CIplus
Band 5/2016

Stacked Generalization of Surrogate Models - A Practical Approach

Thomas Bartz-Beielstein

Technology
Arts Sciences
TH Köln

Stacked Generalization of Surrogate Models A Practical Approach

Thomas Bartz-Beielstein
TH Köln
Computer Science and Engineering Science
thomas.bartz-beielstein@th-koeln.de
http://www.spotseven.de

May 29, 2016

Abstract

This report presents a practical approach to stacked generalization in surrogate model based optimization. It exemplifies the integration of stacking methods into the surrogate model building process. First, a brief overview of the current state in surrogate model based optimization is presented. Stacked generalization is introduced as a promising ensemble surrogate modeling approach. Then two examples (the first is based on a real world application and the second on a set of artificial test functions) are presented. These examples clearly illustrate two properties of stacked generalization: (i) combining information from two poor performing models can result in a good performing model and (ii) even if the ensemble contains a good performing model, combining its information with information from poor performing models results in a relatively small performance decrease only.

1 Introduction

The selection of an adequate meta model is crucial in *model based optimization* (MBO). Model based optimization plays a prominent role in todays modeling, simulation, and optimization processes. It is one of the most efficient technique for expensive and time demanding real-world optimization problems. Especially in the engineering domain, MBO is an important technique. This popularity is caused by recent advances in computer science, statistics, and engineering, in combination with progress in high-performance computing. A combination of theses advanced tools enable the treatment of problems considered unsolvable only a few decades ago. We will consider MBO in the context of global optimization.

Global optimization (GO) can be categorized on different criteria, e.g., the properties of the problems (continuous versus combinatorial, linear versus non-linear, convex versus non-convex, etc.). In many real world situations, GO problems are difficult, because nearly no structural information (e.g., number of local extrema) is available. These kind of GO problems belong to the class of black-box functions, i.e., the analytic form is unknown. Note, the class of black-box function contains also functions that are easy to solve, e.g., convex functions. The optimization problem is given by

$$\text{minimize: } f(\vec{x}) \quad \text{subject to } \vec{x}_l \leq \vec{x} \leq \vec{x}_u,$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is referred to as the objective function and $\vec{x}_l$ and $\vec{x}_u$ denote the lower and upper bounds of the search space (region of interest), respectively. This setting arises in

many real-world systems, i.e., when the explicit form of the objective function f is not readily available or e.g., user has no access to the source code of a simulator.

The term GO will be used for algorithms that are trying to find and explore global optimal solutions with complex, multimodal objective functions [36]. We will use an algorithmic view, i.e., we will consider the properties of algorithms.

First, we will describe stochastic (random) search algorithms and show how surrogate model based optimization can be classified in the context of GO. Therefore we introduce the following taxonomy.

1. Deterministic
2. Random Search
   (a) Instance based.
   (b) Model based optimization (MBO).
      i. Distribution based.
      ii. Surrogate Model Based Optimization (SBO).
         A. Single surrogate based.
         B. Multi-fidelity based.
         C. Evolutionary surrogate based.
         D. Ensemble surrogate based.

Then, we will try to answer the question of selecting an adequate surrogate model in the context of SBO.

Tis report is structured as follows: First, SBO is presented in the context of stochastic search algorithms (Section 2). Section 3 presents some general considerations about using multiple surrogate models. The *sequential parameter optimization* (SPO), which uses SBO, is introduced in Section 4. Stacked generalization is one important modeling technique in SPO. An industrial application is used in Section 5 for introducing the key features of the stacked generalization approach. To gain further insight, a second study, which uses artificial test functions, is presented in Section 6. This report concludes with a short summary and an outlook in Section 7.

2 Stochastic Search Algorithms

To motivate the importance of model selection in MBO, we will describe the related GO algorithms first. Stochastic search algorithms perform an iterative search. They use a stochastic procedure to generate the next iterate. The next iterate can be a candidate solution to the GO or a probabilistic model, where solutions can be drawn from. Stochastic search algorithms do not depend on any structural information of the objective function such as gradient information or convexity. Hence, they are robust and easy to implement. Stochastic search algorithms can further be categorized as *instance-based* or *model-based* algorithms [51]

Instance-based algorithms use a single solution, $\vec{x}$, or *population*, $P(t)$, of candidate solutions. The construction of new candidates depends explicitly on previously generated solutions. Prominent examples are simulated annealing or evolutionary algorithms.

Model-based optimization algorithms generate a population of new candidate solutions $P'(t)$ by sampling from a model. In statistics, the terms *model* and *distribution* are used synonymously. Therefore, we will use the term *surrogate model*, when we are referring to an explicit model. The model (or distribution) reflects structural properties of the underlying true function, say f. Adapting the model (or the distribution), the search is directed into regions with improved solutions. One of the key ideas in MBO can be formulated as follows: replace expensive, high fidelity, fine grained function evaluations, $f(\vec{x})$, with evaluations, $\hat{f}(\vec{x})$, of an adequate surrogate model, say M. Surrogate models also known as the cheap model, the response surface, the meta model, the approximation, or the coarse grained model.

2.1 Distribution-based Approaches

Distribution-based optimization algorithms maintain a distribution as a metamodel. A sequence of iterates, i.e., probability distributions, $\{p(t)\}$ is generated. The iterates should have the property, that

$$p(t) \to p^* \text{ as } t \to \infty,$$

where p^* is the limiting distribution, which assigns most of its probability mass to the set of optimal solutions. Note, distribution-based optimization algorithms propagate a probability distribution from one iteration to the next, whereas instance-based algorithms propagate candidate solutions $\vec{x}$.

Estimation of distribution algorithms (EDA) are very popular in the field of *evolutionary algorithms* (EA). Variation operators such as mutation and recombination are replaced by a distribution based procedure. A probability distribution, which is estimated from promising candidate solutions from the current population, is used to generate new population. [27] review different ways for using probabilistic models. [18] discuss advantages and outline many of the different types of EDAs. [19] present recent approaches and a unified view.

Although distribution-based approaches play an important role in GO, they will not be discussed further in this report. We will concentrate on surrogate model based approaches, which have their origin in statistical design and analysis of experiments, especially in response surface methodology [12] [31].

2.2 Surrogate Model-based Approaches

In general, the term *surrogate* is used, when the outcome of a process cannot be directly measured. A surrogate tries to imitate the behavior of the real model as closely as possible while being computationally cheaper to evaluate. Simple surrogate models can be constructed using a data-driven approach. They can be refined by integrating additional points or domain knowledge, e.g., constraints.

A wide range of surrogates were developed in the last decades. This results in complex design decisions. Following the discussion in [47], surrogate design decisions are necessary for the selection of (i) metamodels, (ii) designs, and (iii) model fitting methods.

Metamodels Typical metamodels are (a) classical regression models such as polynomial regression or response surface methodology [12] [31] (b) support vector machines} (SVM) [46] (c) neural networks [52] (d) radial basis functions [35], or (e) Gaussian process (GP) models, also known as design and analysis of computer experiments or Kriging [42] [7], [1], [26], [41]. [10] presents a comprehensive introduction to SBO.

Designs A broad variety of designs are available, e.g., classical experimental designs such as factorial, fractional factorial, central composite, or A-, D-optimal (alphabetically) designs. Alternatively, space filling designs, such as simple grids, Latin hypercube designs, orthogonal or uniform designs can be used. In addition, hybrid designs, random or human selection, and sequential design methods are available.

