Preface

Computational systems inspired by nature are often analyzed experimentally and principled methods can make this analysis reliable and effective. Severe requirements have been transmitted to draw objective conclusions from computational experiments. At the same time profitable ways of looking into the data are used to improve the design and configuration of the computational systems. The quest for these methods is spawning a considerable amount of research integrating and expanding existing work in the field of statistics.

The contributions to the workshop mainly exploit an interesting and important link between optimization of stochastic algorithms and model-based analysis in simulation and production engineering. Several issues within the model-based approach require to be understood and adapted to the particular context. Model validation and data transformation are two of them that received particular attention at the workshop.

The first contribution, written by *Tobias Wagner*, considers an interesting and important link between model-based analysis and optimization of stochastic algorithms and the experiment-based optimization of processes in production engineering. *Bernd Bischl*, *Olaf Mersmann*, and *Heike Trautmann* present and compare resampling strategies as methods for model validation. *Simon Wessing* and *Tobias Wagner* demonstrate that a rank transformation of the data before the modeling and aggregation improves the mean performance of the sequential parameter optimization (SPO). An example from pratice concludes these proceedings: *Patrick Koch*, *Wolfgang Konen*, *Oliver Flasch*, and *Thomas Bartz-Beielstein* present how parameter tuning can be applied in the context of stormwater prediction.

We wish to thank all authors and reviewers for their contributions and fruitful discussions. We hope that these proceedings constitute a further small step forward in the setting up of a methodology and related software.

September 2010

The organizing committee
WEMACS 2010

Organization

WEMACS 2010 was organized jointly to the International Conference on Parallel Problem Solving From Nature (PPSN 2010)

Organizing Committee

Thomas Bartz-Beielstein, Cologne University of Applied Sciences
Marco Chiarandini, University of Southern Denmark
Luís Paquete, University of Coimbra, Portugal
Mike Preuss, TU Dortmund, Germany

Advisory Board

Mauro Birattari, Université Libre de Bruxelles
Juergen Branke, University of Warwick
Dimo Brockhoff, INRIA Saclay Île-de-France
Gusz Eiben, Vrije Universiteit Amsterdam
Carlos Fonseca, Universidade do Algarve
David Ginsbourger, Universität Bern
Yuri Goegebeur, Syddansk Universitet
Wolfgang Konen, Cologne University of Applied Sciences
Jörn Mehnen, Cranfield University
Boris Naujoks, TU Dortmund
Ruxandra Stoean, Universitatea din Craiova
Heike Trautmann, TU Dortmund

Table of Contents

A Subjective Review of the State of the Art in Model-Based Parameter Tuning

Tobias Wagner

Institute of Machining Technology (ISF), TU Dortmund
Baroper Straße 301, 44227 Dortmund, Germany
wagner@isf.de, http://www.isf.de

Abstract. Over recent years, the model-based parameter tuning of computational systems has become an emergent research topic. When the considered systems are subject of stochastic effects, the parameter tuning can be seen as a noisy optimization task, as often faced in production engineering. In this paper, the current research on parameter optimization is discussed based on experiences from an engineering background. An extended framework of important steps in a model-based parameter-tuning procedure is proposed which enhances known frameworks with data transformation and model validation steps. For each step, references to important literature are provided and topics for future research are identified.

Keywords: Design and Analysis of Computer Experiments (DACE), Expected Improvement (EI), Sequential Parameter Optimization (SPO)

1 Introduction

The model-based analysis and optimization of computational systems which are subject to stochastic (uncontrollable) effects, i. e. noise, is closely related to the experiment-based optimization of processes in production engineering. In both scenarios, the objective function, e. g., the response of the system to a specific design, cannot be determined exactly. Thus, certainty about the true quality of a design can only be obtained by repeating the corresponding experiment and performing an estimation of the distribution of the responses. As another important factor, an efficient use of experiments is necessary since each experiment is expensive. Consequently, sequential model-based approaches which use the information of previous experiments for the determination of new design points are popular in both disciplines.

Nevertheless, important differences between model-based parameter optimization of computational systems and production-engineering processes exist. While the main cost factor of computer experiments is the computational runtime, real experiments in production engineering also produce costs for the machines, the tools, and the operator. The increase of computational power and the use of parallelization allow high number of runs for each design to be conducted when analyzing computational systems. In contrast, only a very small number of

validation experiments is possible for production-engineering problems. Consequently, the focus of the research on parameter optimization for computational systems in the last years has been on effective, but robust schemes for identifying the current best design [15, 17, 18]. Unfortunately, the application of these approaches for applications in production engineering is not possible since the experimental budget N ist strictly limited, i. e., $N < 50$ in most cases. The main aim in this scenario is the detection and validation of a single design in the vicinity of the optimum. Therefore, more effort is spent on the effective generation of accurate surrogate models [35, 37], on the validation of these models [6], and on specific criteria for the model-based sequential optimization [25, 26, 36].

In this paper, the findings in both application areas, computational systems and production engineering, are brought together in an extended framework for model-based parameter optimization. The steps of the framework are reviewed and open research questions are discussed. Before the extended framework is proposed, the terms, definitions, and methods used therein are introduced. In this context, also references to important literature are provided. Based on these basics, the framework is presented and discussed. Conclusions are drawn and an outlook on further research is provided.

2 Basics

The initial situation for parameter tuning turns out to be a classical optimization problem $\min_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x})$.[1] $\mathbf{X}$ is called space of allowable parameter settings [15] or design space. Consequently, a vector $\mathbf{x} \in \mathbf{X}$ is denoted as parameter setting or design. It contains specific settings for each tuning parameter $x_i,\ i = 1, \ldots, n$. The design space $\mathbf{X}$ is often bounded by box constraints $l_i \leq x_i \leq u_i$ which define the region of interest (bounds) $\mathbf{B} = \begin{pmatrix} l_1 & \cdots & l_n \\ u_1 & \cdots & u_n \end{pmatrix} = \begin{pmatrix} \mathbf{l} \\ \mathbf{u} \end{pmatrix}$. The corresponding value of the objective function $y(\mathbf{x})$ is evaluated by conducting experiments on a set of test instances $\mathcal{T}$. In the frequently analyzed case of parameter tuning for optimization algorithms, $\mathcal{T}$ comprises a set of test functions and a target quality or a runtime budget. In machining experiments, $\mathcal{T}$ can be seen as a specification of the geometry to be machined and the tool to be used. Based on the outcome of the experiment, a response y, e. g., runtime or final solution quality of a computational system or the material volume that can be machined before the tool is worn, can be derived. Based on stochastic effects in the algorithm or problem instance and uncontrollable environmental effects during the process, the actual objective value of a design can only be evaluated as $y = y(\mathbf{x}) + \epsilon$, where ϵ can be denoted as random error or noise of an evaluation.

An empirical surrogate model $\mathcal{M}$ is based on the available sets of designs $\mathbf{D} = (\mathbf{x}_1^T, \ldots, \mathbf{x}_k^T)^T$ and responses $\mathbf{y} = (y_1, \ldots, y_k)^T$ and approximates the actual objective function $y(\mathbf{x})$. In this paper, the discussion only focuses on surrogate models as used in the Design and Analysis of Computer Experiments

[1] Minimization problems are considered in this paper. Maximization problems can be transformed to corresponding minimization problems $\min_{\mathbf{x}\in\mathbf{X}} -f(x)$.

