)-Evolutionary Strategy (ES) with 1/5th success probability rule for step size adaptation to find the best ensemble weights (cf. Algorithm 2, Line 3).

In Section 4.4 several adaptations of the algorithm were proposed to ensure that the method functions properly on large sets of base models. From these adaptations, the additional approaches were at an advantage in some cases. The experiments also proved that a model can make a beneficial contribution to an

ensemble although a better-ranked model is not able to do. Therefore, in the experiments carried out in this chapter the adaptation introduced as 'Additive ES without stop on stagnation' (cf. Section 4.4.6) is used. Algorithm 3 depicts the main steps of the modified $(1 + 1)$-ES method which are introduced in more detail in the following.

Algorithm 3: Adapted (1+1)-evolution strategy (ES) with 1/5th success probability rule for step size adaptation and

Data: Initial population $P \subset [0, 1]^s$
Available models $\mathbf{av} = (av_1, \cdots, av_s) \in \{0, 1\}$
Result: Complete population P

```
1: begin
2:     Choose the best individual as the first parent individual ;
3:     Specify initial search space (active models) act = (act_1, ..., act_s) ∈ {0,1};
4:     Initialize ES max step count C_max ← 5|act|^2 ;
5:     Initialize ES step size adaption interval φ ← 5|act| ;
6:     Initialize ES initial step size σ ← σ_init;
7:     while Stop criteria not met do
8:         if Steps for this level exhausted then
9:             Adjust search space;
10:            Reset ES step size σ ← σ_init ;
11:            Adjust ES max step count C_max ← C_max + 5s^2 ;
12:        Generate offspring by perturbation of all active α_i with a normal
           distribution with standard deviation σ (step size);
13:        Evaluate offspring;
14:        Select new parent individual by choosing the offspring or the current
           parent depending on the objective function value;
15:        if Step count is multiple of φ then
16:            Adjust step size σ using 1/5th success rule;
```

The search starts with an initial population P, e.g., the corners of the search space, representing the active base models and a vector $\mathbf{av} \in \{0, 1\}^s$ of flags, where s corresponds to the number of base models in the set, stating which models are currently suspended ($\mathbf{av}_i = 0$) and which models are to be used for

this optimization process ($\mathbf{av}_i = 1$), and thus defining the search space. For each base model, all predictions made during cross-validation, as well as each model's wRMSE value is known. From these data, any CCM given by a specific weights combination can be derived.

Due to the incremental data update that is characteristic for sequential optimization, the position of the optimal weight combination in the current iteration is likely to not deviate much from the one obtained in the previous optimization. Therefore, the previously obtained solution is also added to the initial population P.

The individual that is used as a starting point for the search is chosen by its fitness value (Algorithm 3, Line 2). However, the search favors a combination of models over a single model only if its overall fit in terms of wRMSE is strictly better. Therefore, the ensemble from the previous optimization step is also chosen only if strictly better.

To enable the search to handle large sets of base models, the search starts on a smaller subset of the available models (Algorithm 3, Line 3) and then stage-wise extends or adjusts the search space throughout the search. Which models are currently active is specified in an additional vector $\mathbf{act} \in \{0,1\}^s$ of flags.
For the initial subset, the base models that are part of the individual chosen as the starting point are automatically added. If needed, additional base models are chosen by their fitness values to ensure, that at least three base models are part of the initial search space.

Then, the remaining parameters for the ES are initialized (Algorithm 3, Line 4-6). Finally, the main optimization loop is started. This loop is terminated solely when all available models ($\mathbf{av}_i = 1$) were part of the search space, at least for the length of the stage that they were added in. As mentioned before, the actual search is performed stagewise, doing restarts with different subsets of models. With each stage the search space is adjusted, the parameters of the ES are reset and if needed adjusted to the new search space. For this, the last model added to the search space is removed again if its addition did not lead to a better solution. However, the initial three models are never removed from the search space. Then the best performing model that has not yet been part of the search space is added. The number of search steps granted for this stage depends on the number of models that are part of the search space at this stage ($|\mathbf{act}|$). (cf. Algorithm 3, Lines 9-11).

One offspring is then generated per mutation from the best individual of the

actual population P (cf. Algorithm 3, Line 12). This is done by a random mutation of the active models' weights. We assume that the benefit of a model with a contribution to the ensemble of less than two percent is negligible. Thus the mutation step of the ES is adopted such that single weights are at least two percent or zero. This correction is done randomly, so that also when the step size of the search algorithm is rather small, the weight has a chance to surpass this barrier.

Next the offspring is evaluated for its fitness (cf. Algorithm 3, Line 13). For this the prediction of the model is to be evaluated, that is a mixture of the predictions of the base models. These base model predictions have been previously determined (cf. Algorithm 2, Line 2) so that the mixture now can be easily calculated. The offspring individual is chosen as the new parent individual if its fitness value is better than the parent's fitness value.

The search is finished when all stages are completed, which means, that all models that are currently available were part of the search space at least for the duration of one complete stage.

5.1.4 Building the Ensemble Function

With the completion of the search, the best weights combination for the ensemble is known. Models that do not contribute to the ensemble ($\beta_i = 0$) are suspended for the remainder of the actual suspension interval ($\mathbf{av}_i := 0$). All models that contribute to the ensemble are fitted to the complete data set. With the weights and the fitted base models an ensemble function, that represents the CCM, is built and returned (cf. Algorithm 2, Line 4).

5.2 Sequential Optimization Using Dynamic Ensembles

The necessary adaptations to apply the CCMs, developed in Chapter 4, to sequential optimizations were made in Section 5.1.

In this section, the dynamically adapted CCM method is thoroughly tested to obtain further experimental justification of the usefulness of these adaptations.

Furthermore, the questions posed at the introduction of this chapter are to be addressed as follows:

- To investigate the performance and reliability of the proposed dynamically adapted CCM method, it is tested in sequential optimization processes on a large set of diverse objective functions.
- The performance of the CCM is compared to the performance of the base models to evaluate if the CCM can still compete with the base models in a sequential optimization process.
- Additionally, the performance of the CCM method is compared to the performance of two strong ensemble approaches that both hold only some of the features of the CCM method.
- Finally, the CCM method is analyzed for its performance in terms of calculation time.

On each of the functions, several independent complete optimization processes are carried out using the SPO Framework as introduced in Section 2.2. The experimental setup for these experiments and the functions used is given in Section 5.2.1 and the general setup for the CCM is presented in Section 5.2.2. In Section 5.2.3, the competitors that the CCM is also compared to is introduced. Experiments using different settings for the rebuild interval τ and the suspension interval λ are carried out in Section 5.2.4. The results are discussed with a special focus on the impact of the settings of these intervals on the performance and the computation time of the CCM. Finally, the performances of two CCMs using different settings for τ and λ are compared to the performances of the base models and the two competitors presented in Section 5.2.3.

5.2.1 Experimental Setup and Objective Functions

To obtain meaningful insights about the performance of the proposed dynamically adapted CCM method it is aimed for a set of functions as diverse as possible. Therefore, a set of 10 objective functions is chosen, all showing different characteristics. Part of the set are two Gaussian landscape generator functions (GLG4D, GLG8D), four instances of two classical test problems for optimization algorithms (Ackley2D, Ackley4D, Rosenbrock4D, Rosenbrock8D), and four test functions based on physical models (piston function, robot arm function, otl-circuit function, wing weight function) (cf. Section 2.3). The Gaussian landscape generator functions are instantiated with 80 Gaussian process realizations

for the four-dimensional GLG function and 320 Gaussian process realizations for the eight-dimensional GLG function respectively.

Function	Dimension	Initial Design Size	Number Sequential Steps
GLG4D	4	60	100
GLG8D	8	100	220
Ackley2D	2	20	100
Ackley4D	4	60	100
Rosenbrock4D	4	60	100
Rosenbrock8D	8	160	100
Otl-circuit function	6	30	50
Piston function	7	110	50
Robot function	8	110	50
Wing weight function	10	280	100

Table 5.1: Settings used for the experimental setups per function.

Table 5.1 gives an overview of the different experimental setups. The initial design size used for each function is determined in a preliminary experiment and depends on the difficulty of the function. As before, the defining criterion is the share of base models that perform better than the mean predictor.
For all functions, distinct initial designs are generated in advance to ensure that all experiments have the same precondition. For the GLG functions as well as for the classical optimization problems 20 repetitions of the experiment are performed, for the functions based on physical models, ten repetitions are carried out. Each of the repetitions starts from one of the predefined initial designs.

The initial designs for the Ackley function as well as for the otl-circuit function contain a smaller number of points since previous experiments showed that the optimum is reached quite early with few points already. However, for the wing weight function, the initial design contains 280 data points since the function is rather hard.

5.2.2 Setup of the CCM Building Method

The CCM building algorithm as described in Section 5.1 has a few settings which have to be considered. Some of these settings are easy to be set to reasonable values; some of them allow for several options to be negotiable. Table 5.2 gives an overview of all of these settings and the values chosen for the experiments.

Description	Variable	Value
Ensemble building interval	τ	{1, 5, 10, 20}
Base model suspension interval	λ	{1, 5, 10, 20}
Exclusion time limit	-	300s
Number of neighbours considered for density calculation	k	20
Model-accept weight		2%

Table 5.2: Settings of the CCM building method used in the experiments

The variables τ and λ are crucial for the CCM building method. As stated before (cf. Section 5.1) these values have a significant influence on the overall computation time of the CCM method but may also influence the performance of the CCM. With too large values chosen for τ and λ, the CCM may not be able to adapt fast enough to changes in the underlying data while too small values will unnecessarily increase the computation time. The experiments presented consider a range of different settings for these values.

The algorithm allows setting a fixed time limit for fitting a single predictor during cross-validation. Models that exceed the preset time limit are excluded from the system. This ensures reasonable calculation times by enabling the algorithm to exclude models that need unreasonable long calculation times. However, this parameter should be chosen carefully since this exclusion is final; models are not reincluded after the expiration of the suspension interval. Also, it has to be taken into account that computation time is no quality indicator; sometimes longer calculation times might be preferable. In our experiments, the exclusion time limit is set to 300 seconds. This time limit is considered as safety switch only, in case that a model takes extraordinary much time for fitting and is not expected to be reached. With this setting, the use of entirely unknown models may be encouraged.

The number of nearest neighbors that are considered for the density calculation of the point's neighborhood is set to 20 points. This is assumed to be large enough

to obtain a reliable value and yet not too large so that the calculation focuses on the closer neighborhood.

The model-accept weight specifies the minimum weight that a model should have to be accepted as part of the model. In the experiments presented here this value is set to 2% since we do not intend to restrict the algorithm more than needed but consider a weight of less than 2% as negligible.

Description	Variable	Value
Designated search steps per stage	-	$10 \cdot s_{act}^2$
Maximum search steps without improvement	-	$\frac{2}{3}(10 \cdot s_{act}^2)$
Interval for $\frac{1}{5}$-success rate check	-	$5 \cdot s_{act}$
Initial step size	σ_{init}	0.4
Minimum step size	σ_{min}	0.1
Learning rate (step size variation factor)	η	0.9

Table 5.3: Settings of the (1+1)-ES used for searching the best weights

Additional parameters that are used for the (1+1)-ES are specified in Table 5.3. The learning rate η is chosen in the recommended range $[0.817, 1)$, but slightly higher than the recommended value of 0.817 to improve exploration. Also, the other settings follow the recommended settings in Bäck et al. [110]. As mentioned before (cf. Section 5.1) the search algorithm is adapted stage wise. Some of the parameters specified here are reset or adjusted with the beginning of every stage. The designated number of search steps per stage is one of these variables. It also depends on the number of base models that are part of the search space in the considered stage of search (s_{act}). However, the search on one stage may be finished earlier if no progress is made. In the experiments presented here a stage is been terminated when no progress is made in 2/3 of the total number of search steps allowed for this stage. If progress is made the model is supposed to be valuable to the ensemble and is granted the designated amount of search steps.

In a fixed interval of search steps, the success rate of the search is calculated, and the search step width is adapted accordingly. The length of this interval also depends on the number of models that are part of the search space at that time (s_{act}). The step width of the ES is reset to its initial value with the start of each stage.

5.2.3 Competitors

To investigate the performance of the proposed dynamically adapted CCM method its performance is compared to the performance of the base models as well as to two ensemble-like approaches. These approaches will hereafter be referred to as 'Initial' and 'Choose'.

'Initial' builds the CCM only in the first iteration of SPO ($\tau = \infty$). In case of failure of one base models during the prediction, this model is excluded from the ensemble. For this prediction, the weights of the remaining models are adjusted to keep their relation and fulfill the requirement to sum up to one, despite the missing model. Additionally, the malfunctioning model is directly and finally excluded from the set of model choices. The CCM building process is then newly started in the next step of the SPO. By using this method, we want to get some insights into the benefits of adjusting the ensemble during the optimization process.

'Choose' selects a single base model in every λ-th iteration of the SPO. This method also uses the weighted 10-fold cross-validation like the CCM building method, since in preliminary experiments it showed better performance using weighted cross-validation than by using the standard cross-validation. However, in contrast to the CCM building method Choose is restricted to choosing a single best model only. Thereby we assess the benefits of using mixtures of models instead of selecting and updating only single base models.

5.2.4 The Impact of Rebuild- and Suspension Intervals

Before turning to the main experiments, some thought should be given on the settings of the values τ and λ. As stated before, these values have a crucial influence on the computation time of the CCM building method as well as on its prediction performance during sequential optimization. To get some insights into the impact of these values on the quality of the optimization result, experiments are run with all reasonable combinations for $\tau \in \{1, 5, 10, 20\}$ and $\lambda \in \{1, 5, 10, 20\}$. Settings, where the model suspension interval λ is shorter than the model rebuild interval τ, are not considered as reasonable since the reactivation of temporarily suspended models only comes into account during the next model rebuilding process. Experiments are run on the complete set of objective functions as presented in Section 5.2.1.