Model Fit Model fitting can be based on several criteria, e.g., weighted least squares regression or maximum likelihood estimation. Special fitting techniques exist for specific modeling approacches, e.g., backpropagation for neural networks.

2.3 SBO Applications

Simulation-based design of complex engineering problems, e.g., *computational fluid dynamics* (CFD) and *finite element modeling* (FEM) methods are the most popular application areas for SBO. To generate exact solutions, the corresponding solvers require a large number of expensive computer simulations. Generally, there are two SBO variants: (i) metamodel based methods, which use one or several different metamodels and (ii) multi-fidelity approximation methods, which uses several instances with different parameterizations of the same metamodel.

Helicopter rotor design optimization [4], aerodynamic shape design [13], multi-objective optimal design of a liquid rocket injector [38], and aerospace design [11] are only a few examples of SBO in industry.

Multi-fidelity approximation is used in [20], who use several simulation models with different grid sizes in FEM and in [44], who optimize a sheet metal forming process.

2.4 Surrogate-assisted Evolutionary Algorithms

Surrogate-assisted EA use a cheap surrogate model to replace evaluations of expensive objective function. Several variants of surrogate-assisted EAs were developed in the last years, e.g., a combination of a genetic algorithm and neural networks for aerodynamic design optimization [16], an approximate model of the fitness landscape using Kriging interpolation to accelerate the convergence of EAs [39], an *Evolution strategy* (ES) with neural network based fitness evaluations [24], or a surrogate-assisted EA framework with online learning [50]. A survey of surrogate-assisted EA approaches is presented in [23]. SBO approaches for evolution strategies are described in [9].

3 Multiple Models and Model Selection

Instead of using one surrogate model only, several models M_i, $i = 1, 2, \ldots, p$, generated and evaluated in parallel can be used. Each model $M_i : X \rightarrow y$ uses the same candidate solutions, X, from the population P and the same results, y, from expensive function evaluations.

Multiple models can also be used to partition the search space. Here we can mention tree-based Gaussian processes, which use regression trees to partition the search space and fit local GP surrogates in each region [15]. A tree-based partitioning of an aerodynamic design space, which uses independent Kriging surfaces in each partition, is described in [33]. The combination of an evolutionary model selection algorithm with *expected improvement* (EI) criterion, which selects the best performing surrogate model type at each iteration of the EI algorithm was proposed by [8].

Ensembles of surrogate models gained popularity. An adaptive weighted average model of the individual surrogates was presented in [49]. An approach which uses the best surrogate model or a weighted average surrogate model instead was introduced in [14]. In these approaches, the models for the ensemble are chosen based on their performance. Usually, the weights are adaptive and inversely proportional to the local modeling errors.

The simplest model selection process is a refinement method: the same (initial) model will be refined during the optimization. This method requires a selection criteria for sampling new points, so-called *infill points*. The balance between exploration, i.e., improving the model quality (related to the model, global), and exploitation, i.e., improving the optimization and determining the optimum (related to the objective function, local) plays a central role in this strategy. *Expected improvement* (EI) is a popular adaptive sampling method [30] [25].

The EI approach handles the initialization and refinement of a surrogate model, but not the selection of the model itself. For example, the popular *efficient global optimization* (EGO) algorithm uses a Kriging model, because Kriging inherently determines the prediction variance (necessary for the EI criterion). But there is no proof that Kriging is the best choice. Alternative surrogate models, e.g., neural networks, regression trees, support vector machines, or lasso and ridge regression may be better suited. However: An a priory selection of the best suited surrogate model is conceptually impossible in the framework treated in this report, because of the black-box setting.

Regarding the model choice, the user can decide whether to use (i) one single model, i.e., one unique global model or (ii) multiple models, i.e., an ensemble of different, possibly local, models.

Now, we do not consider the selection of a new sample point (as done in EI). Instead, we

consider criteria for the selection of one (or several) surrogate models. Usually, surrogate models chosen according to their estimated true error [22], [43].

Commonly used performance metrics are the mean absolute error (MAE) or the root mean square error (RMSE). Generally, attaining a surrogate model that has minimal error is the desired feature. Methods from statistics, statistical learning [17] and machine learning [32] are popular, e.g., simple holdout methods, cross-validation, or the bootstrap.

An alternate approach is presented in [28]. Here, the model error is not the only criterion for selecting surrogate models. The authors present an evolvability learning of surrogates approach, which uses fitness improvement for determining the quality of surrogate models.

4 Sequential Parameter Optimization

The *sequential parameter optimization* (SPO) uses a centralized, global information based approach for handling surrogate model information. It implements a stacked generalization approach developed by [48]. Early versions of the SPO [3], [2] combined methods from *design of experiments* (DOE) [37], *response surface methodology* (RSM) [5] [31], *design and analysis of computer experiments* (DACE) [29] [41], and regression trees [6]. The statistical analysis and an understanding of optimization algorithms was the main goal of the SPO. In addition, it was recognized that SPO can be used as an optimizer.

The SPO provides a sequential, model based approach to optimization and is nowadays an established parameter tuner and an optimization algorithm. It was extended in several ways, e.g., [21] benchmark an SPO derivative, the so-called *sequential model-based algorithm configuration* (SMAC) procedure, on the BBOB set of blackbox functions. Given a small budget of $10 \times d$ evaluations of d-dimensional functions, SMAC in most cases outperforms the state-of- the-art blackbox optimizer CMA-ES.

The most recent version, SPO2, is currently under development. It integrates state-of-the-art ensemble learners. The SPO2 ensemble engine, which is described in this report, uses a portfolio of surrogate models, such as regression trees and random forest, least angle regression (lars), and Kriging as level-0 models. It uses cross validation to generate a weighted combination of several surrogate models to build a generalized level-1 model.

Stacked generalization is implemented to combine several level-0 models of different types with one level-1 model into an ensemble [48]. The level-1 training algorithm is a simple linear model.

The SPO2 ensemble engine can lead to significant performance improvements. [40] present a comparison of different data driven modeling methods, e.g., a Bayesian model, several linear regression models, a Kriging model, and genetic programming. These methods were used to model the behavior of a robust gas sensor. The underlying data has a limited amount of samples and a high variance.

5 Stacked Generalization in Practice. Part I: Industrial Application

This section illustrates in detail how the stacked generalization works. It uses the programming language Python, see `https://www.python.org`. Section 5.1 describes the technical requirements, e.g., the Python libraries, which are needed to perform the experiments. Section 5.2 describes he data. The k-fold cross validation is prepared in Section 5.3. How models are added to the SPOT2 ensemble engine is explained in Section 5.4. Cross-validation for the stacking procedure is described in Section 5.5. The level-1 model construction and how it can be used for predictions is shown in Section 5.6. A schematic illustration of the stacked generalization is shown in Fig. 1.