(DACE) [28, 29]. These models consider each of the k responses as produced by the model $y_j = \mathbf{f}(\mathbf{x}_j)^T\beta + Z(\mathbf{x}_j),\ j = 1, \ldots, k$, where $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_p(\mathbf{x}))^T$ are monomial regression functions, e. g., x_1, x_2^2, or $\sin(x_3)$, $\beta = (\beta_1, \ldots, \beta_p)^T$ is p-dimensional vector of corresponding regression coefficients, and $\mathrm{Z}(\mathbf{x})$ is a zero-mean (centered) stationary Gaussian process (GP) with dependencies specified by the covariance $Cov\{Z(\mathbf{x}_{j_1}), Z(\mathbf{x}_{j_2})\} = \sigma_Z^2 r(\mathbf{x}_{j_1}, \mathbf{x}_{j_2})$ for a known correlation function r and a process variance σ_Z^2. It can be shown [29] that the best predictor $\hat{y}(\mathbf{x})$ for a design $\mathbf{x}$ with respect to the mean squared prediction error is

$$\hat{y}(\mathbf{x}) = \underbrace{\mathbf{f}(\mathbf{x})^T\hat{\beta}}_{\text{regression function}} + \mathbf{r}(\mathbf{x})^T\mathbf{R}^{-1} \underbrace{(\mathbf{y} - \mathbf{F}\hat{\beta})}_{\text{observed residuals}}, \tag{1}$$

where $\hat{\beta} = \left(\mathbf{F}^T\mathbf{R}^{-1}\mathbf{F}\right)^{-1}\mathbf{F}^T\mathbf{R}^{-1}\mathbf{y}$ are the regression coefficients estimated in the sense of least squares and $\mathbf{F} = (\mathbf{f}(\mathbf{x}_1)\cdots\mathbf{f}(\mathbf{x}_k))^T$ is the $k \times p$ matrix of regression function values for each of the k designs. Analogously, the vector $\mathbf{r}(\mathbf{x}) = (r(\mathbf{x}, \mathbf{x}_1), \ldots, r(\mathbf{x}, \mathbf{x}_k))^T$ of correlations between $\mathbf{x}$ and the already evaluated designs and the $k \times k$ intercorrelation matrix $\mathbf{R} = (\mathbf{r}(\mathbf{x}_1)\cdots\mathbf{r}(\mathbf{x}_k))$ are defined in terms of the correlation function r. From the definition in equation (1), it can be seen that the prediction of the model is improved by an estimation of the corresponding residual to the regression function based on a linear combination of the residuals already observed, where the weight of each residual depends on the correlation with the evaluated design $\mathbf{x}$. Usually, a constant regression function $f(\mathbf{x}) = \beta_1$ and the most general correlation model

$$r(\mathbf{x}_{j_1}, \mathbf{x}_{j_2}) = \exp\left(-\sum_{i=1}^{n}\theta_i|x_{j_1,i} - x_{j_2,i}|^{p_i}\right) \tag{2}$$

of Sacks et al. [28] are used. The model parameters θ and $\mathbf{p}$ control the slope and the smoothness of the correlation function and allow a wide range of functional relations to be modeled. Thus, no assumptions on the underlying process are postulated a-priori. In many parameter tuning applications [2, 15], the exponents are fixed to $p_i = 2$, leading to an infinite times differentiable model $\mathcal{M}$.

Based on the strength of the correlations $\mathbf{r}(\mathbf{x})$ and the process variance σ_Z^2, also the corresponding uncertainty

$$\hat{s}(\mathbf{x}) = \sqrt{\sigma_Z^2\left(1 - \mathbf{r}(\mathbf{x})^T\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}) + \frac{(1 - \mathbf{1}^T\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}))^2}{\mathbf{1}^T\mathbf{R}^{-1}\mathbf{1}}\right)} \tag{3}$$

of a prediction $\hat{y}(\mathbf{x})$ can be computed.[2] The stochastic output of the experiments can be considered by means of the so-called nugget effect [14, 30]. The nugget c_{nugget} scales down the correlation function r by multiplying it with $(1 - c_{\text{nugget}})$ and thereby avoids the exact interpolation of the observations. The influence of outliers is relaxed and a smoother model can be computed. Moreover, the

[2] The formula is shown for the constant regression function $f(\mathbf{x}) = \beta_1$. Nevertheless, the uncertainty $\hat{s}(\mathbf{x})$ can also be computed for more complex regression functions.

Algorithm 1: Extended framework for model-based parameter optimization

Require: $\mathcal{T}$ {set of test instances}
$\mathbf{B}$ {region of interest (box constraints)}
N {experimental budget}
N_{init} {size of the initial design set}
r_{init} {initial number of runs per design}

1: $\mathbf{D} = generateInitialDesign(\mathbf{B}, N_{\text{init}})$ {choose initial design}
2: $\mathbf{Y} = runDesign(\mathbf{D}, r_{\text{init}})$ {perform experiments}
3: **while** $entries(\mathbf{Y}) < N$ **do**
4: $\widetilde{\mathbf{Y}} = transformLocal(\mathbf{Y})$ {transformation of the responses}
5: $[\mathbf{y}, \mathbf{s}] = aggregateRuns(\widetilde{\mathbf{Y}})$ {calculate performance and corresponding std.}
6: $[\widetilde{\mathbf{y}}, \widetilde{\mathbf{s}}] = transformGlobal(\mathbf{y}, \mathbf{s})$ {transformation of the performance}
7: $\mathcal{M} = fitModel(\mathbf{D}, \widetilde{\mathbf{y}}, \widetilde{\mathbf{s}})$ {fit surrogate model of the response}
8: $\mathcal{Q} = validateModel(\mathcal{M}, \mathbf{D}, \widetilde{\mathbf{y}}, \widetilde{\mathbf{s}})$ {validate surrogate model of the response}
9: $[\mathbf{x}^*, y^*] = defineCurrentBest(\mathbf{y}, \mathbf{s}, \mathbf{Y}, \mathcal{M})$ {decide on output design}
10: $\mathbf{x}_{\text{new}} = modelOptimization(\mathcal{M}, y^*)$ {find promising design points}
11: $r = adjustRunsPerDesign()$ {determine number of runs for new design}
12: $\mathbf{D} = concatenate(\mathbf{D}, \mathbf{x}_{\text{new}})$ {add new design point}
13: $\mathbf{Y} = concatenate(\mathbf{Y}, runDesign(\mathbf{x}_{\text{new}}, r))$ {evaluate new design point}
14: **end while**
15: **return** $\mathcal{M}$, $\mathbf{x}^*$, y^* {return model, best design point, and its performance}

uncertainty of already evaluated designs is no longer zero since the weighted correlation of a solution with itself $(1 - c_{\text{nugget}})r(\mathbf{x}, \mathbf{x})$ is smaller than one. This effect can be used to integrate the uncertainty of an evaluation into the model. The single factor c_{nugget} is then replaced by a vector of nugget factors for each observation [24].[3]

3 Discussion

The extended framework for model-based parameter optimization is shown in Algorithm 7. In the following, current realizations of the steps in the framework are summarized and discussed. Implementations of some of these realizations are available in the Sequential Parameter Optimization Toolbox (SPOT) [1, 3].

3.1 Pre-experimental planning and the choice of input parameters

The set of test instances $\mathcal{T}$ and the region of interest $\mathbf{B}$ have already been introduced in the previous section. Whereas a meaningful selection of test instances and runtime budgets is a crucial point in the assessment of optimization algorithms [16, 34], the surrounding conditions of a specific process of interest are

[3] The formulas of Picheny et al. [13, 24] are based on covariances in spite of correlations.

usually known. More specific, the main interest in the former is the design of the system and its assessment versus other competing systems. Therefore, a good parameterization is necessary for the competitiveness of the algorithm. In the latter, the comparison between different processes is rarely of interest. The aim of the model-based optimization is to find a suitable setting for the available machines and tools in the given experimental budget N.

In cases where no experiences about the analyzed system exist, a statistical screening using fractional factorial design should be conducted before the tuning parameters and the region of interest are defined [23]. Unfortunately, this is only rarely the case in the analysis of computational systems [27]. It has been shown that a dimensionality-reduction is important when using DACE models, whose correlation function mainly depends on the distance between designs [33]. In this context, also the use of transformations of the tuning parameters should be considered. DACE models are relying on a stationary GP, i. e., a constant activity of the response over the considered domain. However, a change of the population size from $\mu = 1$ to $\mu = 10$ surely has a bigger effect on the response compared to change from $\mu = 100$ to $\mu = 110$. The change of the magnitude of the parameter is more important than the change in the absolute value. Thus, a logarithmic transformation is suitable (see, e. g. [5]). Whenever some theoretic knowledge on suitable parameters exist, e. g., a mutation rate should be $p_m = 1/n$, where n is the dimension of the problem, this knowledge should be used for the transformation. Applied to the mutation rate, the tuning parameter p_m could be substituted by $m > 0$ with $p_m = n^{-m}$. These transformations often significantly improve the quality of the surrogate models.

3.2 The initial design

In order to compute an initial surrogate model, which can then be used for the sequential determination of new design points, a design set has to be evaluated a-priori. In this context, the following questions arise:

1. What kind of experimental design should be chosen?
2. How many different designs N_{init} should be evaluated?
3. How many replications of each design r_{init} should be performed?

Over the last years, many studies have been conducted in order to provide answers for these questions [4, 6, 8, 15, 16, 21, 22, 29]. Unfortunately, the amount of significant results is rather low. Koehler and Owen provide an overview of theoretical results for many different types of designs. However, these results do not provide direct recommendations for the choice of the initial design. It is generally assumed that space-filling designs, such as Latin hypercube sampling (LHS) [22], are superior to factorial or central composite designs [8] for estimating a DACE model, but comparative studies [15, 22, 29] can only conclude that LHS is superior to random sampling, and even this holds only in most cases. Nevertheless, LHS are used in almost all popular DACE-based approaches [2, 20].