Table 5.4 gives an overview of the results of these experiments. The table is ar-

$\tau = 1$	$\lambda = 1$	$\lambda = 5$	$\lambda = 10$	$\lambda = 20$
ackley2D	4.736 (±1.8360)	3.082 (±2.1987)	2.421 (±2.2847)	**1.954** (±2.0943)
ackley4D	8.453 (±2.6557)	7.728 (±3.4242)	6.932 (±3.1568)	6.163 (±3.5904)
GLG4D	21.96 (±11.6650)	21.92 (±11.9038)	24.21 (±11.3581)	23.77 (±11.4260)
GLG8D	30.45 (±8.9146)	28.11 (±11.1735)	**26.05** (±9.9418)	26.67 (±11.3850)
otl-circuit	2.621 (±0.050386)	**2.605** (±0.002278)	**2.605** (±0.001937)	**2.605** (±0.001924)
piston	**0.1647** (±0.000499)	0.1655 (±0.002251)	0.1676 (±0.004043)	0.1679 (±0.004456)
robot	**0.01078** (±0.014679)	0.01592 (±0.020619)	0.01768 (±0.027308)	0.0227 (±0.026355)
rosenbrock4D	2.673 (±1.599095)	**2.236** (±1.728283)	2.412 (±1.649369)	2.988 (±1.545986)
rosenbrock8D	4075 (±2482.58)	3684 (±2353.88)	**3119** (±1606.20)	4302 (±3218.83)
wingweight	177.7 (±12.7954)	179.6 (±6.154)	176.2 (±13.9320)	178.2 (±16.4675)
Md RankSums	56	46	**27.5**	45
Mn RankSums	60	43	**33**	46
Md Rank	5	3	**1**	2
Mn Rank	6	2	**1**	3
$\tau = 5$		$\lambda = 5$	$\lambda = 10$	$\lambda = 20$
ackley2D		3.68 (±2.1033)	2.936 (±2.1041)	2.427 (±1.7135)
ackley4D		7.685 (±2.4899)	7.122 (±2.6919)	6.323 (±3.5570)
GLG4D		20.58 (±12.8548)	24.01 (±11.0480)	20.66 (±14.1705)
GLG8D		30.21 (±8.2603)	30.36 (±10.3108)	29.78 (±9.2280)
otl-circuit		2.645 (±0.066244)	2.606 (±0.003172)	2.611 (±0.017736)
piston		0.1692 (±0.005782)	0.1684 (±0.004193)	0.1709 (±0.007741)
robot		0.01772 (±0.027462)	0.02818 (±0.0283594)	0.02828 (±0.024504)
rosenbrock4D		2.452 (±1.600449)	18.95 (±72.166486)	7.898 (±21.774267)
rosenbrock8D		3812 (±1473.23)	3939 (±1807.51)	3895 (±1645.08)
wingweight		**173** (±9.1062)	176.3 (±13.1074)	178.9 (±9.8847)
Md RankSums		60	72	62
Mn RankSums		54	65	63
Md Rank		6	10	7
Mn Rank		5	9	7
$\tau = 10$			$\lambda = 10$	$\lambda = 20$
ackley2D			3.268 (±2.0659)	2.641 (±2.0901)
ackley4D			7.452 (±3.0298)	6.386 (±3.2157)
GLG4D			26.26 (±7.4316)	25.68 (±9.8755)
GLG8D			30.7 (±12.4646)	28.56 (±12.0462)
otl-circuit			2.632 (±0.056935)	2.624 (±0.052627)
piston			0.1672 (±0.003480)	0.1694 (±0.008199)
robot			0.0191 (±0.021059)	0.02285 (±0.020645)
rosenbrock4D			2.66 (±1.941899)	3.005 (±2.081024)
rosenbrock8D			4429 (±2114.36)	4155 (±1879.44)
wingweight			178.8 (±11.0519)	175.9 (±18.6308)
Md RankSums			68	65
Mn RankSums			74	64.5
Md Rank			9	8
Mn Rank			10	8
$\tau = 20$				$\lambda = 20$
ackley2D				2.323 (±2.2779)
ackley4D				**6.09** (±2.9370)
GLG4D				**18.27** (±12.5356)
GLG8D				27.31 (±11.0315)
otl-circuit				2.61 (±0.012475)
piston				0.1693 (±0.006238)
robot				0.0211 (±0.020513)
rosenbrock4D				25.45 (±96.6529205)
rosenbrock8D				4348 (±3245.30)
wingweight				175.9 (±14.9929)
Md RankSums				48.5
Mn RankSums				47.5
Md Rank				4
Mn Rank				4

Table 5.4: Experiment results for the comparison of different settings for τ and λ. Given is the mean and standard deviation of the optimization results of the ensemble for each reasonable combination of τ and λ. The two lowest rows give the ranking results of these runs in comparison. Best results are marked bold.

ranged in four major rows; each row gives the results for one value of the model rebuild interval τ while each column represents one setting for the model suspension interval λ. Each entry names the mean optimization result with standard deviation that has been achieved during the optimization processes using the corresponding CCM. The best values that were achieved on each function are marked bold. To allow for an evaluation of the methods over the set of functions with so strongly differing features function wise rankings are used. Although the result table only names the mean optimization results the median optimization results also have been ranked. The sums of these ranks are shown in the rows 'Md RankSums' and 'Mn Ranksums' respectively. The last two rows of each major row 'Md Rank' and 'Mn Rank' only shows the rankings of 'Md RankSums' and 'Mn Ranksums' respectively.

It can be seen, that the CCM using $\tau = 1$ and $\lambda = 10$ is ranking first place for mean optimization value as well as for median. Looking at the RankSums one can say that the result is not even tight. Furthermore, the three best-performing settings can be found in the first major row ($\tau = 1$). Also noteworthy is the fact, that the model which is ranked fourth, in both mean and median optimization result, is the CCM using $\tau = 20$ and $\lambda = 20$.

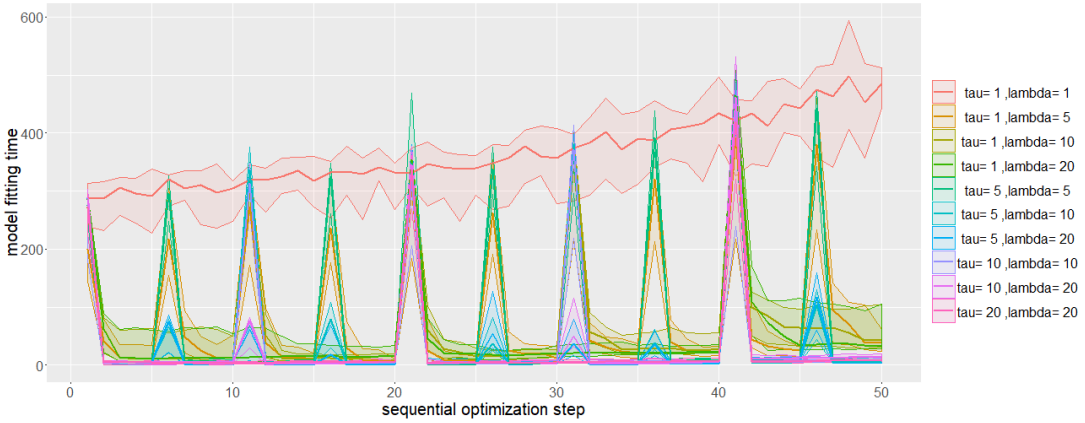


Figure 5.2: Average model fitting times during sequential optimization for the different settings of τ and λ per sequential step. The solid line marks the mean time while the opaque area shows the standard deviation of the calculation time. Most time-consuming is the approach using $\tau = \lambda = 1$ since it is rebuilding the ensemble on the complete model set in every step. The least time consuming is the one using $\tau = \lambda = 20$, which shows a peak in calculation time only in every 20th step.

However, as mentioned before, these settings do not only have an impact on the

performance of the models but also on its calculation times. Figure 5.2 gives an insight into the behavior of the different setups.
Three different situations may be encountered during the model fitting step. These are model rebuild steps where all models are part of the search space, model rebuild steps where some models are suspended and steps where the model is not rebuilt.
As expected the steps where the ensemble has to be rebuilt using the complete set of base models are the most expensive steps, in terms of calculation time. Whereas the steps where no new ensemble is built are negligible. This behavior can be easily seen in Figure 5.2, comparing the lines for $\tau = 1$ and $\lambda = 1$ versus $\tau = 20$ and $\lambda = 20$. The calculation times per sequential step for the CCM with $\tau = 1$ and $\lambda = 1$ ranges, slowly increasing, between 200 and 600 seconds. Whereas the CCM with $\tau = 20$ and $\lambda = 20$ has calculation times close to zero in most of the sequential steps, only every 20-th sequential step the ensemble is rebuilt and the calculation time shows a peak with calculation times corresponding to the calculation times of the CCM using $\tau = 1$ and $\lambda = 1$.

The calculation time needed in such steps, where some models are suspended from the system, is strongly depending on the number of models that are not suspended. The line depicting the times for the setting using $\tau = 1$ and $\lambda = 20$ shows this. Although the model is built in every step, the calculation time is negotiable and steadily decreases step by step, as further models are suspended. In step 20 and step 40 respectively all models are reincluded to the system, which has a strong impact on the calculation time of this CCM after step 40.

Table 5.5 shows exemplary running times for the same choice of configurations of λ and τ summed up for an entire optimization process on two different functions. The configurations where λ is set to the same value as τ are not affected by the suspension since it is released in the same step as the model is rebuilt, and therefore in every model building step the full model set is available.

The results suggest that two choices are best, depending on the priorities set. For one the CCM with $\tau = 1$ and $\lambda = 10$ seems to be the best choice when calculation time has not to be taken into consideration too much. For another, the CCM with $\tau = 20$ and $\lambda = 20$ seems to be a worthwhile choice.
With a setting of $\tau = 1$ and $\lambda = 10$, the CCM is rebuilt in every step and thus can quickly adapt to changes in the underlying data. The suspension interval is large enough to reduce the calculation time remarkably and small enough to allow for an adaptation of the active models throughout the optimization process.
Using a CCM with $\tau = 20$ and $\lambda = 20$ results in an ensemble, that shows a good performance while having the best results in terms of calculation time. Though we

		λ			
		1	5	10	20
τ	1	17659 (± 4956)	4717 (± 1435)	3763 (± 720)	3068 (± 943)
	5		3765 (± 1011)	2464 (± 452)	1782 (± 422)
	10			2096 (± 801)	1772 (± 509)
	20				1421 (± 432)

Table 5.5: Average computation times (s) for a complete sequential optimization process on the GLG4D function depending on the values for τ and λ. On the diagonal, the results for such settings are shown where λ has no influence since it matches to the value of τ. In these cases, the full set of available models is used in the ensemble building process. The same applies for combinations of τ and λ where $\lambda < \tau$, therefore these fields are left blank.

aim for problem setups with high-cost objective functions where the calculation time of the model is neglectable, this still might be an interesting choice in some cases.

Recapitulating it can be said that the choice of the values for τ and λ should be well considered. The results show that these values have a heavy impact on the performance of the model in terms of prediction quality as well as computation time. A smaller value for the model rebuild interval seems to be preferable though the results show that also larger values can lead to good results (cf. Table 5.4, $\tau = 20$, $\lambda = 20$). However, a smaller value for the suspension interval must not necessarily lead to better results (cf. Table 5.4, $\tau = 1$, $\lambda = 1$). We assume that a large set of base models still makes it harder to build the best fitting model. So it might indeed be beneficial to suspend models that are not contributing to the system for some time. Of course, these values also should be chosen considering the number of sequential steps that are to be performed.

5.2.5 The Performance of Dynamical Adapted CCM in SPO

In the following, the performance of the CCM in sequential optimization processes is closer investigated. For these experiments the CCM with $\tau = 1$ and $\lambda = 10$ is chosen as well as the CCM using $\tau = 20$ and $\lambda = 20$. The ensembles are compared to the base models as well as to the competitors 'Choose' and 'Initial' as introduced in Section 5.2.3. Again, experiments are run on the complete set

of objective functions as presented in Section 5.2.1.

Table 5.6 gives an overview of the Results of the experiments. The structure of the table resembles the structure of Table 5.4 in the previous section. Main difference is the columns 'Best Base Model'. Since for the main experiment setup the chosen CCM methods are compared against 'Choose' and 'Initial' as well as all base models that are part of the set, the complete result is condensed to the relevant information to preserve the readability.