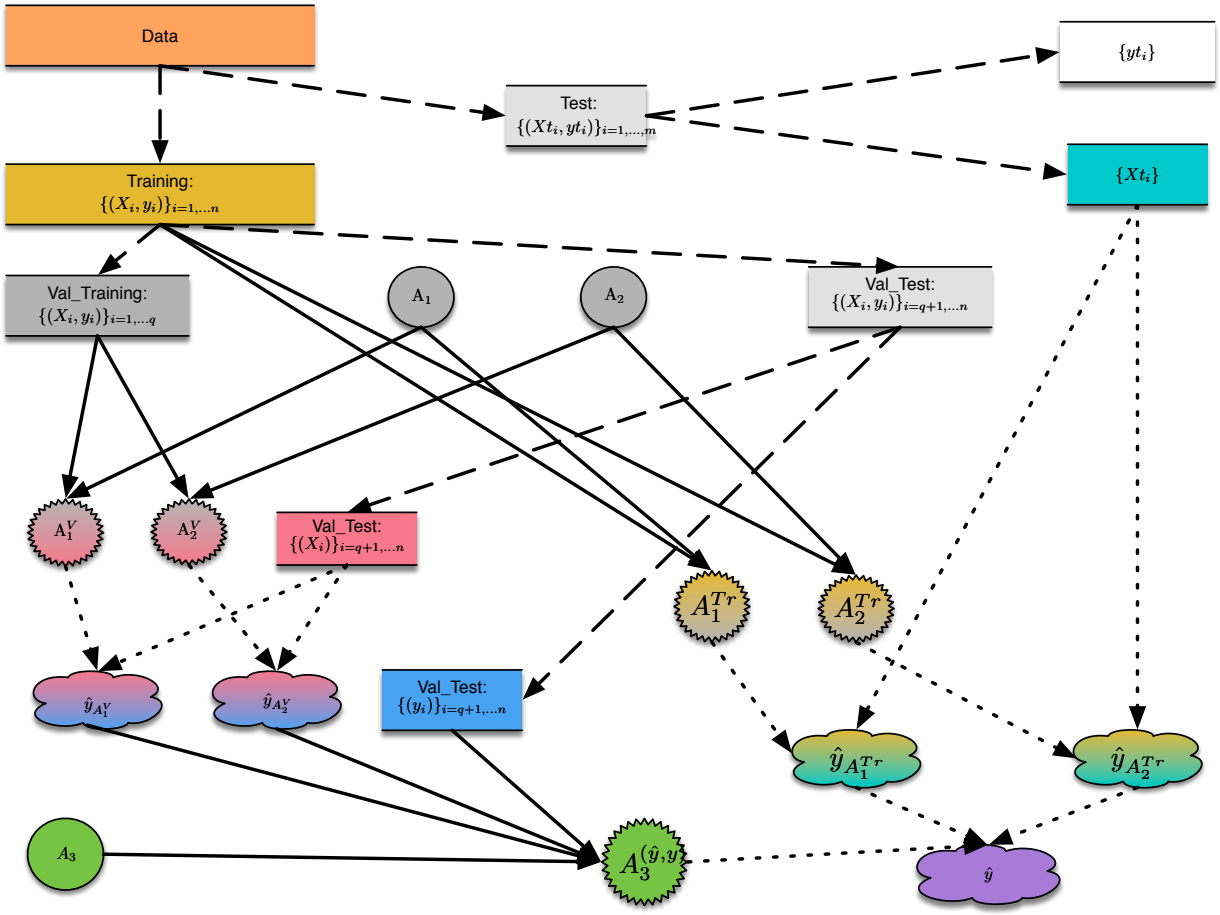

Figure 1: Illustration of the data flow in stacked generalization. The data is split into a test and training set. The training data is used for CV, i.e., in each fold, a new training and validation data set is generated. Here, we consider two algorithms, say A_1 and A_2, which are used to generate two level-0 models, A_1^V and A_2^V. These models are used for predictions on the validation data set, which result in predictions $\hat{y}_{A_1^V}$ and $\hat{y}_{A_2^V}$. A level-1 algorithm, A_3 is fitted to the data sets $\hat{y}_{A_1^V}$, $\hat{y}_{A_2^V}$, and the y_i's from the validation data set. The resulting model, $A_3^{(\hat{y},y)}$, is used for the final predictions.

5.1 Technical Requirements

This section illustrates and demonstrates the key ingredients of the SPO2 ensemble engine. It implements ideas from [48] and is based on Python code from [34]. First, we have to

1. import libraries and
2. set the SPO2 parameters, i.e., the number of folds for the cross-validations.

```
import sys
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import statsmodels.formula.api as sm
import math
from IPython.display import set_matplotlib_formats
from __future__ import division
from sklearn.model_selection import KFold
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from pandas import read_csv

np.random.seed(0) # seed to shuffle the train set
n_folds = 10
```

5.2 The Data

The complete data set is described in [40]. It consists of two separate data sets for two different gas sensors: one training data set and one test data set. Here, we consider data from the second sensor. There are seven input values and one output value (y). The goal of this study is to predict the outcome y using the seven input measurements.

```
In [10]: dfTrain = read_csv('training.csv')
         dfTest = read_csv('testing.csv')
         XTrain = dfTrain.ix[:,0:7]
         yTrain = dfTrain.ix[:,7:9]
         yTrain1 = yTrain.ix[:,1]
         X = XTrain.as_matrix()
         y = yTrain1.as_matrix()
         XTest1 = dfTest.ix[:,0:7]
         yTest1 = dfTest.ix[:,7:9]
         yTest = yTest1.ix[:,1]
         XTest = XTest1.as_matrix()
```

5.3 CV Splits

The training data are split into folds. `KFold()` divides all the samples in $k = n_{folds}$ groups of samples (called folds) of equal sizes (if possible). The prediction function is learned using $k-1$ folds, and the fold left out is used for test.

```
In [11]: skf = KFold(n_folds);
         skf.get_n_splits(X, y);
```

5.4 Level-0 Models in the Ensemble

A linear regression model and a random forest regression model are included in this study. Additional models such as Lasso or Gaussian process models can be included very easily.

```
In [12]: models = [LinearRegression()
                  , RandomForestRegressor()
                  ]
```

5.5 Cross-Validation

Let n denote the size of the training set (number of samples in the training set) and p the number of models. Summarizing, we will consider the following dimensions:

- n: size of the training set (samples)
- k: number of folds for CV
- p: number of models
- m: size of the test data (samples)

We will use two matrices to store the CV results:

1. Y_{CV} is a $(n \times p)$-matrix. It stores the results from the cross validation for each model. The training set is partitioned into k folds (n_folds=k).
2. Y_{BT} is a $(m \times p)$-matrix. It stores the aggregated results from the cross validation models on the test data. For each fold, p separate models are build, which are used for prediction on the test data. The predicted values from the k folds are averaged for each model, which results in $(m \times p)$ different values.

```
In [13]: YCV = np.zeros((X.shape[0], len(models)))
         YBT = np.zeros((XTest.shape[0], len(models)))

         for j, AV in enumerate(models):
                 YBT_j = np.zeros((XTest.shape[0], skf.n_folds))
                 for i, (train, val) in enumerate(list(skf.split(X,y))):
                     XValTraining = X[train,]
                     yValTraining = y[train]
                     XValTest = X[val]
                     AV.fit(XValTraining, yValTraining)
                     YCV[val, j] = AV.predict(XValTest)
                     YBT_j[:, i] = AV.predict(XTest)
                 YBT[:,j] = YBT_j.mean(1)
```

5.6 The Level-1 Model

5.6.1 Model Building

The level-1 model is a function of the CV-values of each model to the known, training y-values. It provides an estimate of the influence of the single models. For example, if a linear level-1 model is used, the coefficient β_i represents the effect of the i-th model.

5.6.2 Model Prediction

The level-1 model is used for predictions on the Y_{BT} data, i.e., on the averaged predictions of the CV-models. It is constructed using the effects of the predicted values of the single models (determined by linear regression) on the true values of the training data. If a model predicts a similar value as the true value during the CV, then it has a strong effect.