Since LHS is a very general framework, different optimization criteria for Latin hypercube designs exist [19]. The entropy criterion $e(\mathbf{D}) = \det \mathbf{R}$ is directly

related to the information for the correlation matrix $\mathbf{R}$ obtained by the design. It can be computed based on fixed model parameters θ and $\mathbf{p}$. For an isotropic model $\theta_1 = \ldots = \theta_n$ and $p_1 = \ldots = p_n$, the entropy criterion results in a design optimizing the minmax-criterion. If some evidence on the importance of the different tuning parameters exist, e. g., by conducting a parameter screening before the model-based optimization, these importances can also be considered by adjusting the values of θ_i, whereby a higher θ results in a higher importance of the corresponding tuning parameter.

With respect to the second question, the recommendation $N_{\text{init}} \approx 10n$ was established [20]. It has been shown that this number produces good predictions also for noisy responses [6]. More recent studies have shown that even $N_{\text{init}} < 10n$ can produce good results in a sequential optimization given that the modeled response is not too bumpy [4, 16]. A lower bound for the size of the initial design should be the number of model parameters, i. e., $N_{\text{init}} \geq 2n + 1$ for a model with the power exponential correlation kernel (equation 2) and a nugget factor c_{nugget}. A solution to the problem of local overfitting for small initial designs [16] is offered by an adaptive choice of the criterion for determining the next design. This is discussed in subsection 3.6. For the initial number of runs of each design r_{init}, a low number $r_{\text{init}} \leq 3$ is recommended [4]. In particular, some studies indicate that the budget for the initial design should be completely allocated to exploration, i. e., $r_{\text{init}} = 1$ [6, 13].

3.3 Transformations of the response

In classical statistics, transformations are common means to adjust the experimental data in order to obtain a better agreement with the assumptions of the modeling methods, e.;g., normality, linearity, or stationary variance. For DACE, in particular the logarithmic transformation is used for improving the fit of the model [15, 18, 20]. The use of this transformation is motivated by better results after a complete sequential optimization. Despite the applicability of the logarithmic transformation seeming to hold for many types of parameter tuning problems [15], a closer look at the actual benefit of the transformation would increase understanding.

In classical approaches, the data of all runs, independently of having the same design $\mathbf{x}$, are used to fit the surrogate model [14, 20]. Consequently, only one transformation can be performed. It has recently been shown that an a-priori aggregation of the runs and a transformation of the aggregated responses allows more performance indices to be realized and models to be computed efficiently [15]. However, with regard to the assumptions behind the aggregation and behind the surrogate model, different targets have to be accomplished by the transformation. Consequently, two different transformation steps are distinguished in the extended framework.

In the first step, the data over different runs of a parameter setting has to be aggregated into a response or performance value. When a parametric index such as the mean of the runs is desired, an approximately gaussian distribution of results is required. This is especially important when the uncertainty of the

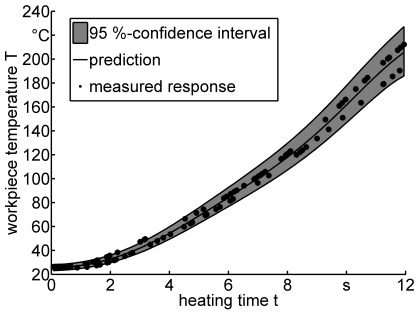

Fig. 1: Incorporation of a mean-variance-relationship by using a homoscedastic model on a logarithmic transformation of the response.

model (equation 3) is directly related to the uncertainty of the response [13, 14, 24]. To accomplish this precondition, a power transform (also known as Box-Cox technique) [7] or rank transform [9] can be used.

In the second step, the aggregated response can be transformed again in order to improve the prediction quality of the surrogate model. As mentioned before, the logarithmic transformation has shown to be suitable in the context of parameter tuning of optimization algorithms. This is not only due to an improved prediction [15], but also due to the usually existing mean-variance-relationship, i. e., a positive correlation of the aggregated response y and the corresponding measurement noise ϵ [5]. Based on a logarithmic transformation, this effect can be modelled using homoscedastic DACE models with a constant nugget c_{nugget}. An example of a homoscedastic model of an induction heating process using a logarithmic transformation, which is computed based on three independent thermocouple measurements, is shown in Fig. 1 [35].

3.4 Fitting and validating the model

The determination of the parameters θ and $\mathbf{p}$ in equation 2 is usually termed model fitting. To accomplish this, maximum likelihood estimation (MLE), i. e., the optimization of the term $\sigma_Z^2 \det \mathbf{R}$ has become the state of the art. If non-interpolating models with strictly positive uncertainties for already evaluated designs are desired, also c_{nugget} can be determined by MLE [14]. Compared to an evaluation based on resampling strategies, such as leave-one-out cross-validation (CV) [20, 32], it is more efficient and has shown to produce better models in most of my engineering applications. However, MLE can provide bad estimations for small sample sizes [12] and noisy data. In former studies, it has been shown that DACE models allowing $p_i \leq 2$ and using only a constant regression term $f(\mathbf{x}) = \beta_1$ are often superior to models using a quadratic regression function and $p_i = 2$ [6].

Even when some authors state that noise-free DACE models can be used for the modeling of the aggregated responses [2, 15], it has to be noted that the

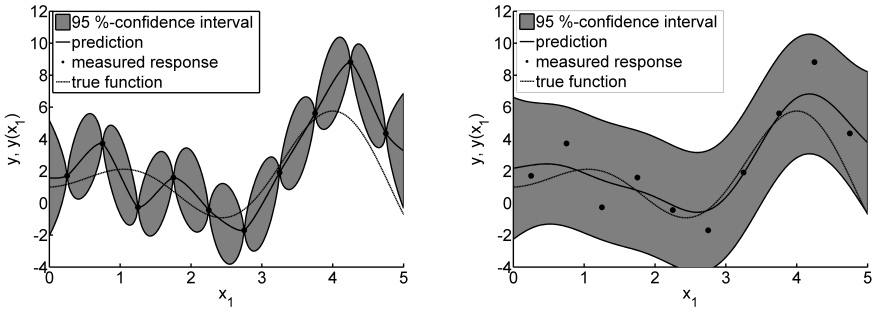

Fig. 2: Comparison of the predictions of interpolating (left) and non-interpolating (right) DACE models.

aggregated response is still subject to error in the estimation; in particular when the performance index is computed based on a small sample.[4] This has to be considered when choosing the modeling approach. In Fig. 2, a one-dimensional test functions is modeled based on 10 designs with corresponding noisy responses. Whereas the interpolating model clearly shows overfitting and an overestimation of the activity θ, the non-interpolating model provides predictions much closer to the true function. Moreover, the estimation of the uncertainty reflects the variation in the estimation of the true response.

In contrast to production engineering, the surrogate models used for the sequential determination of designs in parameter tuning are usually not validated – though this has been recommended from the beginning [20]. Based on a CV of the surrogate model, the prediction quality of the model can be evaluated, and if required, transformations of the response can be performed or further exploratory designs can be added. The validation of the model is particularly important when the model is used to determine the current best solution [14]. In the example shown in Fig. 2, the coefficient of determination is $R^2_{\text{CV}} = 0.26$ for the interpolating model and $R^2_{\text{CV}} = 0.46$ for the non-interpolating model.[5] Thus, the validation agrees with the true quality of the model. Empirical studies have shown that a good fit in the cross-validation usually ensures a good prediction quality [6, 32]. Nevertheless, it has to be noted that a validation based on CV is only suitable when the model parameters were not fit using CV.

3.5 Who is the current best?

After all designs have been evaluated, usually two decisions have to be made. The first one is about validation experiments of the currently best design, the

[4] Given a random sample of size n with independent observations, the standard error of the estimated arithmetic mean is σ_ϵ/n, where σ_ϵ is the standard deviation of the different runs of a design. For the estimation of the standard error of nonparametric aggregated responses, resampling strategies like bootstrapping [10] can be used [5].