FUN	$\tau = 1, \lambda = 10$	$\tau = 20, \lambda = 20$	Choose	Initial	Best Base Model	
ackley2D	2.496 (±1.931)	1.934 (±2.041)	5.250 (±2.604)	0.869 (±1.392)	**0.343** (±0.694)	MLP
ackley4D	6.850 (±3.179)	6.022 (±3.242)	9.189 (±3.182)	**4.403** (±3.101)	5.206 (±1.074)	Lm
GLG4D	22.43 (±12.00)	23.58 (±13.02)	**14.31** (±14.47)	25.07 (±7.83)	21.10 (±13.17)	corrgauss
GLG8D	29.62 (±9.42)	30.34 (±7.17)	40.02 (±13.39)	35.31 (±10.42)	**28.70** (±11.39)	corrgauss
otl-circuit	2.605 (±0.002)	2.610 (±0.0123)	2.695 (±0.080)	2.619 (±0.046)	**2.604** (±0.0002)	Earth, MLP
piston	**0.167** (±0.003)	0.170 (±0.008)	0.172 (±0.004)	0.168 (±0.001)	0.167 (±0.001)	MLP
robot	0.013 (±0.020)	0.018 (±0.018)	0.040 (±0.031)	0.021 (±0.022)	**0** (±0)	MLP, Neuralnet
rosenbrock4D	**2.414** (±1.522)	4.198 (±2.088)	2.665 (±1.331)	160.4 (±208.2)	3.554 (±2.490)	corrgauss
rosenbrock8D	4127 (±2989)	3224 (±2042)	5755 (±4100)	3171 (±1758)	**697** (±582)	Lm
wingweight	182.5 (±13.11)	**173.0** (±10.35)	180.4 (±15.10)	174.0 (±15.65)	174.8 (±14.12)	correxp
Md RankSums	29	27.5	41	32	**20.5**	
Mn RankSums	28	31	42	31	**18**	
Md Rank	3	2	5	4	**1**	
Mn Rank	2	3	5	4	**1**	

Table 5.6: Results for the main experiment setup. Depicted are mean optimization result with standard deviation. The presentation of the results for the base models is consolidated to only depict the performance of the best model for each function.

Thus, the column 'Best Base Model' names the mean optimization result with standard deviation and the name of the corresponding base model which performed best on this function. In case that two base models showed the same performance in terms of mean optimization result, both base models are named here. For differing performances in terms of standard deviation only, the smaller standard deviation value is shown. Additionally, the complete results for the comparison of the two CCMs to all base models is given in the Appendix A.2.

Inspecting the overall rankings of the models depicted, it can not be denied that the base model is ranked best. However, it has also to be taken into account, that the best base model is changing, depending on the objective function. Given that the most appropriate base model is not known to the user, the next best choice is the CCM. We assume the better choice to be the CCM with $\tau = 1$ and $\lambda = 10$, but the differences in performance seem to be negligible, at least for these experiments.

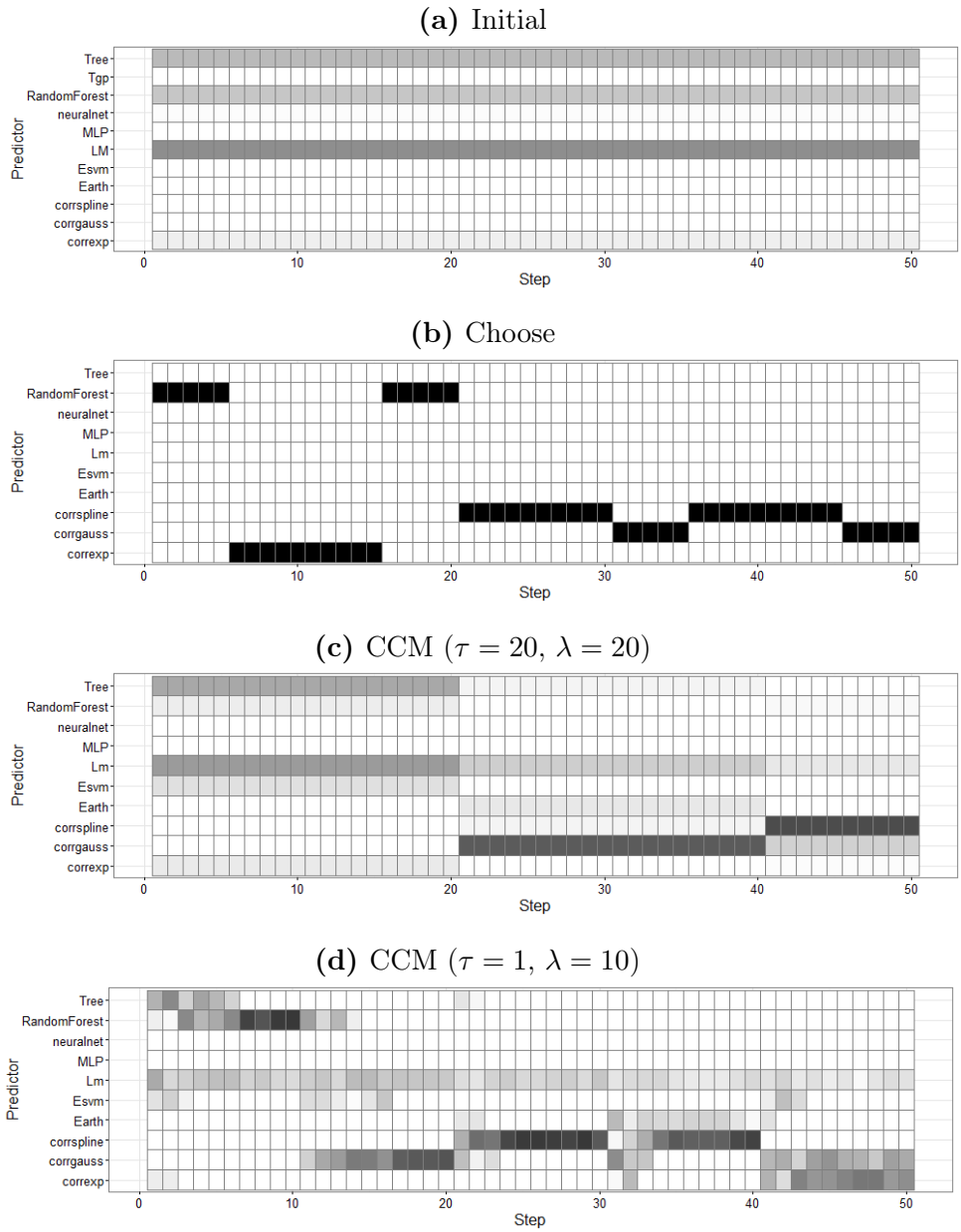


Figure 5.3: Distribution of weights during one exemplary sequential optimization process on the GLG4D function. 'Initial' is forced to use the same ensemble throughout the whole optimization process, while 'Choose' switches to the best choice in every fifth step. The ensemble ($\tau = 1, \lambda = 10$) starts with a setup that resembles the setup of 'Initial', but adjusts its weights in the next steps, often giving large parts of the weight also, but not exclusively, to the model also preferred by 'Choose'.

Figure 5.3 gives further insights into the behavior of the different types of models. As introduced in Section 5.2.3, 'Initial' builds its ensemble only in the very first step and then sticks to it. Whereas 'Choose' selects the best performing model every fifth step. Here it can be seen, that the preferred base model chosen by 'Choose' switches several times. The two CCM models start with a similar ensemble as 'Initial', but while the CCM with $\tau = 20$ and $\lambda = 20$ keeps its setup for 20 steps without changes the CCM with $\tau = 1$ and $\lambda = 10$ slightly modifies its setup in every step. This improved ability to adapt to changes may also explain the fact, that the distribution of weights resembles more to the choice of 'Choose' than the weights distribution of the CCM using $\tau = 20$ and $\lambda = 20$.

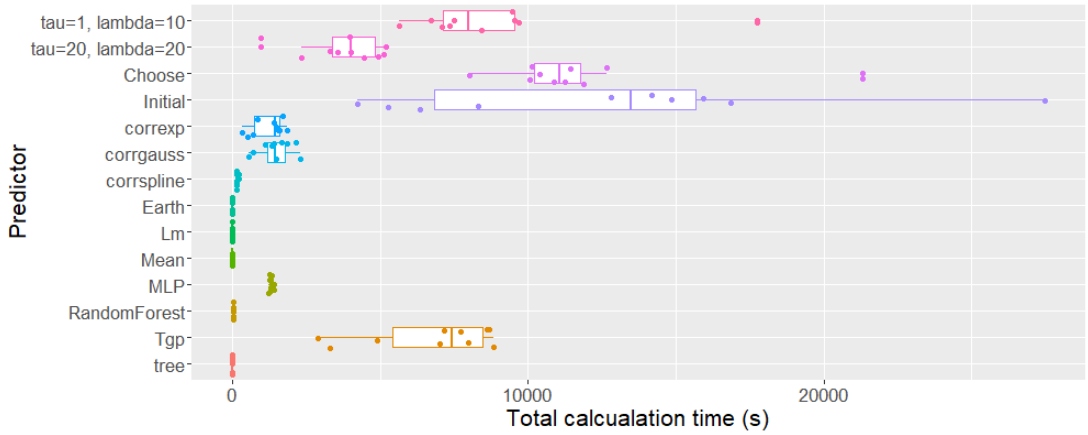


Figure 5.4: Shown is a boxplot of the calculation times that are needed by the different models during a complete sequential optimization process on the wing weight function. Noteworthy is, that on this function 'Choose' needs significantly more time than both ensembles. Also, the calculation times of 'Initial' are surprisingly long with an also rather large variance. Therefore, also 'Initial' takes more calculation time than both ensembles, in most cases.

Figures 5.4 and 5.5 give another insight into the calculation times of the different models. The times depicted here each represent the calculation times of a complete optimization process carried out on the wing weight function, the most expensive function in terms of model calculation time, and on the GLG4D function respectively. Although the calculation time on this wing weight function builds an exception, the relation of the calculation times between the different models is similar on all considered functions.

Remarkable in these results is, that the computational cost needed for the two competitors 'Choose' and 'Initial' is not necessarily less than for the two CCM

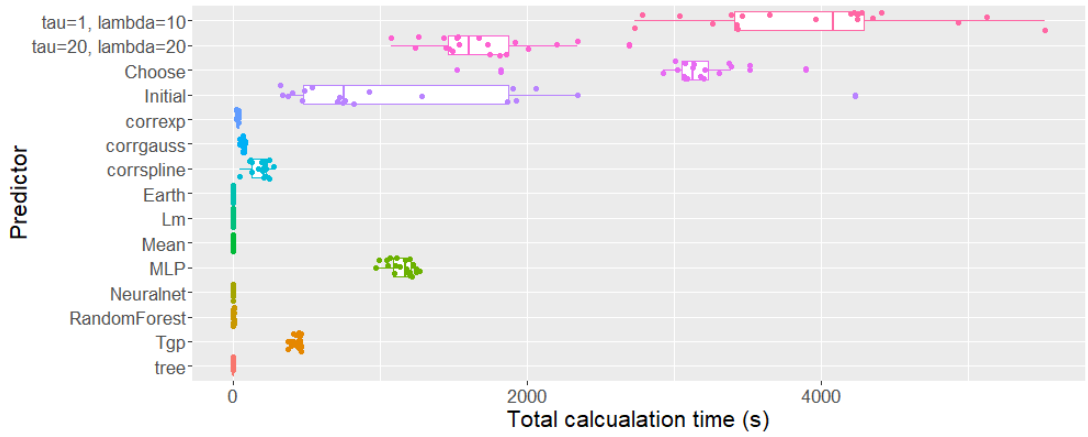


Figure 5.5: Shown is a boxplot of the calculation times that are needed by the different models during a complete sequential optimization process on the GLG4D function. Noteworthy is, that 'Choose' takes significantly more time than the CCM using $\tau = 20$ and $\lambda = 20$. 'Initial' has a large variance in its calculation times but in most cases performs faster than both ensembles.

instances. On the wing weight function, the computation times for 'Choose' are significantly longer than for both ensembles, while on the GLG4D function it still performs slower than the CCM using $\tau = 20$ and $\lambda = 20$. This may be based on the fact, that 'Choose' performs a cross-validation on the full set of base models in every fifth step whereas the ensembles use the full set of base models only every tenth and twentieth step respectively.
The calculation times for 'Initial' are surprisingly long, taking into account, that the cross-validation of the full set of base models is done only in the very first step of the SPO process. A possible explanation for this might be an unfortunate choice of base models in the first step. I.e., in most of the twenty repetitions carried out on the GLG4D function 'Initial' chose at least one, sometimes even more of the slower performing base models. However, this is only an assumption and requires further investigation.

5.3 Conclusion

The primary goal of this chapter was to adapt the CCM method developed in Chapter 4 such that it can be applied to SPO and thereby release the user from the burden to select the right surrogate model, especially in sequential optimization

tasks. As before, it was aimed for a method that in any situation can compete with the best performing base model. To achieve this, in Section 5.1 several adjustments were made to the basic CCM method to improve its accuracy, reliability and computation time in sequential optimization processes. In Section 5.2 the dynamically adapted CCM method was thoroughly tested for these virtues in different setups and compared to the base models as well as to two strong ensemble competitors.

The results showed that the dynamically adapted CCM performs reliably and, in terms of accuracy, can compete with the base models as well as the competitors. It was shown that in most cases a base model showed the best optimization result. However, in these cases, a CCM was placed second in general. Moreover, though the 'BestBaseModel' was ranked first in the overall evaluation, it has to be taken into account that the concrete best base model was differing for each function. Therefore, in comparison to the base models the CCM would be the best choice, given that the best base model is not known beforehand.
Concerning the ensemble competitors, it could be shown that the CCM method has a clear advantage over approaches like 'Choose' that only select a single best base model in fixed intervals. The difference to 'Initial', a CCM instance using $\tau = \infty$, is a bit smaller; still the CCM instances that updated their ensemble setup throughout the optimization process showed better results.
The comparison of CCM instances using different values for τ and λ gave further evidence for the advantageousness of regular updates of the ensemble setup. Though it was visible that an update on the full set of available models is neither needed nor useful, the results showed that a regular update on a reduced set of base models is beneficial.

Concerning the calculation time, the CCM method and the parameter τ and λ showed the envisioned behavior. As expected, in terms of calculation time, the CCM cannot compete to a single base model. However, the difference between the ensemble competitors and the CCM instances was not that clear. As expected 'Choose' performed slower than the CCM using $\tau = 20$ and $\lambda = 20$ but depending on the function it even performed slower than the CCM using $\tau = 1$ and $\lambda = 10$. 'Initial' showed a surprisingly large variance in its computation times and therefore was, other than expected, not always performing faster than the CCM instances.