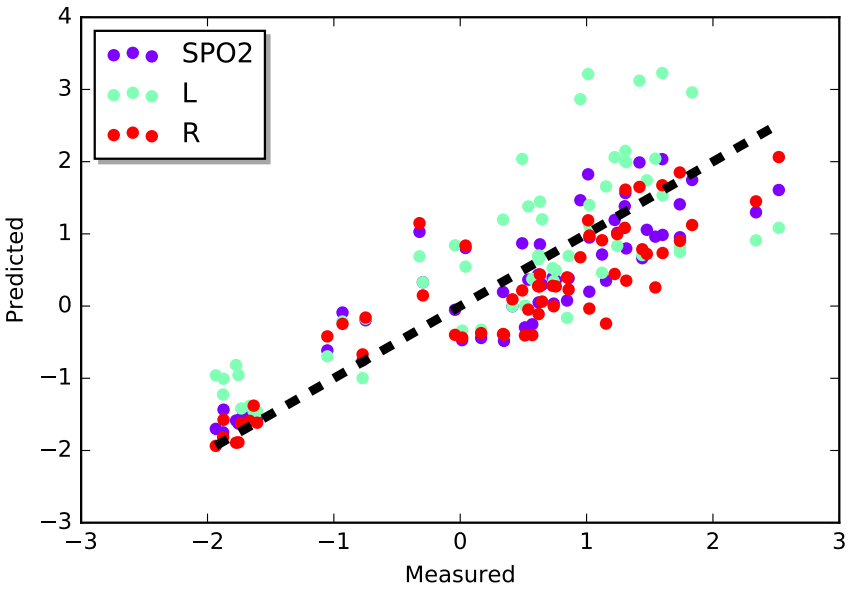

Figure 2: Visualization of the results from the first part of this study, which is based on the sensor data. Predicted values plotted against measured values. Violet dots represent results from the SPO2 ensemble approach, which combines information from the linear and the regression models. Results from the single model based approaches are also shown: green dots represent results from the linear model, whereas red dots represent results from the random forest.

The final predictions are made using the coefficients (weights) of the single models on the Y_{BT} data. Note that the Y_{BT} data are the predicted values from the corresponding models on the final test data.

```
In [14]: A3 = LinearRegression()
         A3.fit(YCV, y)
         yHat = A3.predict(YBT)
```

5.7 Results

A comparison of the mean squared error from the SPO2 ensemble and the single models reveals that the SPO2 ensemble outperforms the single models. This result can also be illustrated using a plot of the predicted versus the measured values: data from the SPO2 ensemble are closer to the bisecting line.

Numerical values read as follows:

```
SPO2 (MSE): 0.284948273406
L    (MSE): 0.673695001324
R    (MSE): 0.367652881967
```

Since smaller results are better, it can be seen that the ensemble-based approach, which combines information from the two level-0 base models, performs best.

The SPO2 approach also outperforms the remaining modeling approaches in [40], where two data sets from sensors were fitted. The MSE was chosen as a quality criterion for the models. Results from the first sensor read as follows: Linear model (0.76), OLS (0.79), Lasso (0.56), Kriging (0.57), Bayes (0.79), genetic programming (0.58), and SPO2 (0.38). Results from the second sensor are: Linear model (0.67, the same value as in this report), OLS (0.80), Lasso (0.49), Kriging (0.49), Bayes (0.79), genetic programming (0.27), and SPO2 (0.28).

Table 1: Results from the experiments with the artificial test functions. Mean and standard deviation (in brackets) from $r = 100$ repeats. R^2 values are shown.

Function	SPO2	Linear	Random Forest	Kriging (GP)
f_1	0.7821 (0.0331)	0.4025 (0.0713)	0.7856 (0.0319)	0.7655 (0.0356)
f_2	0.7951 (0.0360)	0.2145 (0.0766)	0.7949 (0.0360)	0.7951 (0.0360)
f_3	0.7940 (0.0278)	0.1168 (0.0569)	0.7924 (0.0274)	0.7939 (0.0278)
f_4	0.7414 (0.0578)	0.0065 (0.01490)	0.7530 (0.0513)	0.3172 (0.0794)
f_5	0.8363 (0.0238)	0.836 (0.0238)	0.8363 (0.0237)	0.8363 (0.0238)
f_6	-0.0203 (0.1031)	-0.0003 (0.0151)	0.3586 (0.0623)	0.1004 (0.0536)

6 Stacked Generalization in Practice. Part II: Artificial Test Functions

6.1 Why are Ensembles Better?

Results demonstrate that the combination of two models (linear regression and random forrest), which separately perform poorly, can result in an ensemble model, that performs excellent.

To investigate this behavior, we performed additional experiments using an artificial test suite.

6.2 Artificial Test Functions

Motivated by [45], we consider the following six test functions. All simulations involve a univariate random variable X drawn from a uniform distribution in [−4, +4].

Let $I(\cdot)$ denote the usual indicator function and ϵ is drawn from an independent standard normal distribution in all simulations. The outcome follows the function described below:

$$f_1(x) := -2 \times I(x < -3) + 2.55 \times I(x > -2) - 2 \times I(x > 0) + 4 \times I(x > 2) - 1 \times I(x > 3) + \epsilon$$
$$f_2(x) := 6 + 0.4x - 0.36x^2 + 0.005x^3 + \epsilon$$
$$f_3(x) := 2.83 \sin(\pi/2x) + \epsilon$$
$$f_4(x) := 4.0 \sin(3 \times \pi x) \times I(x \geq 0) + \epsilon$$
$$f_5(x) := x + \epsilon$$
$$f_6(x) := x + \epsilon, \text{ with } x \text{ realization of } X \sim N(0, 1)$$

Plots of the six test functions are shown in Fig. 3.

6.3 Results

A sample of size $r = 100$ will be drawn for each scenario. The coefficients can be interpreted as weights in the linear combination of the models. We will consider R^2 (larger values are better) and standard deviation. Results from the six experiments are summarized in Table 1. To analyze the behavior in detail, two kind of boxplots are shown for each test function.

1. The first boxplots (on left in each figure) show the coefficients of the level-1 linear model, i.e., the first entry represents the value of the intercept or β_0. Three additional boxplots are shown to visualize the effect of the single models, i.e., values of the coefficients β_1 (linear model), β_2 (random forest), and β_3 (Gaussian process model, Kriging) are shown.