[5] R^2_{CV} is determined based on CV since $R^2 = 1$ for interpolating DACE models.

second one about the evaluation of new design points. Based on recent studies [15, 18, 24], it is accepted that the first decision is made based on the available evaluations, whereas the second one is made based on an optimization of the surrogate model. In all these approaches, the objective of both decisions is the identification of the solution being the minimizer of the considered aggregation. However, when comparing different algorithms, algorithm A is accepted as superior to algorithm B only when the difference between the result distributions of both algorithms is significant. In order to significantly detect a difference δ in the means which is equal to the standard deviation σ_ϵ of the results based on a paired t-test, at least $r = 50$ runs of each algorithm are necessary [11]. Consequently, a significant discrimination of the actually best design for a stochastic optimization algorithm is almost impossible based on the available runtime budget. With respect to the efficiency of the parameter tuning procedure, it may be sufficient to find a design that is not significantly worse to the unknown optimal design based on a reasonable number of runs $r \approx 30$, i. e., $\delta = 1.5\sigma_\epsilon$. Nevertheless, the best design $\mathbf{x}^*$ returned should be validated with the highest number of runs over all evaluated designs [15].

3.6 Sequential design – The role of the infill criterion

In the phase of the sequential design, both informations provided by the model, the prediction of the aggregated response and the corresponding uncertainty, are used to determine new designs for evaluation. For a model-based internal optimization, the scalarization of these information into an optimization criterion, the so-called infill criterion [30], is required. Sequential Parameter Optimization (SPO) [2] uses a generalized form of the expected improvement (EI) criterion [31]

$$EI(\mathbf{x}) = (y^* - \hat{y}(\mathbf{x}))\Phi(u(\mathbf{x})) + \hat{s}(\mathbf{x})\phi(u(\mathbf{x})), u(\mathbf{x}) = \frac{y^* - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\,, \tag{4}$$

which puts more emphasis on the predicted uncertainty $\hat{s}(\mathbf{x})$. In general, the EI is used for an automated balancing between local exploitation and global exploration on deterministic global optimization problems [20]. Although the EI and its variants are also considering a global exploration of the search space, they are based on a strong confidence in the predictions of $\hat{y}(\mathbf{x})$ and $\hat{s}(\mathbf{x})$. Thus, whenever the model quality is poor, e. g., due too bumpy responses and a too small initial sample, the EI can lead to a local overfitting of the model [16]. As already mentioned in subsection 3.4, a cross-validation of the model can indicate such situations, providing the chance to adaptively choose the infill criterion. In cases of poor model quality, exploratory infill criteria, such as the entropy criterion [19] or the maximization of the minimal distance to other designs can be used. The entropy criterion still relies on a reasonable estimation of the correlation parameter vectors θ and $\mathbf{p}$, where the second criterion can be used when the model is completely infeasible.

Even in cases where the model quality with respect to the predictions $\hat{y}$ is fine, say $R^2_{\mathrm{CV}} > 0.5$, many conceptual problems can occur when applied to

problems with stochastic responses. A good discussion of the roles of y^* and $\hat{y}$ has already been published [24]. In brief, the exact y^* of the already evaluated designs is not known (cf. subsection 3.5) and the true response $y(\mathbf{x})$ and the evaluated response y are not necessarily the same. Nevertheless, the role of $\hat{s}$ is even more critical. When an interpolating DACE model is used, the noise in the estimation of the response leads to an overestimation of the activity parameters, as shown in Fig. 2. This effect becomes extreme if two neighboring designs are evaluated with considerably different responses due to the stochastic effects [5]. It leads to huge confidence intervals of equal spread σ_Z, where no correlation to other solutions exist. Consequently, only the prediction $\hat{y}$ can be used to guide the optimization. For non-interpolating DACE models, two possible scenarios exist. When a homoscedastic model with a single nugget factor c_{nugget} is used, the prediction is likely to be surrounded by a confidence interval of constant width $(1 - c_{\text{nugget}})\sigma_Z$, as shown in Fig. 2 (right). By a-priori applying a logarithmic transformation, even a proportional mean-variance-relationship can be included in the model (cf. Fig.1). However, still the EI is mainly depended on the prediction $\hat{y}$. Therefore, the only variant of DACE models where both components of the EI can show a complex interaction is the heteroscedastic case with a vector of noise variances for each response, but even in this case conceptual problems exist. The noise variance is the second moment of the response. Its estimation is much harder than the one of the aggregated response itself. Since the uncertainty is still expressed in terms of the standard deviation of a normal distribution, transformations to approximate normality are required. Moreover, the EI would emphasize on high uncertainties. Usually, robust solutions with low variance are desired.

A recent approach that tackles most of the conceptual problems of the standard EI is presented by Picheny et al. [24]. They use a heteroscedastic DACE model for considering the noise variance, i. e., the accuracy, of each response. By these means, the uncertainty of the model is not longer only related to the uncertainty of the prediction, but rather to the uncertainty of evaluation.[6] By taking a β-quantile for $\beta \geq 0.5$ as aggregation criterion, a favor for low variances can be introduced. Moreover, they propose an automated method for the determination of the computation time spent on a new design $\mathbf{x}_{\text{new}}$ based on the evolution of the EI of this design with increasing accuracy of evaluation. This adaptive increasing of evaluations for promising designs is called intensification and will be topic of the following subsection.

3.7 Intensification

In current versions of SPO, the number of evaluations for new designs is increased with the iterations. First, this increase was multiplicative $r_{\text{new}} = 2r_{\text{old}}$ [2], later it was adjusted to an additive increase $r_{\text{new}} = r_{\text{old}} + 1$ [3]. Moreover, adaptive

[6] The noise variance used in the model is not equal to the variance σ_ϵ of the different runs of a design. It is related to the standard error in estimating the true aggregated response $y(\mathbf{x})$ (cf. subsection 3.4) and can be decreased by further replications of $\mathbf{x}$.

approaches for the choice of r based on statistical tests [4] and heuristics [18, 24] have been proposed. All adaptive approaches aim at a reduction of runs for new designs which early show to be inferior to the current best design.

4 Conclusion and Outlook

In this paper, the state of the art in model-based parameter tuning was critically discussed and an extended framework of steps in a sequential parameter optimization (SPO) procedure was proposed. Compared to recent SPO variants, this framework also includes steps of data transformation and model validation. For each step of the framework, a subjective classification of recent approaches was made and basic ideas for new methods were presented.

Based on the classifaction of recent approaches, topics for future research were identified. In particular, the aim of the application of an SPO-procedure has to be specified more clearly. Is it really desired to find the design $\mathbf{x}^*$ that provides the best objective value on the true, but unknown, underlying objective $y(x)$ or is it already satisfying to obtain a design that is at least competive with this design based on statistical tests on a reasonable number of runs, say $r = 30$. Based on the agreement on the aim of SPO, suitable infill criteria should be designed. The commonly used expected improvement criterion was shown to have some weaknesses when applied to noisy responses.

Acknowledgments. This paper is based on investigations of the collaborative research center SFB/TR TRR 30, which is kindly supported by the Deutsche Forschungsgemeinschaft (DFG).

References

1. Bartz-Beielstein, T.: SPOT: An R package for automatic and interactive tuning of optimization algorithms by sequential parameter optimization. ArXiv e-prints 1006(4645B) (2010), http://arxiv.org/abs/1006.4645v1
2. Bartz-Beielstein, T., Lasarczyk, C., Preuß, M.: Sequential parameter optimization. In: McKay, B., et al. (eds.) Proc. 2005 IEEE Congress on Evolutionary Computation (CEC 2005). pp. 773–780. IEEE press, US (2005)
3. Bartz-Beielstein, T., Lasarczyk, C., Preuß, M.: SPOT Sequential Parameter Optimization Toolbox (2009), http://www.gm.fh-koeln.de/imperia/md/content/personen/lehrende/bartz_beielstein_thomas/spotdoc.pdf
4. Bartz-Beielstein, T., Preuß, M.: Considerations of budget allocation for sequential parameter optimization (SPO). In: Paquete, L., et al. (eds.) Proc. Workshop on Empirical Methods for the Analysis of Algorithms (EMAA 2006). pp. 35–40 (2006)
5. Biermann, D., Joliet, R., Michelitsch, T., Wagner, T.: Sequential parameter optimization of an evolution strategy for the design of mold temperature control systems. In: Fogel, G., Ishibuchi, H., et al. (eds.) Proc. 2010 IEEE Congress on Evolutionary Computation (CEC 2010). IEEE Press, US (2010)