However, as emphasized in Section 3.6, it was aimed for a strategy that works reliably and as accurately as possible on arbitrary objective functions knowingly accepting that this is probably going to happen at the expense of the ensembles computation time. Still, the computation time of the CCM depends on the choice

of the values of τ and λ and can, within bounds, be adapted to the needs of the user and with a focus on expensive real-world applications is expected to be neglectable.

To a great extent, this chapter is based on the articles *'Weighted Ensembles in Model-based Global Optimization'* by Friese et al. [74] and *"Optimally Weighted Ensembles of Surrogate Models for Sequential Parameter Optimization"* by Friese et al. [75]. Major parts from the original articles were adopted verbatim. Of course, the text was adapted to fit the structure and notation of this thesis.

Chapter 6

Summary and Outlook

In this work, we specified a taxonomy of ensembles and discussed known methods for its strengths and weaknesses (e.g., restrictions to particular types of models or applications). The main contribution of this work aims to overcome these weaknesses and develop an ensemble strategy that works reliably and accurately on arbitrary objective functions and models and thus releases the user from the burden to select the right surrogate model. The proposed ensemble strategy is first developed and analyzed in a small setup of two base models and later extended for the use of a large heterogeneous set of base models. Finally, the method is adapted for and applied to SPO on a large set of objective functions of various characteristics.

Section 6.1 provides a summary of this work and discusses the conclusions drawn from the performed research. Section 6.2 gives an overview of possible future work on this topic.

6.1 Summary

A common method to perform an optimization on a function that can not be optimized analytically is to perform a search on this function by iteratively and strategically choosing and evaluating points of the function. However, in real-world optimization tasks, the budget in terms of number of function evaluations is often constrained by the time or cost for these function evaluations. In such circumstances, it is a common technique to learn a surrogate model, e.g., regression

model, of the response function from available evaluations and to use this model to decide on the location of future evaluations. Sequential Parameter Optimization (SPO) is a well-known approach for solving black-box optimization problems with expensive function evaluations with the help of surrogate models.

For such optimization processes, the choice of the surrogate model can have a significant influence on the solution quality and performance of the optimizer. However, to make meaningful decisions on which surrogate model to select for a given problem, often expert knowledge about the objective function and the characteristics of the surrogate model likewise is required. However, in many situations, preliminary knowledge about the function, or all available models, is not available. Automated methods that learn all by itself which surrogate model type suits the problem best, could help to overcome this problem.

This thesis introduces new methods on how to manage multiple heterogeneous surrogate models for regression and optimization of expensive black-box optimization problems. The overarching goal is to release the user from the burden to select the right surrogate model and to create an ensemble building strategy that works reliably and accurately on arbitrary objective functions and models. The primary focus is on regression problems, and optimization processes that allow only for a comparatively small number of function evaluations since these are constraints that often come with real-world problems.

In Chapter 3 of this thesis a taxonomy of known methods to do this is introduced and specified. The model selection methods are classified into two types.
The 'Single Evaluation Model Selection' methods, in general, use a predefined strategy to select the most appropriate model. This strategy may also utilize data obtained from previous evaluations. However, such ad hoc rules ignore the rules of parsimony and do not, or only marginally, rely on the data to help select the best model.
Methods classified as 'Multi Evaluation Model Selection' tackle this problem by evaluating all available types of surrogate models. But under circumstances when there is more than one strong model in the set, it might be beneficial to combine inference output across several models.
This is what is done by the 'Model Combination' methods. However, the known strategies, in general, are restricted in one or the other way (e.g., to homogeneous models or particular applications).

Based on these studies of existing ensemble approaches and their strengths and weaknesses, a method is envisioned, that overcomes the deficiencies and combines the advantages of these approaches. To this end, the method should do an ex-

haustive preliminary evaluation of all models to gain the best insight into the models' performances, then trains all models on the data to enable the use of the complete knowledge of all models. Still, the method should follow the principle of parsimony and prefer a combination of predictions over a single prediction only if it is clearly beneficial for the overall accuracy. The same applies to the number of models used, it should not be a decision between a single model or a combination of all, but any number of models that seem to be best.

In Chapter 4 the insights from Chapter 3 are used to develop a new ensemble method. This approach is studied in a fundamental way, by first evaluating ensembles of only two surrogate models in detail. It is shown that the convex combination of models is beneficial in many cases since the convex combination of the predictions of two base models averages positive as well as negative prediction errors of the base models. The ensemble generated by the convex combination of models can compete with the base models and in some cases even outperforms them.

The insights gained in the studies of convex combinations of two base models make convex linear combinations of models an ideal choice for combining models. The advantages recognized in this study are:

- Due to the linear convex combination that is used for combination, the ensemble cannot perform worse than the weakest base model.
- The ensemble can perform better than the base models when compensating opposing prediction errors.
- A CCM is favored over a base model only if the overall fit of the ensemble model is actually better (in terms of RMSE) than the overall fit of both base models.
- The nature of the combination that is given by a weighted sum with a normalized positive weight space is intuitive and interpretable.
- The linear convex combination of predictions for a given set of weights is easy to compute.

In preparation for the implementation of the algorithm with a large set of heterogeneous models, the algorithm is then specified in a more general way using three base models. Furthermore, the exhaustive search used to find the optimal combination weights in a fixed grid is replaced by a more flexible (1+1)-ES. These adaptations are accompanied by experiments to ensure that the changes have no adverse influence on the performance of the method.

On this basis, in the following, the set of base models is extended to the intended size and experiments are carried out to ensure that the performance of the method does not suffer from this change. First experiments show that the (1+1)-ES, without further changes, is not able to handle the extended search space. Therefore, several approaches are discussed and implemented to overcome this difficulty. Finally, the chapter closes with comparing experiments of all approaches on a set of test functions based on physical models. It is shown that the approaches are at an advantage over the base models in some cases. Another essential insight gained in these experiments is that sometimes a model can make a beneficial contribution to the ensemble although no single, high-ranked model is able to do so.

In Chapter 5 the developed ensemble strategy is introduced in more detail and adapted for the application in SPO. These adaptations are needed, e.g., to adjust the models continuously in the presence of dynamically changing and non-uniform data sets. These adaptations are:

- Periodically rebuilding of models and temporarily suspending models, to take dynamical updates into account.
- Local density weighted cross-validation, to deal with non-uniform point distributions.
- Adaptation of the (1+1)-ES weight optimization method, to deal with large ensemble sets.

The dynamically adapting ensemble strategy is then extensively tested for its performance and computation time in sequential optimization processes on various objective functions. First, instances of the method using different settings for the rebuild and the suspension interval are compared and the impact of these settings on the performance and computation time is analyzed. In both, its performance as well as its computation time, the method is also compared to the base models and to two strong ensemble competitors that share different parts of the characteristics of the proposed ensemble method.

The results show that the dynamically adapting ensemble strategy performs reliably and, in terms of accuracy, can compete with the base models as well as the competitors. It is shown that the proposed ensemble strategy method has a clear advantage over approaches that are restricted to the selection of a single best base model in fixed intervals. Given that the best base model is not known beforehand, the dynamically adapting ensemble strategy would be the best choice. The analysis of the impact of the rebuild interval and the suspension interval on

the performance of the ensemble strategy showed that a regular adaptation of the ensemble setup is preferable, though this update should be restricted to a smaller subset of stronger base models that may be updated in a less frequent interval. A well-considered choice for the length of these intervals also has a distinct impact on the calculation time of the ensemble strategy.

In summary, and with the main research question in mind, it can be said that the primary goal to develop a strategy that works as reliably and as accurately as possible on arbitrary objective functions, and that uses arbitrary types of surrogate models is accomplished. The ensemble method proposed in this thesis selects or combines surrogate models from a set of heterogeneous surrogate models to achieve prediction results that can compete with or even improve the predictions of single models. Though it is knowingly accepted that this could happen at the expense of the ensembles computation time, this drawback can be lessened using well-considered settings for the rebuild interval and the suspension time. However, with a focus on expensive real-world applications, the computation times needed is expected to be negligible.

6.2 Outlook

First and foremost the approach would lend itself to be implemented in other sequential optimization packages, such as SUMO [55]. Furthermore, it could be beneficial to use the proposed ensemble method in algorithm configuration frameworks, such as SMAC [15, 112] and IRACE [56].

Some questions remained unanswered and require further investigation; other questions were not addressed yet. Questions that were not addressed in this work concern the ways of combining and evaluating models. We chose to use convex linear combinations of the models' predictions since it is both easy to calculate also for several heterogeneous models and comprehensive in terms of meaningfulness. For the evaluation of the models, we used the RMSE, which we adapted to a weighted variant for the application in SPO. Both, the evaluation method, as well as the way to combine the models, may be further investigated to determine the best way to do this. The question of how to evaluate and find the best model is accompanied by the general questions if better surrogates always result in improved performance.
The first of these questions concerns the characteristics of the search space for the model weights. Some of the results in Chapter 4 suggest the assumption that the function that is searched might be convex. If this assumption could be confirmed

this would allow for more direct search strategies to be applied and thus enhance the proposed ensemble strategy.
In this work, we chose an (1+1)-ES. Building on the assumption that the regarded search space is supposedly but not surely unimodal and convex this would be a reliable and robust search algorithm that performs well on a search landscape as assumed, but can also handle more difficult search spaces. However, other search strategies may be tested to improve the overall performance.
The recommendation for the values of τ and λ are based on a comparatively small number of experiments. Still, these values have a high impact on the overall performance and computation time of the proposed ensemble method. Further investigation may be done to allow for a more precise and reliable recommendation.
Clustering of the observed data points is an interesting idea and might be a beneficial addition to the proposed method. Though for this work, we did not consider this since the main focus was on real-world applications which are often restricted to a smaller number of function evaluations that complicate reasonable clustering of the data points.

The calculation time of the method may be negotiable for expensive real-world applications but remains a drawback for less expensive applications. However, the method has the right prerequisites to be further sped up through parallelization. First and foremost, the calculation time needed for the cross-validation step, which makes the majority of the calculation time, could be immensely reduced by parallelization. Additional calculation time may also be saved by parallelization of the optimization step by applying parallel optimization methods as introduced by Mostaghim et al. [113].

Surrogate models are not only used in Sequential Optimization but also in the selection of Evolutionary Algorithms. Also, for such surrogate-model assisted evolutionary algorithms the approach could be beneficially implemented. For an overview, see Chugh et al. [114].

Finally, the method could be applied to multi-objective optimization problems. First approaches in this direction were presented by Hussein et al. [115, 116]. Since the proposed approach showed good results in single-objective optimization problems, it is expected to show comparable good results in multi-objective optimization problems. Today, optimization with (much) more than 3 objective functions is often referred to as many objective optimization. Also for these problems first surrogate model based approaches have been proposed by Chugh et al. [117] and it could be investigated how these could benefit from mixed models.

Some of the questions and ideas formulated in Section 6.2 are based on questions and ideas of the bookchapter *'Open Issues in Surrogate-Assisted Optimization'* by Stork et al. [118].

Appendices

Appendix A

Supplementary Results

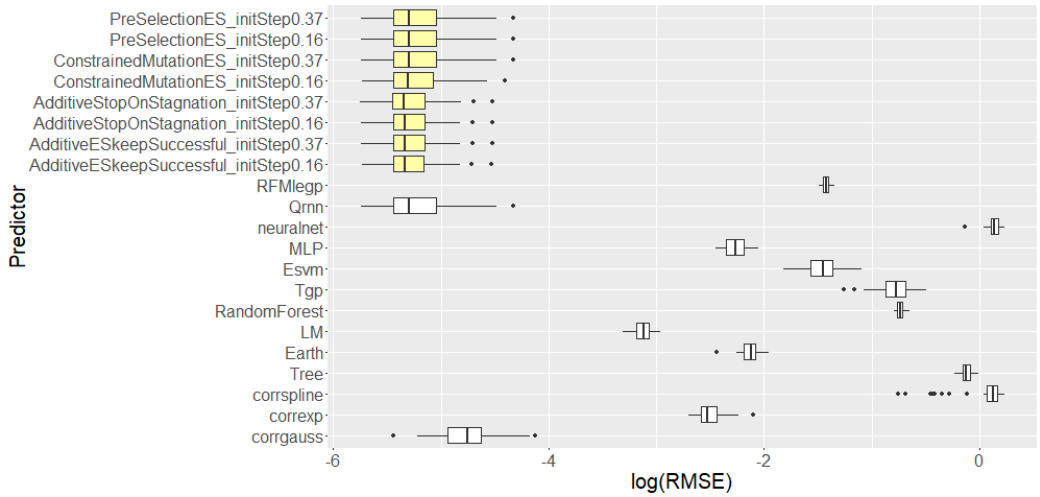
A large number of experiments was carried out for this thesis, and an even larger number of result plots was generated for the evaluation of these experiments. For the sake of readability, only the most relevant plots were shown in the main part of this thesis. In the following, supplementary plots are provided that also may be of interest since they provide further insights into the functionality and performance of the proposed ensemble method.

A.1 N-ary Ensembles on Higher Dimensional Physical Functions

In Section 4.5 experiments were carried out to compare the ensemble building method using different adaptations of the ES to each other and all base models. The results presented were condensed to the relevant information, each time showing only the results of the best performing base model, to preserve the readability. For the sake of completeness and to prove that no important information was omitted, in the following, the results for all base models are presented.