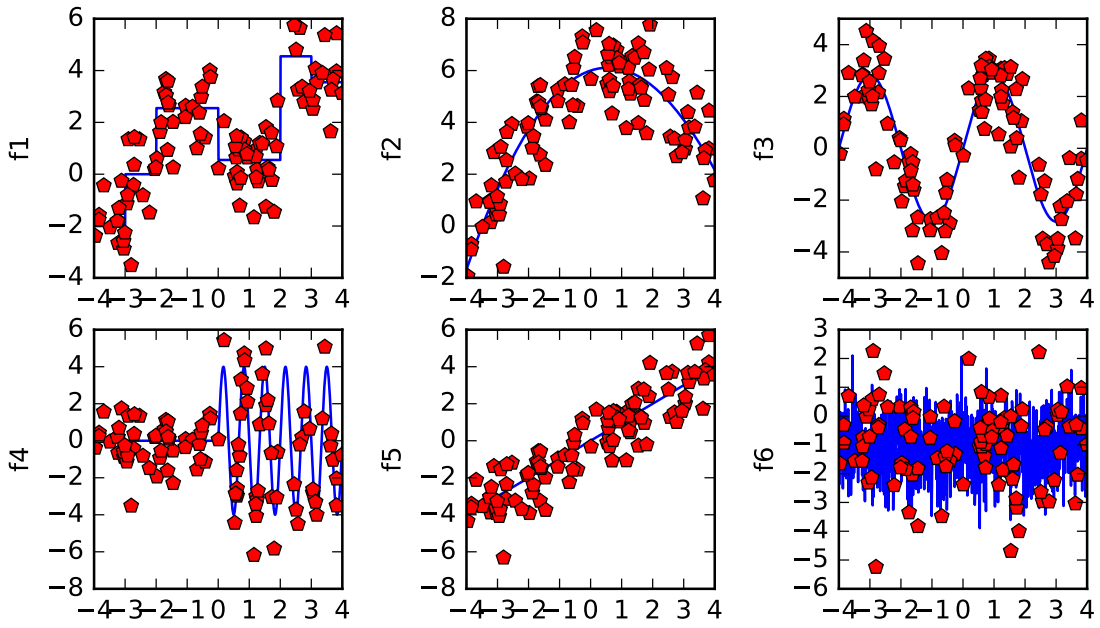

Figure 3: Plots of the six test functions. First row, from left to right: Step function f_1, polynomial function f_2, sine function f_3. Second row, from left to right: Combined function f_4, linear function f_5, random function f_6. Blue lines illustrate the true function, red dots represent (noisy) samples presented to the approximation models.

2. The second set of boxplots (on the right in each figure) show the performance of the four different modeling approaches, i.e., SPO2 (0), linear model (1), random forest (2), and Gaussian process model (3).

This arrangement of boxplots should reveal correlations between the β values from the level-1 model and the R^2 performance.

While analyzing the results from the step function, f_1, the boxplots indicate that random forest outperforms the other modeling approaches (Fig. 4). Slightly worse results were obtained by the Gaussian process models, whereas the linear model performs worse. The SPO2 ensemble engine was able to identify the good performers. The random forest coefficient, i.e., β_2 is the largest coefficient in the level-1 model.

While analyzing the results on the polynomial function, f_2, the outcome is unambiguous (Fig. 5). The Gaussian process model exhibits the best performance, and the random forest performs equally well. The linear model is not able to find a good fit. This situation is reflected in the values of coefficients of the level-1 model: the β_3 values, which are associated with the Gaussian process model, are high, whereas the values of the remaining β coefficients are negligible. A similar situation occurs during the results on the sine function, f_3 (Fig. 6).

Results from the combined function are more interesting (Fig. 7). Here, the random forest model results in the best fit (with respect to the R^2 error). The second best model, i.e., the Gaussian process model, performs better than the linear model. This ranking is mirrored in the values of the β coefficients.

Since the linear model was always the worst model, it might be of interest to see if this observation still holds for the approximation of data which have a simple linear relationship. a linear model. Therefore, test function f_5 was added to the test function portfolio from [45]. Every model was able to generate a good approximation as can be seen in Fig. 8. Interestingly, the SPO2 engine prefers the linear function as can be seen from the values of the β_1 coefficient.

Finally, to present a situation where every approximation must fail, a purely random function with additional noise was added to our portfolio (Fig. 9). The boxplots shown that no model was able get a good R^2 value. The ensemble approach fails completely in this setting.

7 Summary and Outlook

This study nicely illustrates some important benefits of the stacked generalization approach:

- Combining model information, even from models with poor approximation capabilities, might result in an improved approximation.
- Combining model information results only in a small performance decrease, if the portfolio contains a good performing model

The question, why ensembles perform better, is the subject of on-going research. Results from this study indicate, that the stacked generalization approach might work. However, this is not a proof, only an indication that it might be worth going into this direction. An extended study, which compares more surrogate models and includes more test functions, will be published soon.

The good performance of the ensemble approach does not come for free. The additional evaluations slow down the modeling process. In situations, where no information about the structure of the fitness landscape is available, the stacked generalization approach might be an adequate first choice. Models can be added or deleted from the portfolio dynamically during the search.

Acknowledgement

This work has been supported by the *Bundesministeriums für Wirtschaft und Energie* under the grants KF3145101WM3 und KF3145103WM4. This work is part of a project that has received funding from the *European Union's Horizon 2020 research and innovation program* under grant agreement No 692286.

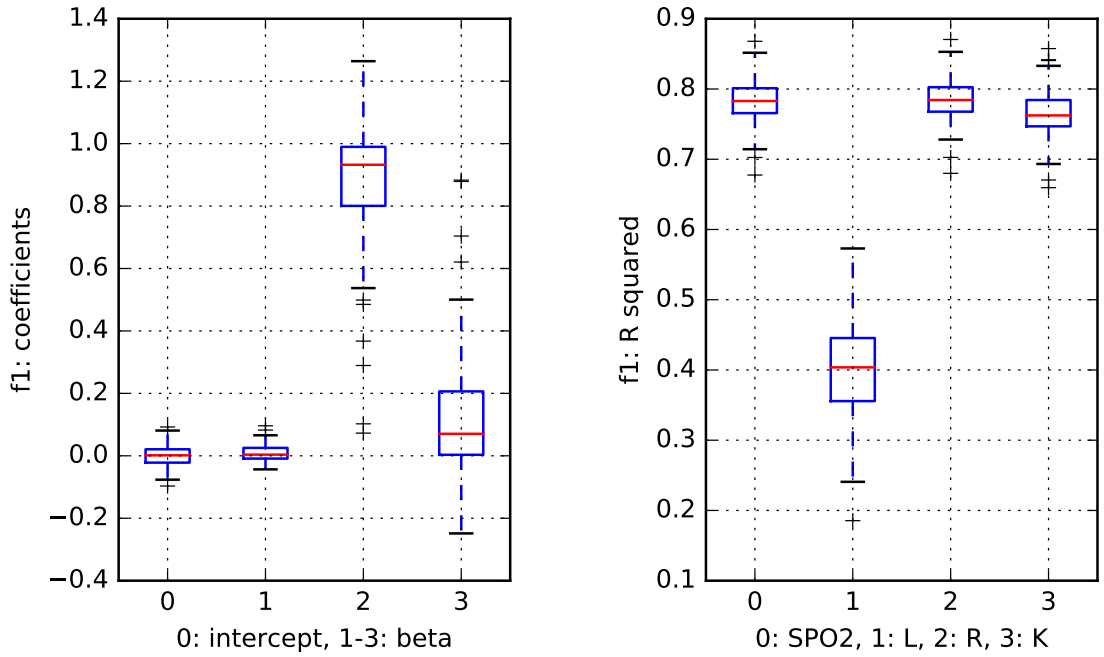

Figure 4: Results from the step function, f_1. *Left:* Boxplots showing the β coefficients of the level-1 model. *Right:* Boxplots showing the R^2 values.