6. Biermann, D., Weinert, K., Wagner, T.: Model-based optimization revisited: Towards real-world processes. In: Michalewicz, Z., Reynolds, R.G. (eds.) Proc. 2008 IEEE Congress on Evolutionary Computation (CEC 2008). pp. 2980–2987. IEEE Press, US (2008)
7. Box, G.E.P., Cox, D.R.: An analysis of transformations. Royal Statistical Society, Series B 26(2), 211–252 (1964)
8. Bursztyn, D., Steinberg, D.M.: Comparison of designs for computer experiments. Statistical Planning and Inference 136(3), 1103–1119 (2006)
9. Conover, W.J., Iman, R.L.: Rank transformations as a bridge between parametric and nonparametric statistics. American Statistician 35(3), 124–129 (1981)
10. Efron, B.: Bootstrap methods: Another look at the jackknife. Annals of Statistics 7(1), 1–26 (1979)
11. Ferris, C., Grubbs, F., Weaver, C.: Operating characteristics for the common statistical tests of significance. Annals of Mathematical Statistics 17(2), 178–197 (1946)
12. Ginsbourger, D., Dupuy, D., Badea, A., Roustant, O., Carraro, L.: A note on the choice and the estimation of kriging models for the analysis of deterministic computer experiments. Applied Stochastic Models for Business and Industry 29(2), 115–131 (2009)
13. Ginsbourger, D., Picheny, V., Roustant, O., Richet, Y.: Kriging with heterogeneous nugget effect for the approximation of noisy simulators with tunable fidelity. In: Proc. Joint Meeting of the Statistical Society of Canada and the Société Francaise de Statistique (2008)
14. Huang, D., Allen, T.T., Notz, W.I., Zheng, N.: Global optimization of stochastic black-box systems via sequential kriging meta-models. Global Optimization 34(4), 441–466 (2006)
15. Hutter, F., Bartz-Beielstein, T., Hoos, H.H., Leyton-Brown, K., Murphy, K.: Sequential model-based parameter optimisation: an experimental investigation of automated and interactive approaches. In: Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuß, M. (eds.) Empirical Methods for the Analysis of Optimization Algorithms, pp. 361–411. Springer, Berlin Heidelberg (2010)
16. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Tradeoffs in the empirical evaluation of competing algorithm designs. Annals of Mathematics and Artificial Intelligence 57(3–4) (2010)
17. Hutter, F., Hoos, H.H., Leyton-Brown, K., Murphy, K.: Time-bounded sequential parameter optimization. In: Blum, C., Battiti, R. (eds.) Proc. Conf. Learning and Intelligent Optimization (LION 4). pp. 281–298. Springer, Berlin Heidelberg (2010)
18. Hutter, F., Hoos, H.H., Leyton-Brown, K., Murphy, K.P.: An experimental investigation of model-based parameter optimisation: SPO and beyond. In: Raidl, G., et al. (eds.) Proc. Genetic and Evolutionary Computation Conf. (GECCO 2009). pp. 271–278. ACM, New York, NY (2009)
19. Jin, R., Chen, W., Sudjianto, A.: An efficient algorithm for constructing optimal design of computer experiments. Statistical Planning and Inference 134(1), 268–287 (2005)
20. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. Global Optimization 13(4), 455–492 (1998)
21. Koehler, J.R., Owen, A.B.: Computer experiments. In: Ghosh, S., Rao, C.R. (eds.) Handbook of Statistics, vol. 13, pp. 261–308. Elsevier, New York, NY (1996)
22. McKay, M.D., Conover, W.J., Beckman, R.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21(1), 239–245 (1979)

23. Montgomery, D.C.: Design and Analysis of Experiments. John Wiley and Sons, New York, 4. edn. (1997)
24. Picheny, V., Ginsbourger, D., Richet, Y.: Noisy expected improvement and online computation time allocation for the optimization of simulators with tunable fidelity. In: Rodrigues, H., et al. (eds.) Proc. Conf. Engineering Optimization (EngOpt 2010) (2010)
25. Ponweiser, W., Wagner, T., Biermann, D., Vincze, M.: Multiobjective optimization on a limited amount of evaluations using model-assisted $\mathcal{S}$-metric selection. In: Rudolph, G., et al. (eds.) Proc. 10th Int'l Conf. Parallel Problem Solving from Nature (PPSN X). pp. 784–794. Springer, Berlin Heidelberg (2008)
26. Ponweiser, W., Wagner, T., Vincze, M.: Clustered multiple generalized expected improvement: A novel infill sampling criterion for surrogate models. In: Michalewicz, Z., Reynolds, R.G. (eds.) Proc. 2008 IEEE Congress on Evolutionary Computation (CEC 2008). pp. 3514–3521. IEEE Press, US (2008)
27. Ridge, E., Kudenko, D.: Screening the parameters affecting heuristic performance. In: Thierens, D., et al. (eds.) Proc. Genetic and Evolutionary Computation Conf. (GECCO 2007). p. 180. ACM, New York, NY (2007)
28. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Statistical Science 4(4), 409–435 (1989)
29. Santner, T.J., Williams, B.J., Notz, W.: The Design and Analysis of Computer Experiments. Springer, New York, NY (2003)
30. Sasena, M.J.: Flexibility and Effciency Enhancements for Constrained Global Design Optimization with Kriging Approximations. Ph.D. thesis, University of Michigan (2002)
31. Schonlau, M., Welch, W.J., Jones, D.R.: Global versus local search in constrained optimization of computer models. In: Rosenberger, W.F., Flournoy, N., Wong, W.K. (eds.) New Developments and Applications in Experimental Design, vol. 34, pp. 11–25. Institute of Mathematical Statistics, Hayward, CA (1997)
32. Tenne, Y., Armfield, S.W.: Metamodel accuracy assessment in evolutionary optimization. In: Michalewicz, Z., Reynolds, R.G. (eds.) Proc. 2008 IEEE Congress on Evolutionary Computation (CEC 2008). pp. 1505–1512. IEEE Press, US (2008)
33. Tenne, Y., Izui, K., Nishiwaki, S.: Dimensionality-reduction frameworks for computationally expensive problems. In: Fogel, G., Ishibuchi, H. (eds.) Proc. 2010 IEEE Congress on Evolutionary Computation (CEC 2010). IEEE Press, US (2010)
34. Trautmann, H., Wagner, T., Naujoks, B., Preuß, M., J.Mehnen: Statistical methods for convergence detection of multi-objective evolutionary algorithms. Evolutionary Computation 17(4), 493–509 (2009)
35. Wagner, T., Bröcker, C., Saba, N., Biermann, D., Matzenmiller, A., Steinhoff, K.: Modelling of a thermomechanically coupled forming process based on functional outputs from a finite element analysis and from experimental measurements. Advances in Statistical Analysis (AStA) 94(4) (2010)
36. Wagner, T., Emmerich, M., Deutz, A., Ponweiser, W.: On expected-improvement criteria for model-based multi-objective optimization. In: Schaefer, R., Cotta, C., Kolodziej, J., Rudolph, G. (eds.) Proc. 11th Int'l Conf. Parallel Problem Solving from Nature (PPSN XI). Springer, Berlin Heidelberg (2010)
37. Wagner, T., Passmann, D., Weinert, K., Biermann, D., Bledzki, A.K.: Efficient modeling and optimization of the property gradation of self-reinforced polypropylene sheets within a thermo-mechanical compaction process. In: Teti, R. (ed.) Proc. 6th CIRP Int'l Conf. Intelligent Computation in Manufacturing Engineering (CIRP ICME '08). pp. 447–452. Edizione Ziino, C. mare di Stabia, Italy (2008)

Resampling Methods in Model Validation

Bernd Bischl, Olaf Mersmann, and Heike Trautmann

TU Dortmund University, Dortmund, Germany
{bischl, olafm, trautmann}@statistik.tu-dortmund.de

Abstract. Meta modeling has become a crucial tool in solving expensive optimization problems. Much of the work in the past has focused on finding a good regression method to model the fitness functions. Examples of such methods include classical linear regression, splines, neural networks, kriging and support vector regression. This paper specifically draws attention to the fact that the model accuracy is a crucial aspect in the meta modeling framework. Resampling strategies such as cross-validation, subsampling, bootstrapping and nested resampling are prominent methods for model validation and are systematically discussed with respect to possible pitfalls, shortcomings and specific features.

Keywords: resampling, meta models, model validation, cross-validation, bootstrap, subsampling, nested resampling, regression

1 Introduction

Many real-world continuous optimization problems are of a complex nature which often requires expensive fitness function evaluations or computationally intensive simulations in case the functions cannot be provided explicitly in analytical form (black-box optimization). Approximations, often called meta models, are an important means to reduce the computational effort by fitting a regression function based on a number of evaluated design points, in particular when the required number of explicit fitness evaluations should be kept as small as possible.