Figure A.1 shows the results for the experiments on the otl-circuit function (cf. Figure A.1a) and on the piston function (cf. Figure A.1b). On these functions, the ensemble adaptations and the best base model show comparable results and perform significantly better than the remaining base models, whose performances are heavily varying.

(a) Result on the otl-circuit function.



(b) Result on the piston function.

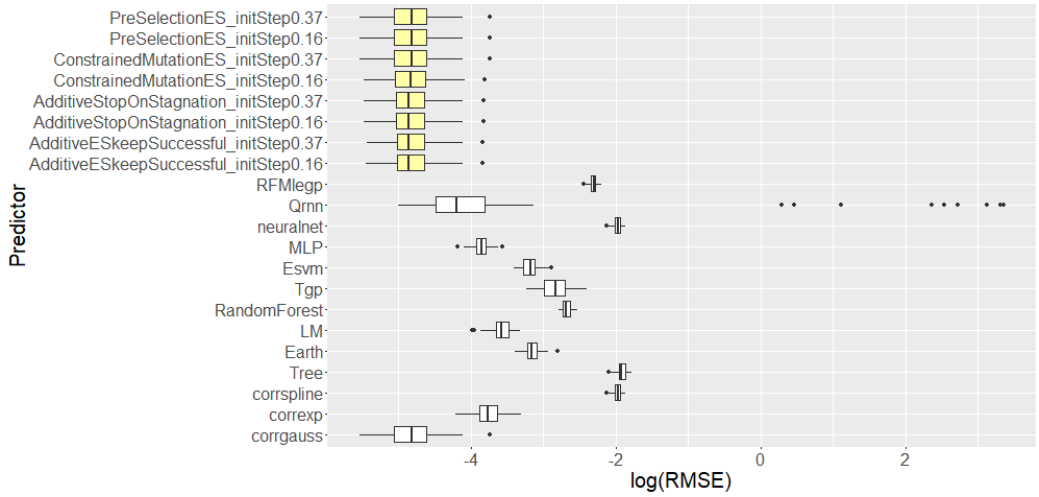
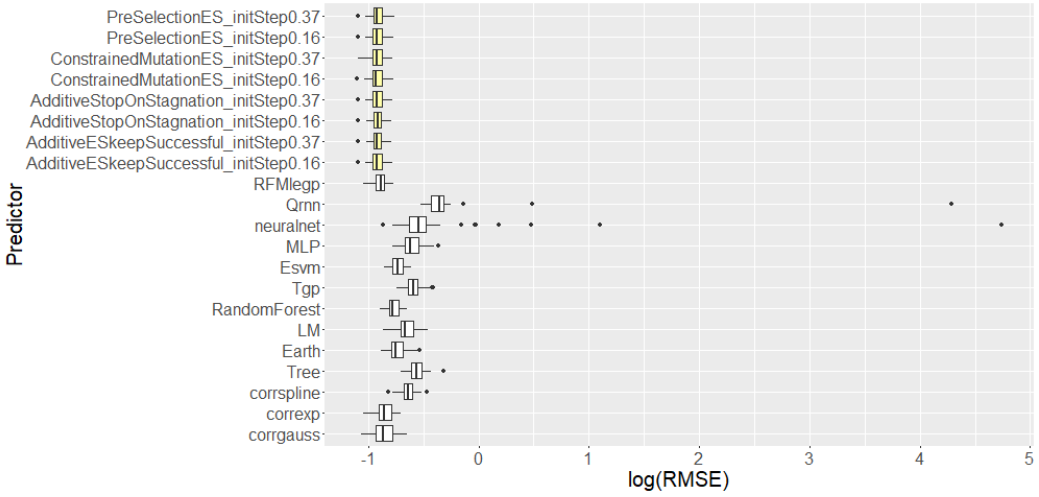


Figure A.1: The plot shows the results of the comparison of the performances, in terms of RMSE, of the different adaptations of the ensemble building method and all base models on the otl-circuit function and the piston function. Ensemble results are colored yellow, the base model result is shown in white. The ensemble adaptations can compete with the best base model and perform clearly better than all remaining base models.

(a) Result on the robot function. The ensemble adaptations can compete with the best base model. Correxp and Corrgauss also show comparable performances, only slightly weaker than the best base model and the ensemble adaptations. All remaining base models perform significantly worse.



(b) Result on the wing weight function. The ensemble adaptations belong to the better performing models.

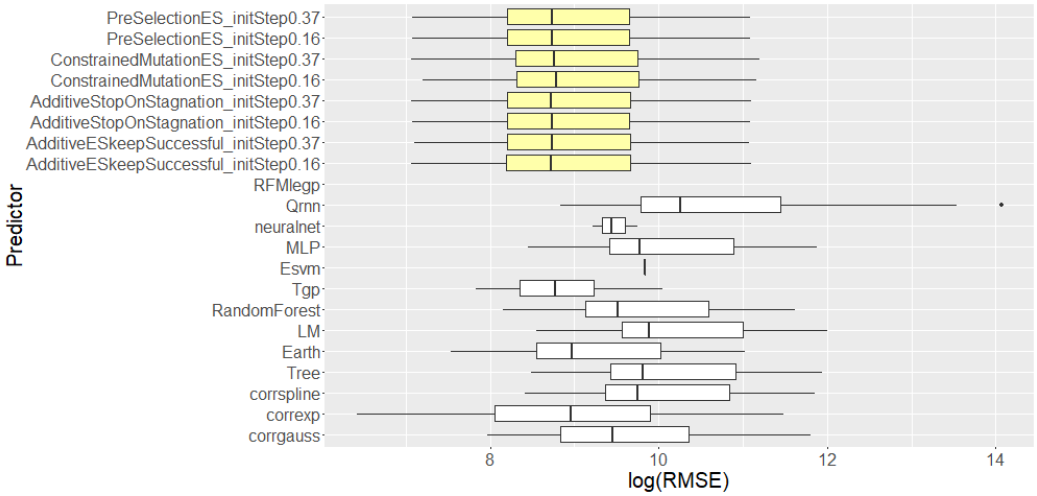


Figure A.2: The plot shows the results of the comparison of the performances, in terms of RMSE, of the different adaptations of the ensemble building method and all base models on the robot function and the wing weight function. Ensemble results are colored yellow, the base model result is shown in white.

Remarkable is that despite the large variance in the performances of the base models, the ensembles show a steady performance with a standard deviation comparable to or even slightly better than the one of the best base model.

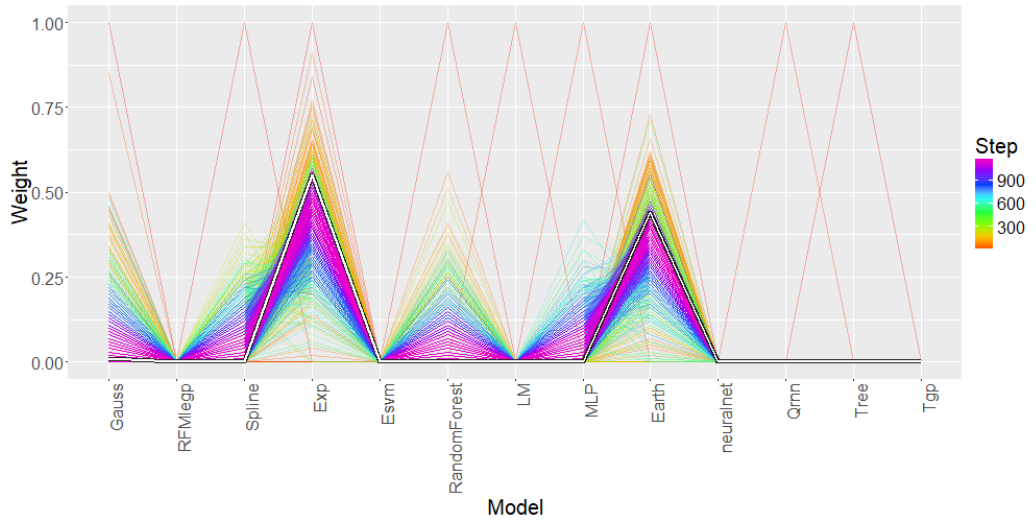
Figure A.2 shows the results for the experiments on the robot function (cf. Figure A.2a) and on the wing weight function (cf. Figure A.2b).
On the robot function, the ensembles and the best base model again show comparable performance. The base models ranked second and third, corrgauss and correxp, already perform slightly worse and all remaining base models perform significantly worse than the ensemble adaptations. Like on the otl-circuit function and the piston function this variance in the performances of the base models does not influence the good performance of the ensemble.
On the wing weight function, the results are more close; only one base model performs significantly worse than all ensemble adaptations. Still, the ensembles are among the best performing models.

In the following, a closer look is taken at the behavior of the different ensemble approaches. Exemplary plots are shown for the optimization of the weights to find the best model on the otl-circuit function (cf. Figures A.3 and A.4). The plots document the development of the weights for each step of the search. Each line represents a single individual considered during the search, and its color marks the related search step when the individual was found.

In all of these plots can be seen, that the base models RFMlegp, Esvm, neuralnet and Tgp did not receive any weight throughout the whole search. This is owed to the fact that these models failed during the preceding cross-evaluation in at least one, but rather most or even all of the evaluations. To obtain a reliable ensemble, models that failed in at least one of the evaluation steps are excluded from the search for the best weights.

Figure A.3 shows the development of the weights during the optimization using the additive approaches. The characteristics of these approaches can be easily read from the plots. The additive approach, which stops after the search stagnates (cf. Figure A.3a), ends the search after the addition of MLP. Previously added base models at least gained small weight during the search, but no improvement could be made with the addition of MLP. As a result, the search stopped.
The additive approach that does not stop on stagnation considered all base models. The course of the optimization can be read from the plot (cf. Figure A.3b). The three base models Gauss, Exp and Earth that were part of the search space from the start on, show nearly the complete range of colors.

(a) Sequential addition with stop on stagnation. After the search stagnates with the addition of MLP the search is stopped. Therefore, the base models LM, Qrnn and tree that are ranked lower than MLP are not considered in this search.



(b) Sequential addition without stop on stagnation. All base models are considered.

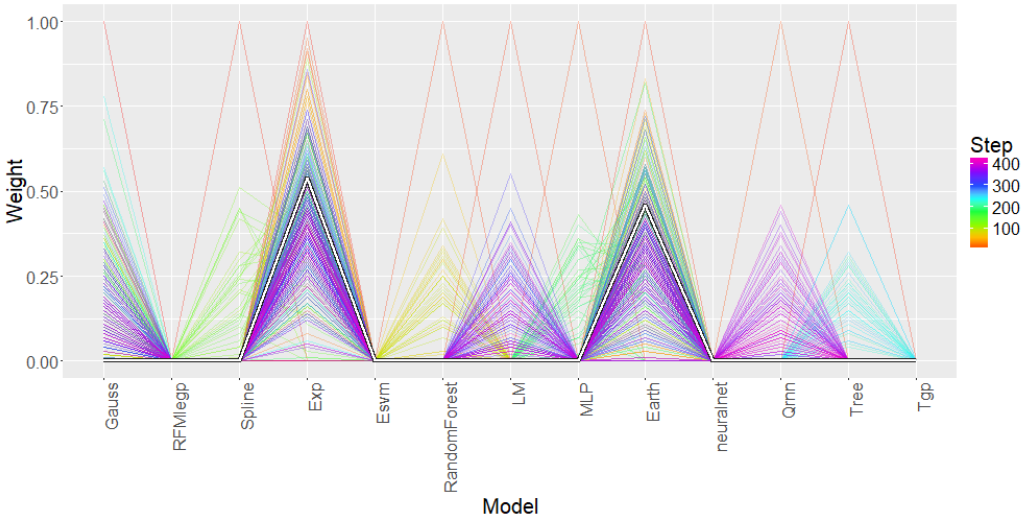
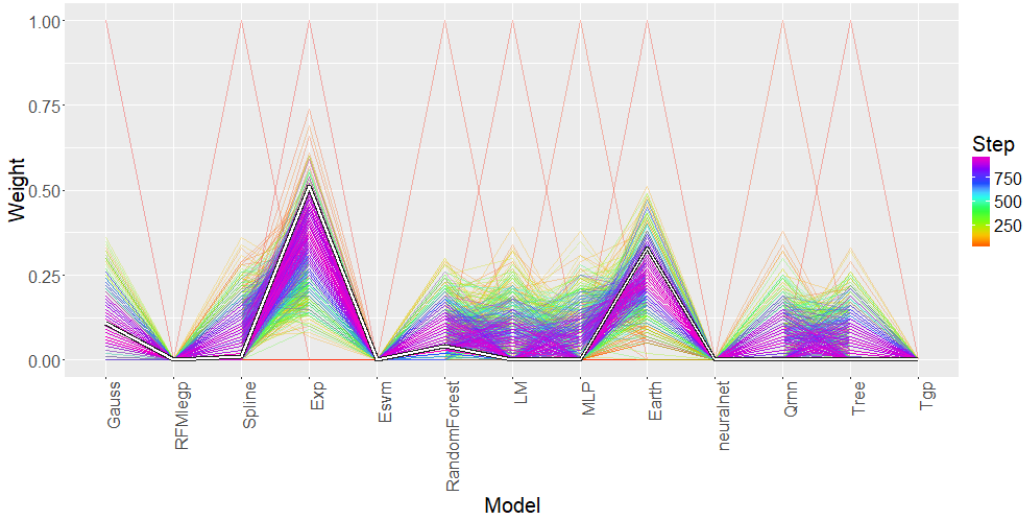


Figure A.3: The plots document the development of the weights during the optimization for the otl-circuit function using the additive approaches. Each line represents a single individual considered during the search, and its color marks the related search step when the individual was found, the white line marks the best weights combination. Though, through the course of the optimization both approaches consider different individuals, in the end, they agree on similar weight combinations.