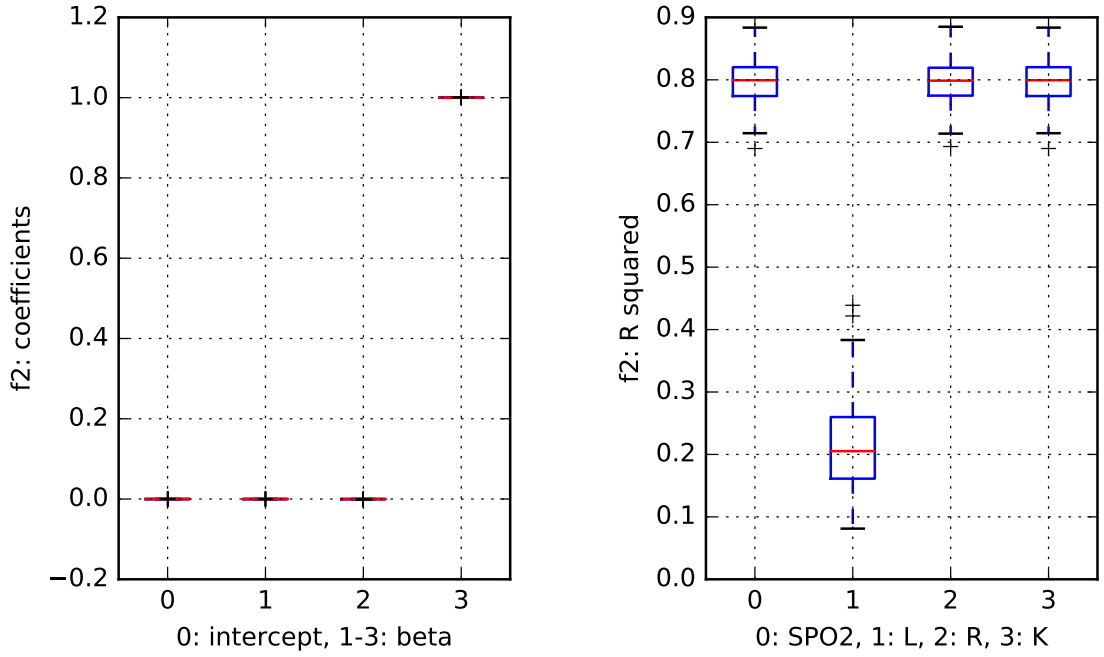

Figure 5: Results from the polynomial function, f_2. *Left:* Boxplots showing the β coefficients of the level-1 model. *Right:* Boxplots showing the R^2 values.

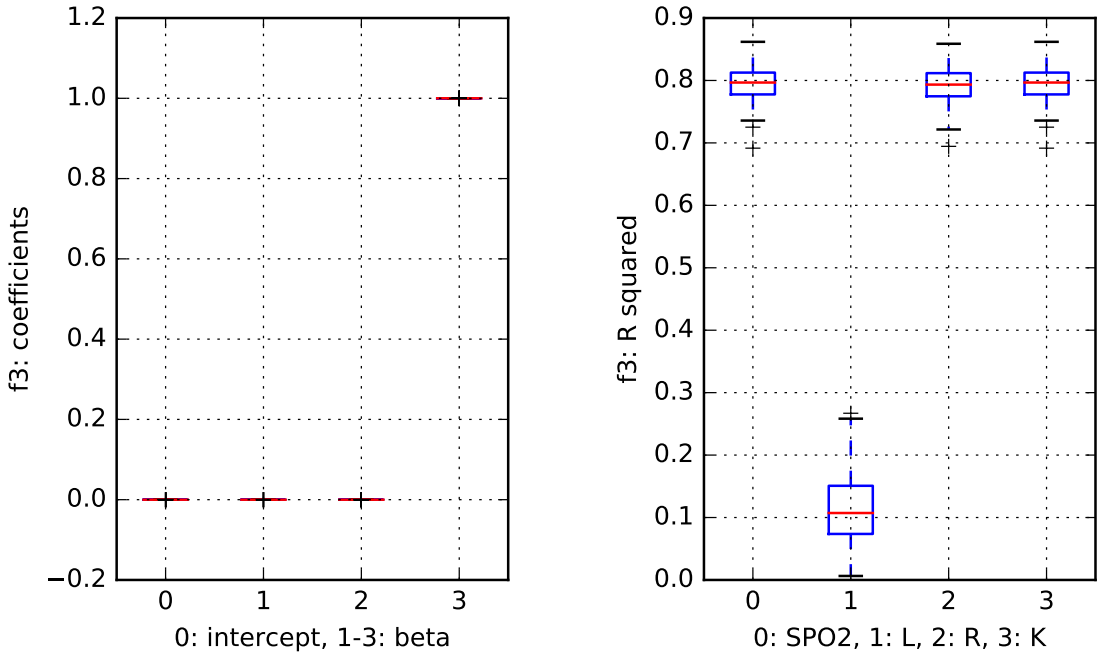

Figure 6: Results from the sine function, f_3. *Left:* Boxplots showing the β coefficients of the level-1 model. *Right:* Boxplots showing the R^2 values.

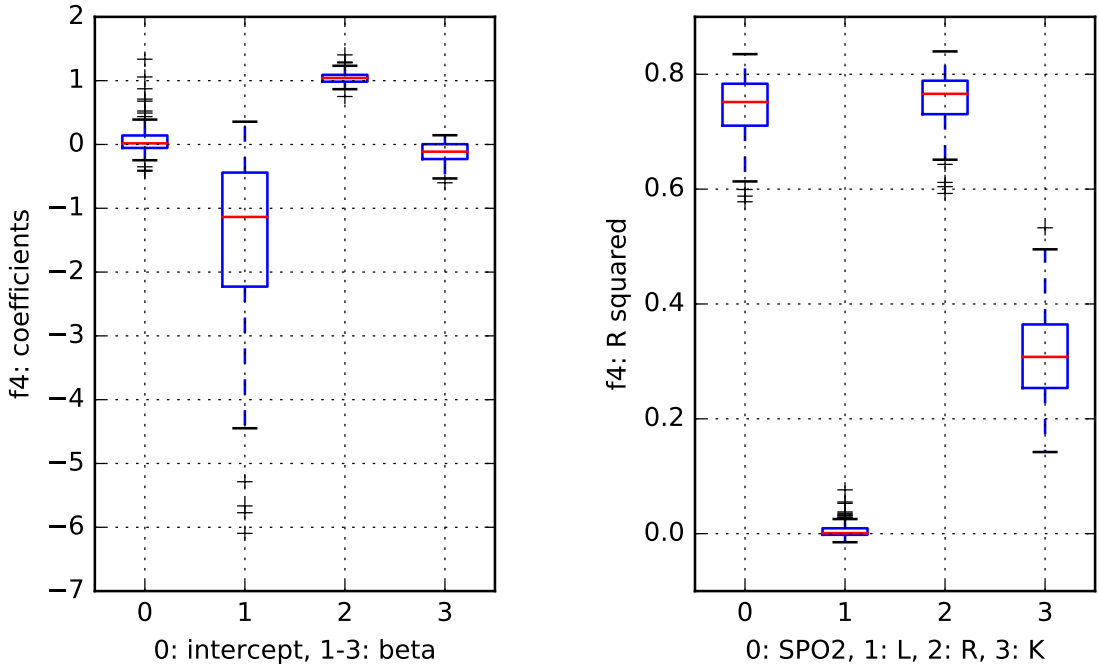

Figure 7: Results from the combined function, f_4. *Left:* Boxplots showing the β coefficients of the level-1 model. *Right:* Boxplots showing the R^2 values.