A comprehensive overview of meta-modeling techniques with respect to evolutionary computation and multiobjective optimization in general is given in [22], [41], [25] and [31]. A detailed discussion of these methods, however, is beyond the scope of this paper. Most prominent approaches are polynomial models resp. response surface analysis [29], kriging together with the expected improvement approach (e.g. [21]), for example used for algorithm parameter tuning in the well-known framework of sequential parameter optimization (SPO, [3]), neural networks (e.g. [32]) or support vector regression (e.g. [47], [5]).

Meta modeling strategies can be implemented by constructing a global meta-model based on initial data points (offline data sampling), e.g. generated by an experimental design, and replacing the fitness function by the meta-model for the whole optimization process. However, in general the meta model should be

used in combination with the original fitness function in order to control the convergence to the meta model optimum which does not necessarily coincide with the optimum of the original fitness function. Different strategies for this so-called evolution control exist [22]. By these means the meta model as well can be updated sequentially during optimization (online data sampling). In addition, interactive approaches combining the explicit function and the meta model can be employed where local meta models are estimated using progressively generated evaluations of the explicit function (e.g. [46], [15]). In multiobjective evolutionary optimization meta models play an important role in pre-screening points based on their predicted values and confidence of these predictions (e.g. [15]). A comparative study of different approaches is given in [42].

The key aspect, however, is that the meta model will always only approximate the original problem [23], which possibly leads to bias, wide confidence bands and also error in the obtained optimum, properties that cannot be neglected in evaluating a modeling approach. Resampling methods can be efficiently used for the following purposes:

Accuracy assessment of a given meta model: Clearly, a meta model without sufficient accuracy should not be used for optimization as biased results will be generated which will not have any relevance for the problem at hand. This stresses the importance of a systematic model validation which indicates the reliability and accuracy of the estimated model. In case the current meta model does not fulfill the requirements, possible alternatives are either a model update based on additional values obtained by more fitness function evaluations or using a different, possibly less complex, modeling technique.

Model selection: In many cases several model classes are candidates for fitting the desired meta model. Resampling methods and the related accuracy assessment efficiently support the selection process of the most appropriate and reliable model type. It should be noted, that it is usually advisable to choose a less complex model in order to achieve good results with the desired small sample sizes since more complex models usually require larger data sets to be sufficiently accurate.

Tuning of hyperparameters: Most modeling strategies require the setting of so-called hyperparameters (e.g. the parameters of the covariance kernel in kriging). Thus, a tuning of the hyperparameters is desired to determine the settings leading to highest model quality which can be evaluated using the discussed resampling techniques.

Unfortunately, not much attention has been given to model accuracy assessment in the evolutionary computation field so far. It is a somewhat disregarded topic, except from the comprehensive survey in [41]. In the following, model quality is solely reflected by model accuracy, which in our view is the most relevant aspect, although other aspects of model quality could also be relevant in some settings. For example, in the early stages of an optimization process, a smooth model might be considered more appropriate than a rough model since it will provide sufficient information to drive the optimization process and simultaneously diminishes the probability of being stuck in local optima. However,

smoothing an accurate but rough model provides this possibility as well and can at the same time in its accurate form be used at the end of the optimization process. Also one should take into account that being able to interpret a meta model is often seen as an important benefit that other optimization techniques lack (e.g. see [37]). A model that is merely helpful in controlling the optimization but does approximate the true structure of the data generating process well enough might therefore mislead one to draw wrong conclusions in a later stage of analysis.

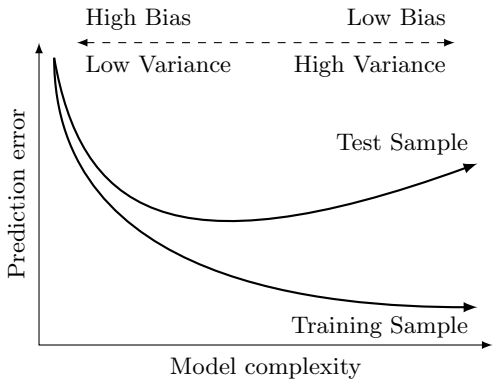

Fig. 1: With increasing model complexity overfitting on the training set becomes more likely [19].

The focus of this paper therefore is on stressing the importance of evaluating model accuracy in meta model optimization together with providing related practical guidelines and recommendations. We thoroughly discuss state of the art approaches with respect to their advantages and shortcomings. However, we do not claim to provide a comprehensive model validation approach covering all optimization specific aspects in general.

An overview of the most important resampling techniques for model validation such as cross validation, bootstrapping, subsampling and nested resampling as a combination of the former approaches is available in sections 2 and 3. Guidelines are presented for choosing an adequate resampling technique suitable for small sample sizes resulting from a small number of fitness evaluations. In section 4, specific attention is drawn to a discussion of potential pitfalls and important assumptions that have to be fulfilled. Conclusions are drawn in section 5. All of these methods and most of the comments are generally applicable for any regression or classification setting in machine learning, and for comparison some remarks are made regarding the latter.

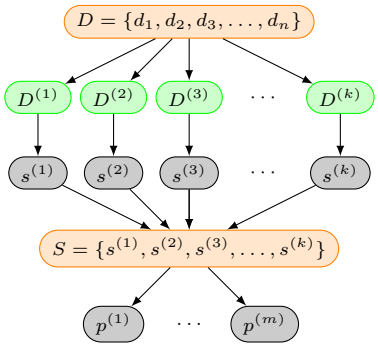

Fig. 2: Generic resampling scheme. A model validation function takes a training set ($D^{(i)}$) and an accompanying test set ($D \setminus D^{(i)}$). Different generators for the data subsets $D^{(i)}$ are cross-validation, bootstrapping or subsampling. S equals the set of aggregrated loss function values which are again aggregated by performance measures $p^{(i)}$.

2 Basic Resampling Techniques and Statistical Properties

Assume we are given a fitness function f, a set of points from the parameter space of f and the associated function values. Denote these by $\{d_1, d_2, d_3, \ldots, d_n\} = D$, where $d_i = (x_i, y_i)$ is a vector of covariates x_i and associated function values $y_i = f(x_i)$. Our aim now is to find a meta model which approximates f using the information contained in D, i.e. we want to fit a regression function $\hat{f}_D$ to our data D.

Now the question arises, how well our model $\hat{f}_D$ approximates the real fitness function f. This requires the definition of an appropriate loss function $L(y, \hat{f}(x))$. For regression this will be the squared loss $(y - \hat{f}(x))^2$ or the absolute loss $|y - \hat{f}(x)|$, if robustness is required with respect to outliers. These loss values are aggregated by the mean to form the mean squared error (MSE) and the mean absolute error (MAE).

One naive way to assess the quality of our model would be to measure the losses on the data we used for fitting. This generally cannot work, as the error measure will be optimistically biased in this setting, see fig. 1. In fact, for some models, it will always be zero, e.g. imagine an interpolating spline. One way to rectify this is to split the data into a *training set* D_{train} and a corresponding *test set* D_{test} such that $D_{\text{train}} \cup D_{\text{test}} = D$ and $D_{\text{train}} \cap D_{\text{test}} = \emptyset$. One would then train the model on D_{train} to obtain $\hat{f}_{D_{\text{train}}}$ and calculate its performance measure using the data points in D_{test}. This approach called the *hold-out* or *test set method* is very easy to implement and to work with from a statistical perspective, as the test set observations are completely independent from the training observations.

Still two important problems remain: A large data set D is required, since we need enough data in the training set to build an adequate model, but also enough samples in the test set to get statistically valid performance results. These large

amount of samples will usually not be available if evaluating f is expensive. Also, one will not detect variance and model instability due to changes in the training set. Some models, especially more complex[1], nonlinear ones, might produce very different results even when the training data only slightly change.

Algorithm 2: Generic resampling

input : A dataset D of n observations d_1 to d_n, the number of subsets k to generate and a loss function L.
output: Summary of the validation statistics.

1 Generate k subsets of D named $D^{(1)}$ to $D^{(k)}$
2 $S \leftarrow \emptyset$
3 **for** $i \leftarrow 1$ **to** k **do**
4 $\quad \bar{D}^{(i)} \leftarrow D \setminus D^{(i)}$
5 $\quad \hat{f} \leftarrow$ `FitModel` $(D^{(i)})$
6 $\quad s_i \leftarrow \sum_{(x,y) \in \bar{D}^{(i)}} L(y, \hat{f}(x))$
7 $\quad S \leftarrow S \cup \{s_i\}$
8 Summarize S

To deal with this situation, resampling techniques have been developed. All of these techniques repeatedly generate training and test sets from the data set at hand, fit a model to each training set and judge its quality on the corresponding test set (and possibly also on the training set). This general framework is depicted in Fig. 2. In the next three subsections we will introduce three different methods to generate these training/test pairs and will show how to estimate performance values.