(a) Restriction of the mutation.



(b) Preselection of models. During the automatized preselection of the base models, the search space is restricted to the three best-ranked base models.

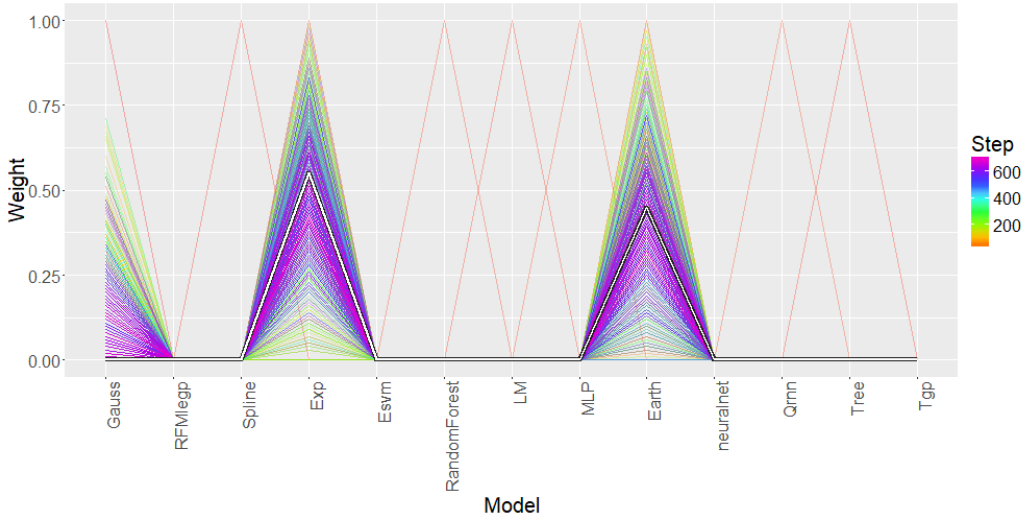


Figure A.4: The plots document the development of the weights during the optimization for the otl-circuit function using the approaches that restrict the search space and the mutation respectively. Each line represents a single individual considered during the search, and its color marks the related search step when the individual was found, the white line marks the best weights combination.

The base models that were added later in the course of the optimization show only a limited range of colors.
In the end, both approaches agree on similar best weights for the best ensemble combination.

Figure A.4 shows the development of the weights during the optimization using the approaches that restrict the considered base models or the mutation respectively. Again, the characteristics of these approaches can be easily read from the plots.
The approach that restricts the mutation searches the complete search space throughout the course of the optimization. No structure in the search can be read from the plot; the course of the search seems random. The final search result visibly differs from the weight combinations preferred by the additive approaches although most weight is given to the same base models.
The approach that initially selects a subset of the available base models for optimization using a rule based on the comparison to the mean predictor as described in Section 4.4.4. Applying this rule, the search is restricted to the base models Gauss, Exp, and Earth. The search on these base models again leads to similar weights as already preferred by the additive approaches.

A.2 The Performance of Dynamical Adapted CCM in SPO

In Section 5.2.5 the CCMs using $\tau = 1, \lambda = 10$ and $\tau = 20, \lambda = 20$ respectively were compared to two strong ensemble competitors and all base models. The results presented were condensed to the relevant information to preserve the readability. For the sake of completeness, in the following, the complete results for all base models are given and discussed.

Table A.1 presents these results. Given are the mean and standard deviation for the results of the optimization processes, best results are marked bold. To allow for a comparison of the models over the different functions, the mean results are function-wise ranked, and the average ranking, as well as the final rank for each model, is given in the two last rows.
Some of the entries show a red 'N/A'. These entries refer to experimental setups where the optimization process using the stated model did not succeed in at least one repetition. Therefore, the related rankings in the two last rows are given in brackets.

	$\tau = 1, \lambda = 10$	correxp	corrgauss	corrspline	Earth	Lm
ackley2D	2.4957 (±1.931)	7.6620 (±1.465)	2.1552 (±3.055)	3.2729 (±3.189)	5.8966 (±1.853)	3.1309 (±1.741)
ackley4D	6.8503 (±3.179)	11.588 (±1.275)	5.3801 (±3.576)	7.8778 (±4.558)	8.1570 (±1.870)	**5.2055** (±1.074)
GLG4D	22.430 (±12.00)	33.021 (±5.766)	**21.101** (±13.17)	26.344 (±10.49)	29.465 (±10.19)	32.479 (±9.483)
GLG8D	29.616 (±9.420)	57.321 (±7.660)	**28.702** (±11.39)	42.634 (±20.82)	60.226 (±4.488)	56.429 (±6.917)
otl-circuit	2.6055 (±0.002)	2.6542 (±0.034)	2.6553 (±0.070)	2.7007 (±0.070)	**2.6039** (±0.000)	N/A
piston	**0.1667** (±0.003)	0.1708 (±0.002)	0.1709 (±0.005)	0.1750 (±0.011)	0.1670 (±0.001)	0.1761 (±0.011)
robot	0.0132 (±0.020)	0.0552 (±0.027)	0.0309 (±0.015)	0.0756 (±0.044)	0.0253 (±0.022)	0.0169 (±0.036)
rosenbrock4D	**2.4138** (±1.522)	288.78 (±300.4)	3.5536 (±2.490)	60.989 (±138.8)	727.23 (±610.5)	70.933 (±63.79)
rosenbrock8D	4127.1 (±2989)	3188.6 (±1474)	6819.3 (±2799)	18887 (±12949)	23688 (±12099)	**696.83** (±582.0)
wingweight	182.50 (±13.11)	174.83 (±14.12)	178.91 (±14.44)	181.05 (±6.350)	175.94 (±6.323)	185.10 (±12.82)
AVG RANK	3.6	7.2	4.2	6.95	7.05	(5.7)
RANK	2	11	3	7	10	(5)

	$\tau = 20, \lambda = 20$	MLP	Neuralnet	RandomForest	Tgp	tree
ackley2D	1.9344 (±2.041)	**0.3425** (±0.694)	7.8315 (±2.263)	2.5598 (±1.198)	1.1200 (±0.522)	5.6125 (±1.443)
ackley4D	6.0215 (±3.242)	9.7943 (±3.098)	15.330 (±2.523)	8.9207 (±1.946)	5.8991 (±1.028)	15.540 (±0.936)
GLG4D	23.576 (±13.02)	25.799 (±10.63)	33.676 (±9.031)	30.801 (±7.160)	21.690 (±12.28)	32.181 (±8.249)
GLG8D	30.335 (±7.166)	48.353 (±7.637)	56.281 (±1.467)	53.407 (±6.857)	59.229 (±6.998)	58.156 (±6.744)
otl-circuit	2.6103 (±0.012)	**2.6039** (±0.000)	2.8980 (±0.191)	2.7849 (±0.116)	2.7580 (±0.045)	3.0001 (±0.092)
piston	0.1704 (±0.008)	0.1669 (±0.001)	0.2107 (±0.014)	0.1706 (±0.003)	0.1770 (±0.002)	0.2027 (±0.016)
robot	0.0181 (±0.018)	**0.0000** (±0.000)	**0.000** (±0.000)	0.0239 (±0.033)	0.06681 (±0.032)	0.0876 (±0.056)
rosenbrock4D	4.1977 (±2.088)	815.76 (±560.3)	N/A	253.76 (±212.2)	306.48 (±222.6)	732.86 (±689.9)
rosenbrock8D	3224.2 (±2041)	17994 (±12828)	N/A	4705.0 (±2657)	6633.4 (±3569)	23605 (±15191)
wingweight	**173.0** (±10.35)	181.99 (±11.02)	N/A	184.59 (±12.15)	182.94 (±9.371)	181.05 (±12.88)
AVG RANK	**3.4**	5.2	(6.65)	7	7	10
RANK	**1**	4	(6)	8.5	8.5	12

Table A.1: The table displays the results for the main experiment setup. Given are mean optimization result with standard deviation for the CCMs in comparison to all base models.

The most important result here is that the two CCMs are ranked first and second, unlike before, where the CCMs were ranked second and third only. This is owed to the fact, that in the previous representation for each function only the best performing base model was considered, and this best base model, though changing for each function, ranked as one in the overall ranking.
Noteworthy are also the results for the rosenbrock functions. The optimization results obtained from the different base models and the two ensembles range between 2.4 and 816 for rosenbrock4D and between 697 and 23688 for rosenbrock 8D. Though the variance in the performances of the different base models is large, the ensembles showed good performances on both functions; the CCM using $\tau = 1, \lambda = 10$ even performed best on the rosenbrock4D function.

Bibliography

[1] Zaefferer, M., Breiderhoff, B., Naujoks, B., Friese, M., Stork, J., Fischbach, A., Flasch, O., Bartz-Beielstein, T.: Tuning multi-objective optimization algorithms for cyclone dust separators. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation. GECCO '14, New York, NY, USA, ACM (2014) 1223–1230

[2] Löffler, F.: Staubabscheiden. (Aus der Lehrbuchreihe Chemieingenieurwesen/Verfahrenstechnik). Georg Thieme Verlag Stuttgart, New York (1989)

[3] Slack, M., Prasad, R., Bakker, A., Boysan, F.: Advances in cyclone modelling using unstructured grids. Chemical Engineering Research and Design **78** (2000) 1098 – 1104 Separation Processes.

[4] Elsayed, K., Lacor, C.: Optimization of the cyclone separator geometry for minimum pressure drop using mathematical models and CFD simulations. Chemical Engineering Science **65** (2010) 6048 – 6058

[5] Wasilewski, M., Duda, J.: Application of Computational Fluid Dynamics to optimization of cyclone dust separators operated in the cement industry. Volume Czasopismo Chemik 67/2013. (2013) 985–994

[6] Iman Pishbin, S., Moghiman, M.: Optimization of cyclone separators using genetic algorithm. International Review of Chemical Engineering (I.RE.CH.E.) **2** (2010)

[7] Singh, P., Couckuyt, I., Elsayed, K., Deschrijver, D., Dhaene, T.: Multi-objective geometry optimization of a gas cyclone using triple-fidelity co-kriging surrogate models. Journal of Optimization Theory and Applications **175** (2017) 172–193

[8] Allmendinger, R., Emmerich, M.T., Hakanen, J., Jin, Y., Rigoni, E.: Surrogate-assisted multicriteria optimization: Complexities, prospective so-

lutions, and business case. Journal of Multi-Criteria Decision Analysis **24** (2017) 5–24

[9] A. Arias-Montaño and C. A. Coello Coello and E. Mezura-Montes: Multi-objective airfoil shape optimization using a multiple-surrogate approach. In: 2012 IEEE Congress on Evolutionary Computation. (2012) 1–8

[10] Skinner, S., Zare-Behtash, H.: State-of-the-art in aerodynamic shape optimisation methods. Applied Soft Computing **62** (2018) 933 – 962

[11] Muyl, F., Dumas, L., Herbert, V.: Hybrid method for aerodynamic shape optimization in automotive industry. Computers & Fluids **33** (2004) 849 – 858 Applied Mathematics for Industrial Flow Problems.

[12] Jones, D.R.: Optimization in the Automotive Industry. In: Optimization and Industry: New Frontiers. Springer US, Boston, MA (2003) 39–58

[13] Jameson, A., Fatica, M.: Using computational fluid dynamics for aerodynamics. (2003) White paper, National Research Council Workshop on the Future of Supercomputing, Santa Fe, NM, 24-25 September 2003.

[14] Bartz-Beielstein, T., Lasarczyk, C., Preuss, M.: Sequential Parameter Optimization. In McKay, B., et al., eds.: Proceedings 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland, Piscataway NJ, IEEE Press (2005) 773–780

[15] Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: International Conference on Learning and Intelligent Optimization, Springer (2011) 507–523

[16] Žilinskas, A., Mockus, J.: On one bayesian method of search of the minimum. Avtomatika i Vychislitel'naya Teknika **4** (1972) 42–44

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

[18] Kenneth P. Burnham, D.R.A.: Model Selection and Multimodel Inference - A Practical Information-Theoretic Approach. Springer-Verlag New York (2002)

[19] Bartz-Beielstein, T., Zaefferer, M.: Model-based Methods for Continuous and Discrete Global Optimization. Applied Soft Computing **55** (2017) 154–167

[20] Li, R., Emmerich, M.T.M., Eggermont, J., Bovenkamp, E.G.P., Bäck, T., Dijkstra, J., Reiber, J.H.C.: Metamodel-assisted mixed integer evolution strategies and their application to intravascular ultrasound image analysis. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence). (2008) 2764–2771

[21] Bartz-Beielstein, T., Stork, J., Zaefferer, M., Rebolledo, M., Lasarczyk, C., Ziegenhirt, J., Konen, W., Flasch, O., Koch, P., Friese, M., Gentile, L., Rehbach, F.: Sequential Parameter Optimization Toolbox (Package "SPOT"). (2018) R package available at `https://CRAN.R-project.org/package=SPOT`, version 1.1.0.