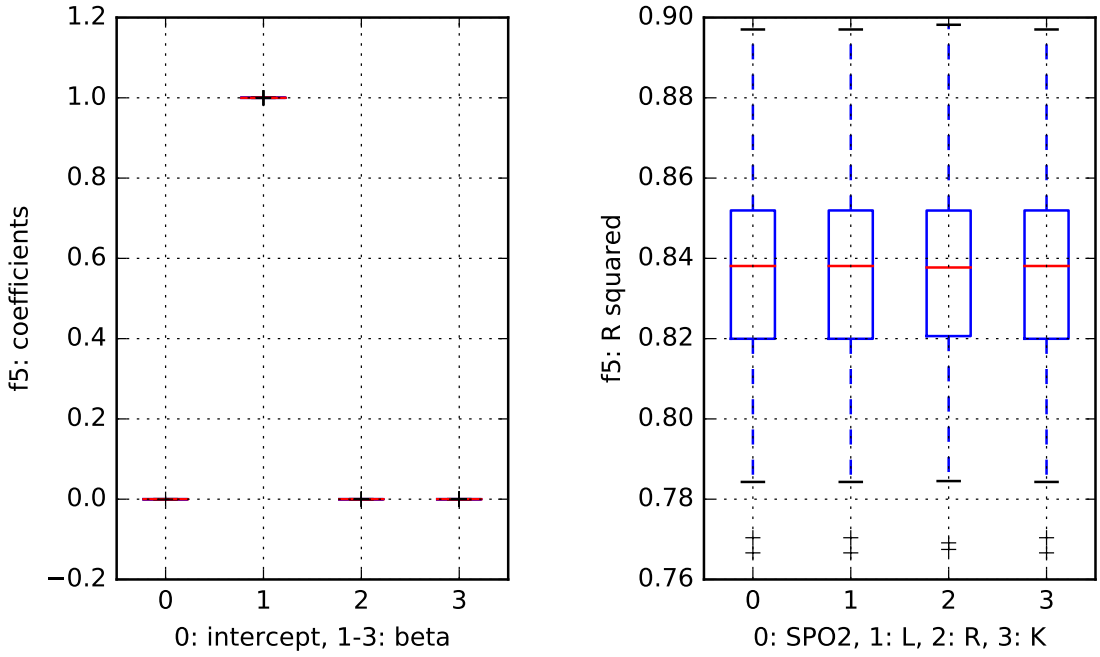

Figure 8: Results from the linear function, f_5. *Left:* Boxplots showing the β coefficients of the level-1 model. *Right:* Boxplots showing the R^2 values.

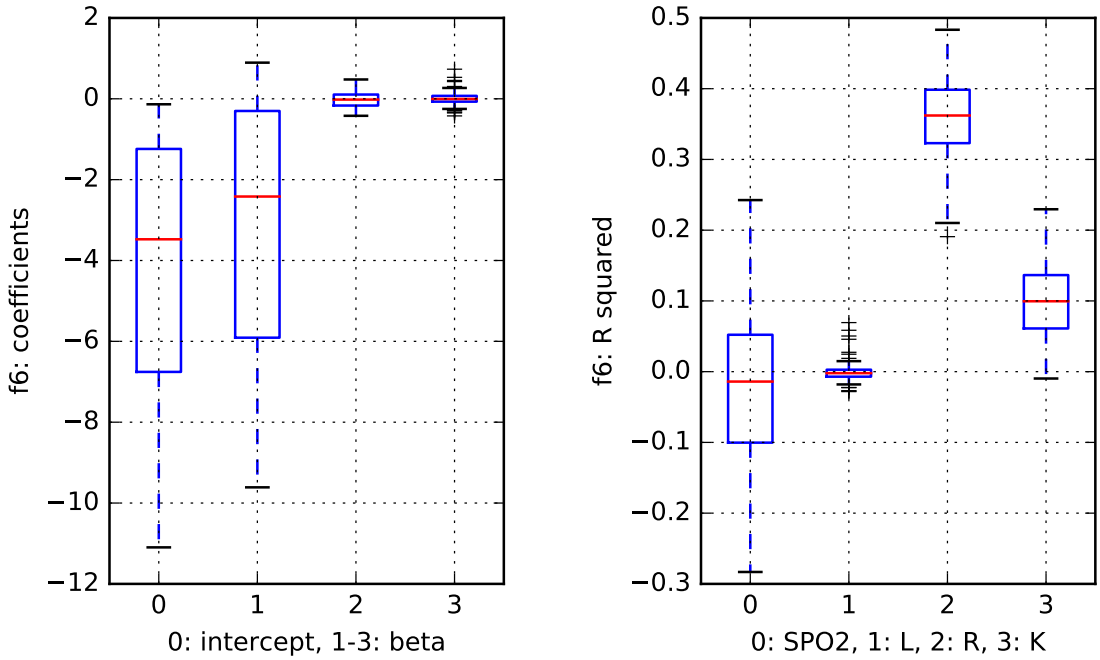

Figure 9: Results from the noise function, f_6. *Left:* Boxplots showing the β coefficients of the level-1 model. *Right:* Boxplots showing the R^2 values.

References

[1] Alessandro Baldi Antognini and Maroussa Zagoraiou. Exact optimal designs for computer experiments via Kriging metamodelling. *Journal of Statistical Planning and Inference*, 140(9):2607–2617, September 2010.

[2] Thomas Bartz-Beielstein, Christian Lasarczyk, and Mike Preuß. Sequential Parameter Optimization. In B McKay et al., editors, *Proceedings 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland*, pages 773–780, Piscataway NJ, 2005. IEEE Press.

[3] Thomas Bartz-Beielstein, Konstantinos E Parsopoulos, and Michael N Vrahatis. Design and analysis of optimization algorithms using computational statistics. *Applied Numerical Analysis and Computational Mathematics (ANACM)*, 1(2):413–433, 2004.

[4] Andrew J Booker, J E Dennis Jr, Paul D Frank, David B Serafini, and Virginia Torczon. Optimization Using Surrogate Objectives on a Helicopter Test Example. In *Computational Methods for Optimal Design and Control*, pages 49–58. Birkhäuser Boston, Boston, MA, 1998.

[5] G E P Box and N R Draper. *Empirical Model Building and Response Surfaces*. Wiley, New York NY, 1987.

[6] L Breiman, J H Friedman, R A Olshen, and C J Stone. *Classification and Regression Trees*. Wadsworth, Monterey CA, 1984.

[7] D Büche, N N Schraudolph, and P Koumoutsakos. Accelerating Evolutionary Algorithms With Gaussian Process Fitness Function Models. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, 35(2):183–194, May 2005.

[8] Ivo Couckuyt, Filip De Turck, Tom Dhaene, and Dirk Gorissen. Automatic surrogate model type selection during the optimization of expensive black-box problems. In *2011 Winter Simulation Conference - (WSC 2011)*, pages 4269–4279. IEEE, 2011.

[9] Michael Emmerich, Alexios Giotis, Mutlu özdemir, Thomas Bäck, and Kyriakos Giannakoglou. Metamodel-assisted evolution strategies. In J J Merelo Guervós, P Adamidis, H G Beyer, J L Fernández-Villacañas, and H P Schwefel, editors, *Parallel Problem Solving from Nature—PPSN VII, Proceedings~Seventh International Conference, Granada*, pages 361–370, Berlin, Heidelberg, New York, 2002. Springer.

[10] Alexander Forrester, András Sóbester, and Andy Keane. *Engineering Design via Surrogate Modelling*. Wiley, 2008.

[11] Alexander I J Forrester and Andy J Keane. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45(1-3):50–79, January 2009.