2.1 Cross-validation

Cross-validation (CV) [38] is probably one of the oldest resampling techniques. Like all other methods presented in this paper it uses the generic resampling strategy as described in Alg. 2. The k subsets (line 1 of Alg. 2) are generated according to Alg. 3. The idea is to divide the dataset into k equally sized blocks and then use $k-1$ blocks to fit the model and validate it on the remaining block. This is done for all possible combinations of $k-1$ of the k blocks. The k blocks are usually called *folds* in the cross-validation literature. So a cross-validation with $k = 10$ would be called a *10 fold cross-validation*. Usual choices for k are 5, 10 and n.

[1] Note that although we use model complexity informally in this text, different rigorous definitions of this notion exist, among them VC dimension and Rademacher complexity [2].

Algorithm 3: Subsets for k-fold CV.

input : A dataset D of n observations d_1 to d_n and the number of subsets k to generate.
output: k subsets of D named $D^{(1)}$ to $D^{(k)}$.

```
1 D ← Shuffle (D)
2 for i ← 1 to k do
3 |   D^(i) ← D
4 for j ← 1 to n do
5 |   i ← (j mod k) + 1
6 |   D^(i) ← D^(i) \ {d_j}
```

Algorithm 4: Subsets for bootstrap.

input : A dataset D of n observations d_1 to d_n and the number of subsets k to generate.
output: k subsets of D named $D^{(1)}$ to $D^{(k)}$.

```
1 for i ← 1 to k do
2 |   D^(i) ← ∅
3 |   for j ← 1 to n do
4 |   |   d ← RandomElement (D)
5 |   |   D^(i) ← D^(i) ∪ {d}
```

This last case of $k = n$ is also referred to as leave-one-out cross-validation (LOOCV) because the model is fitted on the subsets of D which arise if we leave out exactly one observation. In general this is computationally hard but for certain classes of models the LOOCV estimate can be calculated efficiently. Examples include linear regression [45], splines [43], support vector regression [9] and kriging models [40].

2.2 Bootstrap

The development of the bootstrap resampling strategy [12] is almost as old as the idea of cross-validation. Again Alg. 2 is the basis of the method, but the k subsets are generated using Alg. 4. Here k is usually chosen much larger than in the CV case. Values of $k = 100$ up to $k = 1000$ are not uncommon and there is practically no upper limit on k[2] .

The subset generation is based on the idea that instead of sampling from D without replacement, as done in the CV case, we sample with replacement. One of the advantages of this approach is that the size of the training set, in the bootstrap literature often also called the *in bag observations*, is equal to the actual data set size. On the other hand this entails that some observations can and likely will be present multiple times in the training set $D^{(i)}$. In fact, asymptotically only about $1-e^{-1} \approx 63.2\%$ of the data points in D will be present in the training set [12]. The remaining $\approx 36.8\%$ of observations are called *out of bag* and form the test set as in CV.

The fact that some observations are present multiple times in the training set can be problematic for some meta modeling techniques. Several approaches have been proposed to deal with this. Most add a small amount of random noise to the observations. For details see [12].

[2] Do note, that there are n^n different bootstrap samples. So for very small n there are limits to the number of bootstrap samples you can generate.

Another effect of adding some observations multiple times to the training set is that we overemphasize their importance, called oversampling. This leads to an estimation bias for our validation statistic since all data points in the test set are now, in some sense, outliers. A first attempt to counter this was the so called .632 bootstrap procedure by [13]. Here the estimated error of the model is a weighted average of the error on the training set and the test set. The fallacy in this approach is that some modeling techniques will always have an error of 0 on the training set. An example of such a method would be an interpolating spline.

Algorithm 5: .632+ bootstrap

input : A dataset D of n observations d_1 to d_n, the number of subsets k to generate and a loss function L.
output: k values of the validation statistic.

1 Generate k subsets of D named $D^{(1)}$ to $D^{(k)}$
2 $S \leftarrow \emptyset$
3 **for** $i \leftarrow 1$ **to** k **do**
4 $\quad \bar{D}^{(i)} \leftarrow D \setminus D^{(i)}$
5 $\quad \hat{f} \leftarrow$ `FitModel` $(D^{(i)})$
6 $\quad \hat{\gamma} \leftarrow \frac{1}{n^2} \sum_{p,q=1}^{n} L(y_p, \hat{f}(x_q))$
7 $\quad s^{\text{in}} \leftarrow \frac{1}{n^2} \sum_{(x,y) \in D^{(i)}} L(y, \hat{f}(x))$
8 $\quad s^{\text{out}} \leftarrow \frac{1}{n^2} \sum_{(x,y) \in \bar{D}^{(i)}} L(y, \hat{f}(x))$
9 $\quad \hat{R} \leftarrow \frac{s^{\text{out}} - s^{\text{in}}}{\hat{\gamma} - s^{\text{in}}}$
10 $\quad \hat{w} \leftarrow \frac{0.632}{1 - 0.368\hat{R}}$
11 $\quad s_i \leftarrow (1 - \hat{w}) s_i^{\text{in}} + \hat{w} s_i^{\text{out}}$
12 $\quad S \leftarrow S \cup \{s_i\}$

To counter this, Efron proposed a further variant of the bootstrap named the .632+ bootstrap (see [14]). This strategy is a bit more involved and deviates somewhat from the framework proposed in Alg. 2. The details are given in Alg. 5. The main difference here is, that instead of fixed weights, as in the .632 bootstrap, the weights are calculated for each model to reflect how well the model can reproduce the training set.

The k subsets of D are again generated using Alg. 4. Then, as in the general framework, the model $\hat{f}$ is calculated in line 5. Line 6, 7, 9 and 10 are new. In line 6 the loss is estimated for the hypothetical case that our model has no predictive power. This is done by calculating the loss for each possible combination of x and y from D. Because there is now no direct dependence between x and y, the loss one observes tells us something about the error rate the model would achieve even if there was no link between x and y, i.e. our function was pure

noise. This quantity $\hat{\gamma}$ is called the *no-information error rate*. Using this and the *in-bag error rate* from line 7, as well as the usual *out-of-bag error rate* from line 8 the *relative overfitting rate* $\hat{R}$ is calculated in line 9. $\hat{R}$ lies between 0 and 1. If $\hat{R} = 1$, then the model is completely overfitted, i.e. it only has predictive power on the training set. If $\hat{R}$ is almost 0 on the other hand, then $\hat{f}$ has great predictive power on the test set and we can use the error on the training set to increase the accuracy of our error estimate. The final performance criterion is the weighted average of the in-bag and out-of-bag error rates as calculated in line 11 using the weight derived in line 10 from the relative overfitting rate.

2.3 Subsampling

Subsampling (SS) is very similar to the classical bootstrap. The only difference is that observations are drawn from D without replacement. Therefore the training set has to be smaller than D or no observations would remain for the test set. Usual choices for the subsampling rate $|D^{(i)}|/|D|$ are 4/5 or 9/10. This corresponds to the usual number of folds in cross-validation ($k = 5$ or $k = 10$). Like in bootstrapping, k has to be selected a-priori by the user. Choices for k are also similar to bootstrapping, e.g. in the range of 100 to 1000.

Algorithm 6: Subset generation algorithm for subsampling.

input : A dataset D of n observations d_1 to d_n, the number of subsets k to generate and the subsampling rate r.
output: k subsets of D named $D^{(1)}$ to $D^{(k)}$.

```
1 m ← ⌊r|D|⌋
2 for i ← 1 to k do
3     D' ← D
4     D^(i) ← ∅
5     for j ← 1 to m do
6         d ← RandomElement (D')
7         D^(i) ← D^(i) ∪ {d}
8         D' ← D' \ {d}
```

2.4 Further methods

Many variants, extensions and combinations of the above algorithms exist, which we cannot all present as comprehensively as the standard methods presented in the previous section. Some examples of these methods are:

Repeated cross-validation: (REPCV) performs a usual k-fold CV multiple times and aggregates the results (normally by the mean) to reduce the variance of randomly splitting the data.