[22] James, G., Witten, D., Hastie, T., Tibshirani, R.: An Introduction to Statistical Learning: With Applications in R. Springer Publishing Company, Incorporated (2014)

[23] Rencher, A.C., Schaalje, G.B.: Linear Models in Statistics. 2nd edn. Wiley-Interscience. John Wiley & Sons, Hoboken, New Jersey (2008)

[24] Lenth, R.: Response-Surface Methods in R, Using rsm. Journal of Statistical Software **32** (2009) 1–17

[25] Breiman, L., Friedman, J., Stone, C.J., Olshen, R.: Classification and Regression Trees. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis (1984)

[26] Breiman, L.: Random forests. Machine learning **45** (2001) 5–32

[27] Therneau, T.M., Atkinson, B., Ripley, B.: rpart: Recursive Partitioning. (2011)

[28] Liaw, A., Wiener, M.: Classification and regression by randomforest. R News **2** (2002) 18–22

[29] Breiman, L.: Bagging predictors. Machine learning **24** (1996) 123–140

[30] Zell, A.: Simulation neuronaler Netze. Oldenbourg (2000)

[31] Anderson, J.A., Rosenfeld, E.: Neurocomputing: Foundations of Research. MIT Press, Cambridge, MA, USA (1988)

[32] Ramcón y Cajal, S.: Histology of the Nervous System of Man and Vertebrates. Volume 1. Oxford University Press, 200 Madison Avenue, New York, New York 10016 (1995)

[33] Kandel, E.R., Schwartz, J.H., Jessell, T.M.: Principles of Neural Science. 3rd edn. Elsevier, New York (1991)

[34] Fritsch, S., Guenther, F., Suling, M., Mueller, S.: Training of Neural Networks (Package "neuralnet"). (2016) R package available at `https://CRAN.R-project.org/package=neuralnet`, version 1.32.

[35] Cannon, A.J.: Multi-Layer Perceptron Neural Network with Optional Monotonicity Constraints (Package "monmlp"). (2017) R package available at `https://CRAN.R-project.org/package=monmlp`, version 1.1.3.

[36] Cannon, A.J.: Quantile regression neural networks: implementation in R and application to precipitation downscaling. Computers & Geosciences **37** (2011) 1277–1284

[37] Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory. COLT '92, New York, NY, USA, ACM (1992) 144–152

[38] Vapnik, V.: Statistical learning theory. Wiley (1998)

[39] Smola, A.J., Schölkopf, B.: A tutorial on support vector regression. Statistics and Computing **14** (2004) 199–222

[40] Forrester, A., Sobester, A., Keane, A.: Engineering Design Via Surrogate Modelling: A Practical Guide. John Wiley & Sons (2008)

[41] Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., Chang, C., Lin, C.: Misc Functions of the Department of Statistics, Probability Theory Group (Package "e1071"). (2018) Available at `https://CRAN.R-project.org/package=e1071`, version 1.6.7.

[42] Chang, C.C., Lin, C.J.: LIBSVM: A Library for Support Vector Machines. ACM Intelligent Systems and Technology **2** (2011) 27:1–27:27

[43] Friedman, J.H., Roosen, C.B.: An introduction to multivariate adaptive regression splines. Statistical Methods in Medical Research **4** (1995) 197–217

[44] Friedman, J.H.: Multivariate Adaptive Regression Splines. The Annals of Statistics **19** (1991) 1–67

[45] Friedman, J.H.: Fast MARS. Technical Report 110, Stanford University Department of Statistics (1993)

[46] S. Milborrow. Derived from mda:mars by T. Hastie and R. Tibshirani.: earth: Multivariate Adaptive Regression Splines (Package "Earth"). (2011) R package available at `http://CRAN.R-project.org/package=earth`, version 4.4.4.

[47] Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press (2005)

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

[49] Matheron, G.: Principles of geostatistics. Economic Geology **58** (1963) 1246–1266

[50] Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Statistical science (1989) 409–423

[51] Hald, A.: On the history of maximum likelihood in relation to inverse probability and least squares. Statist. Sci. **14** (1999) 214–222

[52] Søren N. Lophaven, Hans Bruun Nielsen, J.S.: Dace - a matlab kriging toolbox. Technical report, Technical University of Denmark (2002)

[53] Gramacy, R.B.: tgp: An R package for bayesian nonstationary, semiparametric nonlinear regression and design by treed gaussian process models. Journal of Statistical Software **19** (2007) 1–46

[54] Dancik, G.M., Dorman, K.S.: mlegp: statistical analysis for computer models of biological systems using r. Bioinformatics **24** (2008) 1966

[55] Gorissen, D., Couckuyt, I., Demeester, P., Dhaene, T., Crombecq, K.: A surrogate modeling and adaptive sampling toolbox for computer based design. Journal of Machine Learning Research **11** (2010) 2051–2055

[56] Manuel López-Ibáñez and Jérémie Dubois-Lacoste and Leslie Pérez Cáceres and Mauro Birattari and Thomas Stützle: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives **3** (2016) 43 – 58

[57] Emmerich, M., Giotis, A., Özdemir, M., Bäck, T., Giannakoglou, K.: Metamodel—assisted evolution strategies. In Guervós, J.J.M., Adamidis, P., Beyer, H.G., Schwefel, H.P., Fernández-Villacañas, J.L., eds.: Parallel Problem Solving from Nature — PPSN VII, Berlin, Heidelberg, Springer Berlin Heidelberg (2002) 361–370

[58] Mockus, J., Tiesis, V., Zilinskas, A.: The application of bayesian methods for seeking the extremum. Towards Global Optimization **2** (1978) 117–129

[59] Bartz-Beielstein, T.: Spot: An r package for automatic and interactive tuning of optimization algorithms by sequential parameter optimization. (2010)

[60] Stoean, C., Preuss, M., Stoean, R., Dumitrescu, D.: Multimodal optimization by means of a topological species conservation algorithm. Trans. Evol. Comp **14** (2010) 842–864

[61] Wong, K.C., Leung, K.S., Wong, M.H.: Protein structure prediction on a lattice model via multimodal optimization techniques. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation. GECCO '10, New York, NY, USA, ACM (2010) 155–162

[62] Qing, L., Gang, W., Zaiyue, Y., Qiuping, W.: Crowding clustering genetic algorithm for multimodal function optimization. Appl. Soft Comput. **8** (2008) 88–95

[63] Gallagher, M., Yuan, B.: A general-purpose tunable landscape generator. IEEE Trans. Evolutionary Computation **10** (2006) 590–603

[64] Bartz-Beielstein, T.: How to Create Generalizable Results. In Kacprzyk, J., Pedrycz, W., eds.: Springer Handbook of Computational Intelligence. Springer Berlin Heidelberg, Berlin, Heidelberg (2015) 1127–1142

[65] Ackley, D.H.: An Empirical Study of Bit Vector Function Optimization. Genetic Algorithms and Simulated Annealing (1987) 170–215

[66] Rosenbrock, H.H.: An automatic method for finding the greatest or least value of a function. The Computer Journal **3** (1960) 175–184

[67] Bäck, T., Schwefel, H.P.: An overview of evolutionary algorithms for parameter optimization. Evol. Comput. **1** (1993) 1–23

[68] Surjanoviv, S., Bingham, D.: Virtual library of simulation experiments. [ONLINE] (2016) Available at: http://www.sfu.ca/ ssurjano/. [Accessed 30 November 2016].

[69] Ben-Ari, E.N., Steinberg, D.M.: Modeling Data from Computer Experiments: An Empirical Comparison of Kriging with MARS and Projection Pursuit Regression. Quality Engineering **19** (2007) 327–338

[70] Kenett, R., Zacks, S.: Modern Industrial Statistics: Design and Control of Quality and Reliability. Volume 41. Cengage Learning (1998)

[71] An, J., Owen, A.: Quasi-regression. Journal of Complexity **17** (2001) 588 – 607

[72] Raymer, D.P.: Aircraft Design: A Conceptual Approach and Rds-student, Software for Aircraft Design, Sizing, and Performance Set (AIAA Education). AIAA (American Institute of Aeronautics & Ast (2006)

[73] Friese, M., Bartz-Beielstein, T., Emmerich, M.: Building ensembles of surrogates by optimal convex combination. In Papa, G., Mernik, M., eds.: Bioinspired Optimization Methods and their Applications. (2016) 131–143

[74] Friese, M., Bartz-Beielstein, T., Bäck, T., Naujoks, B., Emmerich, M.: Weighted ensembles in model-based global optimization. AIP Conference Proceedings **2070** (2019) 020003

[75] Friese, M., Bartz-Beielstein, T., Bäck, T., Emmerich, M.: Optimally Weighted Ensembles of Surrogate Models for Sequential Parameter Optimization. Journal of Global Optimization (submitted to) (2019)

[76] Friese, M., Zaefferer, M., Bartz-Beielstein, T., Flasch, O., Koch, P., Konen, W., Naujoks, B.: Ensemble-Based Optimization and Tuning Algorithms. In Hoffmann, F., Hüllermeier, E., eds.: Proceedings 21. Workshop Computational Intelligence, Universitätsverlag Karlsruhe (2011) 119–134

[77] Gittins, J.C.: Bandit Processes and Dynamic Allocation Indices. Journal of the Royal Statistical Society. Series B (Methodological) **41** (1979) 148–177

[78] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning). MIT Press (1998)

[79] Hawking, S.: On the Shoulders of Giants: The Great Works of Physics and Astronomy. Penguin (2003)

[80] Sober, E.: The principle of parsimony. British Journal for the Philosophy of Science **32** (1981)

[81] Schapire, R.E.: The strength of weak learnability. Machine Learning **5** (1990) 197–227

[82] Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998) 226–239

[83] Symonds, M.R.E., Moussalli, A.: A brief guide to model selection, multimodel inference and model averaging in behavioural ecology using Akaike's information criterion. Behavioral Ecology and Sociobiology **65** (2011) 13–21

[84] Akaike, H.: 15. In: Information Theory and an Extension of the Maximum Likelihood Principle. Springer New York, New York, NY (1998) 199–213

[85] Kittler, J., Hojjatoleslami, S., Windeatt, T.: Weighting factors in multiple expert fusion. In: BMVC. (1997)

[86] Breiman, L.: Stacked regressions. Machine learning **24** (1996) 49–64

[87] Polikar, R.: Ensemble based systems in decision making. Circuits and Systems Magazine, IEEE **6** (2006) 21–45

[88] Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M.: Methods and metrics for cold-start recommendations. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '02, New York, NY, USA, ACM (2002) 253–260

[89] Hess, S.: Sequentielle modellbasierte Optimierung mit Ensembles durch einen Reinforcement-Learning-Ansatz. Master's thesis, Technische Universität Dortmund (2012)

[90] Ghosh, J., Samanta, T.: Model selection–an overview. Current Science (2001) 1135–1144

[91] Baxt, W.G.: Improving the accuracy of an artificial neural network using multiple differently trained networks. Neural Computation **4** (1992) 772–780

[92] Jakob, W., Gorges-Schleuter, M., Sieber, I., Süss, W., Eggert, H.: Solving a highly multimodal design optimization problem using the extended genetic algorithm gleam. In: 6th Internat.Conf.Computer Aided Optimum Design of Structures (OPTI 99), Orlando, Fla., March 16-18, 1999. (1999) 41.03.01; LK 01.

[93] Claeskens, G., Hjort, N.L.: Model Selection and Model Averaging. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press (2008)

[94] Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences **55** (1997) 119 – 139

[95] Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Inc., New York, NY, USA (1995)

[96] van Stein, B., Wang, H., Kowalczyk, W., Emmerich, M., Bäck, T.: Cluster-based kriging approximation algorithms for complexity reduction (2017)

[97] van Stein, B., Wang, H., Kowalczyk, W., Bäck, T., Emmerich, M.: Optimally weighted cluster kriging for big data regression. In: International Symposium on Intelligent Data Analysis, Springer (2015) 310–321

[98] Zerpa, L.E., Queipo, N.V., Pintos, S., Salager, J.L.: An optimization methodology of alkaline surfactant-polymer-flooding processes using field scale numerical simulation and multiple surrogates. Journal of Petroleum Science and Engineering **47** (2005) 197 – 208

[99] Perrone, M., Cooper, L.: When networks disagree: Ensemble methods for hybrid neural networks. Neural networks for speech and image processing (1993)

[100] Goel, T., Haftka, R.T., Shyy, W., Queipo, N.V.: Ensemble of surrogates. Structural and Multidisciplinary Optimization **33** (2007) 199–216

[101] Acar, E., Rais-Rohani, M.: Ensemble of metamodels with optimized weight factors. Structural and Multidisciplinary Optimization **37** (2009) 279–294

[102] Torgo, L.: Functional models for regression tree leaves. In: Proceedings of the Fourteenth International Conference on Machine Learning. ICML '97, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1997) 385–393

[103] Dasarathy, B.V., Sheela, B.V.: A composite classifier system design: Concepts and methodology. Proceedings of the IEEE **67** (1979) 708–713

[104] van Stein, B., Wang, H., Kowalczyk, W., Bäck, T., Emmerich, M.: Optimally weighted cluster kriging for big data regression. In Fromont, E., De Bie, T., van Leeuwen, M., eds.: Advances in Intelligent Data Analysis XIV, Cham, Springer International Publishing (2015) 310–321

[105] Ginsbourger, D., Helbert, C., Carrao, L.: Discrete mixtures of kernels for kriging-based optimization. Qual. Reliab. Eng. Int. **24** (2008) 681–691 Special Issue: The European Network for Business and Industrial Statistics (ENBIS).