[12] K B Wilson G E P Box. On the Experimental Attainment of Optimum Conditions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 13(1):1–45, 1951.

[13] K C Giannakoglou. Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. *Progress in Aerospace Sciences*, 38(1):43–76, January 2002.

[14] Tushar Goel, Raphael T Haftka, Wei Shyy, and Nestor V Queipo. Ensemble of surrogates. *Struct. Multidisc. Optim.*, 33(3):199–216, September 2006.

[15] Robert B Gramacy. tgp: An R Package for Bayesian Nonstationary, Semiparametric Nonlinear Regression and Design by Treed Gaussian Process Models. *Journal of Statistical Software*, 19(9):1–46, June 2007.

[16] P Hajela and E Lee. Topological optimization of rotorcraft subfloor structures for crashworthiness considerations. *Computers & Structures*, 64(1-4):65–76, July 1997.

[17] Trevor Hastie. *The elements of statistical learning : data mining, inference, and prediction*. Springer, New York, 2nd ed. edition, 2009.

[18] Mark Hauschild and Martin Pelikan. An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation*, 1(3):111–128, September 2011.

[19] Jiaqiao Hu, Yongqiang Wang, Enlu Zhou, Michael C Fu, and Steven I Marcus. A Survey of Some Model-Based Methods for Global Optimization. In Daniel Hernández-Hernández and J Adolfo Minjárez-Sosa, editors, *Optimization, Control, and Applications of Stochastic Systems*, pages 157–179. Birkhäuser Boston, Boston, 2012.

[20] Edward Huang, Jie Xu, Si Zhang, and Chun Hung Chen. Multi-fidelity Model Integration for Engineering Design. *Procedia Computer Science*, 44:336–344, 2015.

[21] Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. An Evaluation of Sequential Model-based Optimization for Expensive Blackbox Functions. In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 1209–1216, New York, NY, USA, 2013. ACM.

[22] R Jin, W Chen, and T W Simpson. Comparative studies of metamodelling techniques under multiple modelling criteria. *Struct. Multidisc. Optim.*, 23(1):1–13, December 2001.

[23] Y Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12, October 2003.

[24] Y Jin, M Olhofer, and B Sendhoff. On Evolutionary Optimization with Approximate Fitness Functions. *GECCO*, 2000.

[25] D R Jones, M Schonlau, and W J Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13:455–492, 1998.

[26] Jack P C Kleijnen. Kriging metamodeling in simulation: A review. *European Journal of Operational Research*, 192(3):707–716, February 2009.

[27] P Larraaga and J A Lozano. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer, Boston MA, 2002.

[28] Minh Nghia Le, M N Le, Yew Soon Ong, Y S Ong, S Menzel, Stefan Menzel, Yaochu Jin, Y Jin, B Sendhoff, and Bernhard Sendhoff. Evolution by adapting surrogates. *Evolutionary Computation*, 21(2):313–340, 2013.

[29] S N Lophaven, H B Nielsen, and J Søndergaard. DACE—A Matlab Kriging Toolbox. Technical report, 2002.

[30] J Mockus, V Tiesis, and A Zilinskas. Bayesian Methods for Seeking the Extremum. In L C W Dixon and G P Szegö, editors, *Towards Global Optimization*, pages 117–129. Amsterdam, 1978.

[31] D C Montgomery. *Design and Analysis of Experiments*. Wiley, New York NY, 5th edition, 2001.

[32] K P Murphy. Machine learning: a probabilistic perspective, 2012.

[33] Andrea Nelson, Juan Alonso, and Thomas Pulliam. Multi-Fidelity Aerodynamic Optimization Using Treed Meta-Models. In *Fluid Dynamics and Co-located Conferences*. American Institute of Aeronautics and Astronautics, Reston, Virigina, June 2007.

[34] Emanuele Olivetti. blend.py, 2012. Code available on https://github.com/emanuele/kaggle_pbr/blob/master/blend.py. Published under a BSD 3 license.

[35] MJD Powell. Radial Basis Functions. *Algorithms for Approximation*, 1987.

[36] Mike Preuss. *Multimodal Optimization by Means of Evolutionary Algorithms*. Natural Computing Series. Springer International Publishing, Cham, 2015.

[37] F Pukelsheim. *Optimal Design of Experiments*. Wiley, New York NY, 1993.

[38] Nestor V Queipo, Raphael T Haftka, Wei Shyy, Tushar Goel, Rajkumar Vaidyanathan, and P Kevin Tucker. Surrogate-based analysis and optimization. *Progress in Aerospace Sciences*, 41(1):1–28, January 2005.

[39] Alain Ratle. Parallel Problem Solving from Nature — PPSN V: 5th International Conference Amsterdam, The Netherlands September 27–30, 1998 Proceedings. pages 87–96. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[40] Margarita Alejandra Rebolledo Coy, Sebastian Krey, Thomas Bartz-Beielstein, Oliver Flasch, Andreas Fischbach, and Jörg Stork. Modeling and Optimization of a Robust Gas Sensor. Technical Report 03/2016, Cologne Open Science, Cologne, 2016.

[41] T J Santner, B J Williams, and W I Notz. *The Design and Analysis of Computer Experiments*. Springer, Berlin, Heidelberg, New York, 2003.

[42] M Schonlau. *Computer Experiments and Global Optimization*. PhD thesis, University of Waterloo, Ontario, Canada, 1997.

[43] L Shi and K Rasheed. A Survey of Fitness Approximation Methods Applied in Evolutionary Algorithms. In *Computational Intelligence in Expensive Optimization Problems*, pages 3–28. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[44] G Sun, G Li, S Zhou, W Xu, X Yang, and Q Li. Multi-fidelity optimization for sheet metal forming process. *Structural and Multidisciplinary ...*, 2011.

[45] M J van der Laan and E C Polley. *Super Learner in Prediction*. UC Berkeley Division of Biostatistics Working Paper ..., 2010.

[46] V N Vapnik. Statistical learning theory. Wiley, 1998.

[47] G Gary Wang and S Shan. Review of Metamodeling Techniques in Support of Engineering Design Optimization. *Journal of Mechanical ...*, 129(4):370–380, 2007.

[48] David H Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, January 1992.

[49] Luis E Zerpa, Nestor V Queipo, Salvador Pintos, and Jean-Louis Salager. An optimization methodology of alkaline–surfactant–polymer flooding processes using field scale numerical simulation and multiple surrogates. *Journal of Petroleum Science ...*, 47(3-4):197–208, June 2005.

[50] Z Zhou, Y S Ong, P B Nair, A J Keane, and K Y Lum. Combining Global and Local Surrogate Models to Accelerate Evolutionary Optimization. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, 37(1):66–76, 2007.

[51] Mark Zlochin, Mauro Birattari, Nicolas Meuleau, and Marco Dorigo. Model-Based Search for Combinatorial Optimization: A Critical Survey. *Annals of Operations Research*, 131(1-4):373–395, 2004.

[52] J M Zurada. Analog implementation of neural networks. *IEEE Circuits and Devices Magazine*, 8(5):36–41, 1992.

CIplus
Band 5/2016

Stacked Generalization of Surrogate Models - A Practical Approach

Technischer Bericht /
Technical Report

Prof. Dr. Thomas Bartz-Beielstein
Institut für Informatik
Fakultät für Informatik und Ingenieurwissenschaften
Technische Hochschule Köln

Mai 2016

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Technology
Arts Sciences
TH Köln