[106] Friese, M., Bartz-Beielstein, T., Vladislavleva, K., Flasch, O., Mersmann,

O., Naujoks, B., Stork, J., Zaefferer, M.: Ensemble-based model selection for smart metering data (abstract). Technical report, CIplus (2012)

[107] Flasch, O., Friese, M., Zaefferer, M., Bartz-Beielstein, T., Branke, J.: Learning model-ensemble policies with genetic programming. Technical Report 3/2015, TH Köln (2015)

[108] Beyer, H.G., Schwefel, H.P.: Evolution Strategies: A Comprehensive Introduction. Natural Computing **1** (2002) 3–52

[109] Schwefel, H.P.: Evolution and optimum seeking. Sixth-generation computer technology series (1995)

[110] Bäck, T., Foussette, C., Krause, P.: Contemporary evolution strategies. Springer (2013)

[111] Kennedy, J. In: Particle Swarm Optimization. Springer US, Boston, MA (2010) 760–766

[112] Eggensperger, K., Feurer, M., Hutter, F., Bergstra, J., Snoek, J., Hoos, H., Leyton-Brown, K.: Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In: NIPS workshop on Bayesian Optimization in Theory and Practice. (2013)

[113] Mostaghim, S., Pfeiffer, F., Schmeck, H.: Self-organized invasive parallel optimization. In: Proceedings of the 3rd Workshop on Biologically Inspired Algorithms for Distributed Systems. BADS '11, New York, NY, USA, ACM (2011) 49–56

[114] Chugh, T., Sindhya, K., Hakanen, J., Miettinen, K.: A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. Soft Computing (2017)

[115] Hussein, R., Roy, P.C., Deb, K.: Adaptive Switching Strategy for Metamodeling Based Multi-objective Optimization: Part I,Generative Frameworks. Technical Report COIN Report No. 2019001, Michigan State University (2019)

[116] Hussein, R., Roy, P.C., Deb, K.: Adaptive Switching Strategy for Metamodeling Based Multi-objective Optimization: Part II, Simultaneous and Combined Frameworks. Technical Report COIN Report No. 2019002, Michigan State University (2019)

[117] Chugh, T., Jin, Y., Miettinen, K., Hakanen, J., Sindhya, K.: A surrogate-assisted reference vector guided evolutionary algorithm for computationally

expensive many-objective optimization. IEEE Transactions on Evolutionary Computation **22** (2018) 129–142

[118] Stork, J., Friese, M., Zaefferer, M., Bartz-Beielstein, T., Fischbach, A., Breiderhoff, B., Naujoks, B., Tušar, T.: Open Issues in Surrogate-Assisted Optimization. In: High-Performance Simulation-Based Optimization. Springer (2020) 225 – 244

Glossary

Abbreviations

AIC	Akaike's Information Criterion	37
CCM	Convex Combinations of Models	52
CFD	Computational Fluid Dynamics	3
DOE	Design of Experiment	25
ES	Evolutionary Strategy	64
GMSE	Generalized Mean Squared Error	45
MARS	Multivariate Adaptive Regression Splines	19
MSE	Mean Squared Error	45
RMSE	Root Mean Squared Error	45
SBO	Surrogate Based Optimization	6
SPO	Sequential Parameter Optimization	4
SPOT	Sequential Parameter Optimization Toolbox	4
SVM	Support Vector Machine	17
wRMSE	Weighted Root Mean Squared Error	90

Variables and Identifiers

α, $(1-\alpha)$	The Weights Used for the Binary Model Definition	52

α, β, γ	The Weights for the Convex Combination of Models	61
$\hat{y}$	The Predicted Function Value	9
av	A vector of flags, indicating active and suspended models	93
σ	The Step Width of the (1+1)-ES	64
σ_{init}	The Initial Step Width of the (1+1)-ES	64
σ_{min}	Minimum step size of the (1+1)-ES	99
τ	Ensemble Building Interval	88
C	Candidate set for evaluation on the objective function in each sequential step, , with respect to exploration or exploitation respectively (cf. Chapter 2.2)	27
D	Observed Data	25
d	The Dimension	61
M	The Surrogate Model	26
M^*	The Fitted Surrogate Model	26
P	The population of the (1+1)-ES	93
y	The Observed Function Value	9
ρ	Density of the Immediate Neighborhood of a Point	90
$n_{\mathbf{eval}}$	Function Evaluations per Sequential Step	26

Nederlandse Samenvatting

Een gebruikelijke methode om de extrema van een functie te berekenen als dit niet analytisch mogelijk is, is om een search uit te voeren door iteratief en strategisch punten van de functie te kiezen en te evalueren. Echter voor optimalisatie problemen uit de praktijk, is de bestedingsruimte wat betreft het aantal functie evaluaties vaak beperkt gezien de benodigde tijd en de kosten van deze functie evaluaties. Een gebruikelijke techniek, onder deze omstandigheden, is het leren van een surrogaat model, b.v., een regressie model, gebaseerd op de beschikbare evaluaties van de response functie om vervolgens dit model te gebruiken voor het bepalen van de plaats van toekomstige evaluaties. Sequential Parameter Optimization (SPO) is een wel bekende methode voor het oplossen van black-box optimalisatie problemen behept met dure functie evaluaties met gebruikmaking van surrogaat modellen.

Voor zulke optimalisatie processen, kan de keuze van het surrogaat model een significante invloed hebben op de kwaliteit en de prestatie van de optimizer. Om echter relevante beslissingen te maken over welk surrogaat model te kiezen voor een gegeven probleem, is vaak expert kennis van zowel de functie en de eigenschappen van het surrogaat model nodig. In veel voorkomende situaties, echter, is er geen voorkennis van de functie of van alle beschikbare modellen aanwezig. Geautomatiseerde methoden die geheel zelf leren welk type surrogaat model het beste past bij het probleem zouden kunnen helpen bij het overwinnen van deze moeilijkheid.

Deze thesis introduceert nieuwe methoden voor hoe je meerdere heterogene surrogaat modellen voor regressie en optimalisatie van dure black-box optimalisatie problemen beheert. Het overkoepelende doel is het vrijwaren van de gebruiker van de last met betrekking tot het kiezen van het juiste surrogaat model en het creëren van een ensemble genererende strategie dat betrouwbaar en nauwkeurig werkt voor willekeurige criterium functies en modellen. De primaire focus zijn de regressie problemen en optimalisatie processen die slechts een relatief klein aantal

functie evaluaties toelaten daar deze restrictie vaak gepaard gaat met problemen uit de praktijk.

In Hoofdstuk 3 van deze thesis wordt een taxonomie van bekende methoden die dit beogen geïntroduceerd en gespecificeerd. De methoden voor het kiezen van modellen valt uiteen in twee typen.
De 'Single Evaluation Model Selection' methoden gebruiken, in het algemeen, een vooraf gedefiniëerde strategie om het meest geschikte model te kiezen. Deze strategie kan ook gebruik maken van data verkregen uit voorgaande evaluaties. Zulke ad hoc regels laten de regels van spaarzaamheid ('parsimony') buiten beschouwing en steunen niet of nauwelijks op de data bij helpen van het kiezen van het beste model.
Methoden die als 'Multi Evaluation Model Selection' bekend staan, pakken dit probleem aan door het evalueren van alle beschikbare typen surrogaat modellen. Maar in gevallen dat er meer dan één sterk model aanwezig is in de verzameling, kan het gunstig zijn om de inferentie output van verschillende modellen te combineren.
Dat laatste is wat er gedaan wordt in de 'Model Combination' methoden. De tot nu toe bekende strategieën zijn echter, in het algemeen, op de een of andere manier beperkt (b.v., tot homogene modellen of tot bepaalde toepassingen).

In Hoofdstuk 4 worden de inzichten verkregen in Hoofdstuk 3 gebruikt om een nieuwe ensemble methode te ontwikkelen. Deze aanpak wordt op een fundamentele manier bestudeerd door eerst ensembles van twee surrogaat modellen te evalueren. Er wordt aangetoond dat de convexe combinatie van modellen gunstig kan zijn daar de convexe combinatie van voorspellingen van de twee basis modellen zowel positieve als negatieve voorspellingsfouten van de basis modellen middelt. De CCM kan wedijveren met de basis modellen en in sommige gevallen beter presteren dan deze. De inzichten verworven in de studie van convexe combinaties van twee basis modellen leiden er toe dat convexe lineaire combinaties van modellen een ideale keuze is voor het combineren van modellen. De voordelen die in deze studie gevonden zijn zijn:

- Doordat convex lineaire combinatie gebruikt wordt voor het combineren, kan een ensemble niet slechter presteren dan het zwakste basis model.
- Het ensemble kan beter presteren dan de basis modellen wanneer tegengestelde voorspellingsfouten worden gecompenseerd.
- Een CCM verdient de voorkeur boven een basis model alleen als de algehele fit van het ensemble model in feite beter is (in termen van RMSE) dan de algehele fit van elk van de basis modellen.

- De convex lineaire combinatie van de voorspellingen voor een gegeven verzameling van gewichten is gemakkelijk uit te rekenen.

Als voorbereiding op de implementatie van het algoritme voor een grote verzameling van heterogene modellen, wordt het algoritme op een algemenere manier gespecificeerd door het gebruik van drie basis modellen. Bovendien wordt 'exhaustive search' gebruikt voor het vinden van de optimale combinatie van gewichten in een vast rooster vervangen door een flexibelere (1+1)-ES. Deze aanpassingen zijn vergezeld met experimenten waarmee gegarandeerd wordt dat de veranderingen geen nadelige invloed hebben op de prestatie van de methode.

Voortbouwend op dit fundament, wordt in het vervolg, de verzameling van basis modellen uitgebreid tot de beoogde grootte en worden experimenten uitgevoerd om te garanderen dat de prestatie van de methode onder deze verandering niet zal lijden. De eerste experimenten laten zien dat de (1+1)-ES, zonder verdere aanpassingen, niet overweg kan met de uitgebreide zoekruimte. Daarom worden verschillende aanpakken besproken en geïmplementeerd om deze complicatie te lijf te gaan. Ten slotte, wordt het hoofdstuk afgerond met het vergelijken van de experimenten voor alle aanpakken op een verzameling van test functies welke gebaseerd zijn op fysische modellen. Het blijkt dat in sommige gevallen de aanpakken beter presteren dan de basis modellen. Een ander essentieel inzicht verkregen door het uitvoeren van de experimenten is dat een model soms een gunstige bijdrage tot het ensemble levert terwijl geen enkel 'high-ranked' model dit kan teweegbrengen.

In Hoofdstuk 5 wordt de ontwikkelde ensemble strategie in meer detail geïntroduceerd en aangepast voor toepassing in SPO. Deze aanpassingen zijn nodig, b.v., om de modellen continu af te stemmen ten gevolge van de aanwezigheid van dynamisch veranderende en niet-uniforme data sets. De aanpassingen zijn:

- Periodieke reconstructie van modellen en tijdelijk opschorten van modellen om rekening te houden met dynamische updates.
- Lokaal 'density weighted cross-validation' om om te kunnen gaan met niet-uniforme data distributies.
- Aanpassing van de (1+1)-ES gewichtenoptimalisatie methode om grote ensemble sets te kunnen verwerken.

De dynamisch aanpassend ensemble strategie wordt dan uitgebreid getest met betrekking tot prestatie en rekentijd in sequentiële optimalisatie processen op verscheidene criterium functies. Ten eerste, worden instanties van de methode die gebruik maakt van verschillende settings voor de reconstructie- en opschor-

tingsinterval vergeleken en de invloed van deze settings op de prestatie en rekentijd wordt geanalyseerd. Zowel wat betreft de prestatie als de rekentijd wordt de methode vergeleken met elk basis model en met twee sterke ensemble concurrenten die verschillende kenmerken die deel uitmaken van de kenmerken van de voorgestelde ensemble methode bezitten.

De resultaten laten zien dat de dynamisch aanpasbare ensemble strategie betrouwbaar presteert en wat de nauwkeurigheid betreft kan wedijveren met zowel de basis modellen als de concurrenten. Men kan aantonen dat de voorgestelde ensemble strategie methode een duidelijk voordeel oplevert vergeleken met methoden die zich beperken tot de keuze van een enkele basis model in vaste intervallen. Gegeven dat het beste model op voorhand niet bekend is, is de dynamisch aanpasbare ensemble strategie de beste keuze.
De analyse van de impact (invloed) van het reconstructie interval en het opschortingsinterval op de prestatie van de ensemble strategie laat zien dat een gewone update van de ensemble setup te prefereren is, alhoewel deze update zich moet beperken tot een kleinere deelverzameling van sterkere basis modellen die in een minder frequent interval ge-update mogen worden. Een weloverwogen keuze voor de lengte van deze intervallen heeft ook een duidelijke invloed op de rekentijd van de ensemble strategie.

Samengevat kan men zeggen dat het gelukt is om een strategie te ontwikkelen dat in de eerste plaats zo betrouwbaar en precies mogelijk werkt op willekeurige criterium functies en dat willekeurige typen surrogaat modellen gebruikt als voornaamste doelen is bereikt. Alhoewel het bewust wordt aanvaard dat dit bereikt kan worden ten koste van de ensemble rekentijd, dit nadeel verminderd kan worden door gebruik te maken van weloverwogen settings voor het reconstructie interval en opschortingstijd. Echter, als de nadruk op dure real-world applicaties valt, zullen de benodigde rekentijden verwaarloosbaar zijn.

Curriculum Vitae

Martina Marion Echtenbruck, née Friese, was born on December 2nd, 1977 in Duisburg, Germany. She obtained her general qualification for university entrance (Abitur) in 1997 at the Steinbart Gymnasium in Duisburg and immediately started the study of computer sciences at the Technical University of Dortmund where she received her diploma of computer sciences in 2008.

After the study, she worked as a software engineer (adesso AG) before she learned about the research of the academic working group on computational intelligence (spotseven) of Prof. Thomas Bartz-Beielstein at the Cologne University of Applied Sciences in 2010. She joined the group as a research scientist and tutor and started her research on ensembles of surrogate models under the supervision of Prof. Thomas Bartz-Beielstein. Moreover, since 2013 she was external Ph.D. student at Leiden Institute of Advanced Computer Science (LIACS), Faculty of Science, Leiden University under the supervision of Prof. Thomas Bäck and Dr. Michael Emmerich.