Bootstrap cross-validation: (BCV) generates b bootstrap samples of D and performs cross-validation on each bootstrap sample, then aggregates the resulting error rates by averaging them. This has been shown to be advantageous for small sample sizes. For a detailed description see [17] and [20], but note comment 10 in section 4 and the variation proposed in [20], where entries in the test set are removed, which also occur in the training set.

Stratified cross-validation: (SCV) ensures that all folds of the cross-validation include a roughly equal number of observations per region of the input space. Several methods to achieve this have been proposed, for an example see [10].

3 Nested Resampling

Often the process of deriving a model requires a more complex approach (including model selection) than simply fitting a regression model. This will be the case when hyperparameters (e.g. consider support vector regression) or covariates have to be selected (e.g. screening the covariates for the ones which have a relevant influence on the target value y). Since the optimal hyperparameters are often data-dependent, they cannot be chosen without looking at the data. On the other hand one cannot perform these model selection steps using the same resampling sets that are used for evaluating the model itself, as this can lead to severely biased performance results (see comments 4 and 5 in section 4). This effect is sometimes called "training on the test set", since through the repeated evaluation of the model on the test data, information regarding its structure enters the model which might lead to overfitting.

Another explanation of why overfitting will occur is illustrated by the following thought experiment: Imagine a model, which randomly assigns predictions without using the covariates and with a hyperparameter σ which has no influence on the quality of the fit. "Optimizing" σ over a large number of iterations now corresponds to choosing the minimum statistic of the error values of all generated random models. This minimum will be the lower the more iterations we perform. It will however not be a reasonable estimator regarding the performance on new data (which would be the mean of all observed errors or expected error rate on an unused test set).

Instead, the model selection process has to be treated as part of the fitting and therefore has to be repeated for every training set. Unfortunately, as this kind of model selection usually requires a resampling approach as well, one ends up with a nested process of sampling. While the model is evaluated in an outer loop, every training set is resampled again in an inner loop to select an appropriate model, e.g. see [35]. As an example, consider using subsampling with $k = 100$ for evaluation and 5-fold cross-validation for hyperparameter selection. For each of the 100 training sets $D^{(i)}$ from subsampling, a 5-fold cross-validation on the training set is employed as internal fitness evaluation to select the best setting for the hyperparameters of the regression function. This is an optimization problem itself, but we are not going to cover the differences between the usual approaches here. Typically, optimization methods are used for which no

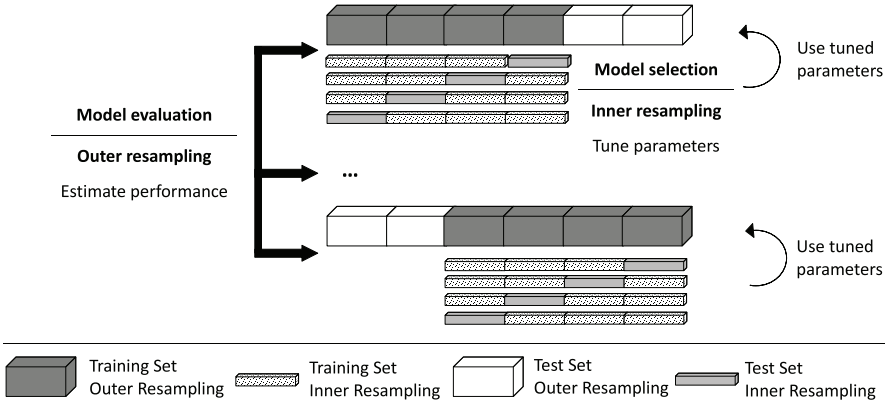

Fig. 3: Nested resampling with two nested cross-validations

parameters have to be specified a priori in order to avoid another level of hyperparameter optimization. The resulting optimization quality of these approaches usually is sufficient for this purpose. The best obtained hyperparameters are used to fit the model on the complete training set and calculate the error on the test set of the outer resampling strategy. Fig. 3 shows this process schematically for two nested cross-validations.

While this course of action is computationally very expensive, it is currently the only known way to ensure unbiased results. If runtime has to be reduced, more efficient optimization or a coarser resampling for model selection in the inner loop should be employed. If the latter produces a negative effect for model selection, at least this will be noticeable in a valid way in the outer evaluation.

4 Common Pitfalls, Recommendations and Statistical Properties

In practical applications, choosing the right resampling method and experimental setup in order to evaluate estimated models might become a tough decision – even for an experienced data analyst. We will describe the most common misconceptions and pitfalls in this section and try to give reasonable recommendations and some theoretical results relevant for applied work. There exist a number of other papers with a similar goal, often from the field of classification, which we build upon. See for example [18, 24, 26, 27, 35] and for more general recommendations [33].

1. **Terminology.**
   Make sure to check formal definitions of resampling methods in papers. Unfortunately, even in the ones cited in this one, sometimes different names are used or even exchanged. For example in [6] what we call out-of-bag bootstrap

is called bootstrap cross-validation (and therefore used differently than we defined BCV) and in [30] what we call subsampling is called cross-validation. Other terms for subsampling also include Monte Carlo cross-validation [34] and repeated hold-out [24].

2. **Stochastic performance values are not a disadvantage.**
   We have encountered a specific preference towards the leave-one-out estimator among non-statisticians "because it does not produce results subject to stochastic variation", but instead delivers a reproducible, deterministic value. This is a misconception which stems from the notion that your given data set completely determines your task. On the contrary, the data was obtained by a stochastic (real-world or simulated) process itself. The task is not to optimize a machine learning model w.r.t. to this data set, but to use the data efficiently to minimize the loss on new, unseen observations. This is and will always be a stochastic optimization problem. Information about uncertainty and statistical variation due to slight changes in the data and unstable models are an advantage and not a drawback.
3. **Be precise when stating how model selection and evaluation were performed.**
   As the interpretation of results relies strongly on the chosen experimental setup, the necessity of this statement should be self-evident. Still, some authors seem to treat this issue as a "minor detail, only worth brief mentioning in passing". Also, when code is not published along with a paper, reproducibility of results sometimes becomes difficult if not impossible.
4. **Do not report performances from repeated tuning on the same resampling.**
   Even though this problem and its remedy have already been discussed in section 3, we emphasize this common error here again, as from our experience this still repeatedly turns up in applied work. Disregarding this makes interpretations and comparisons to other results nearly impossible, as one can never be sure how much positive results are due to the "training on the test set" effect. Quite a number of publications had to be rectified because of this mistake, see [16] for a recent example. Use nested resampling.
5. **Include all modeling steps into the resampling.**
   If data-dependent preprocessing or pre-selection of variables is performed, these have to be included in the resampling as well to avoid biased, optimistic results. Although not doing so is basically the same mistake as the one described in the comment above, this happens more frequently as preprocessing is usually considered independently of the model fitting and performed once in the beginning of the analysis. Simon et al. show in [36] how the reevaluation of experiments from a published paper, in which covariates were excluded by a statistical test in preprocessing, lead to much worse results than it would have been the case if the selection was included into a complete resampling. They also confirm this conclusion by further simulation studies regarding the selection of covariates.
6. **Do not use hold-out, CV with few iterations or subsampling with a low subsampling rate for small samples.**

These methods have a large bias resulting from the reduced training set size (and a larger variance due to to the one or only few random split-ups, except for subsampling), which is especially hurtful when data is scarce anyway.

7. **Comments regarding Leave-One-Out and Cross-validation.**
   Leave-one-out cross-validation has better properties for the squared loss used in regression than for its usual 0-1 counterpart in classification and is an almost unbiased estimator for the mean loss [26]. It can be shown that it is asymptotically equivalent to the well-known AI criterion (AIC), the jackknife and the bootstrap [13, 39]. Although these asymptotic equivalences are important and interesting in their own right, we want to point out that many practically relevant differences exist between the mentioned methods, some discussed in the comments in this section.

   The many possibilities for its efficient computation mentioned in section 2.1 and its deterministic, reproducible value (but see comment 2) and its near unbiasedness make LOO an attractive candidate among the presented algorithms, especially when only few samples are available. But one should be aware of the following facts: LOOCV has a high variance [26, 44] as estimator of the mean loss, meaning quite different values are produced if the data used for cross-validation slightly changes. It also is asymptotically inconsistent and tends to select too complex models. In [34] theoretical reasons for this effect are presented, and subsampling and balanced leave-d-out CV are shown to be superior estimators in a simulation study. Kohavi [26] arrives at similar results regarding LOO and demonstrates empirically that 10-fold CV is often superior. He suggests a stratified version.

   For these reasons we recommend LOO mainly for efficient model selection, still keeping in mind that this might lead to somewhat suboptimal choices. Repeated and stratified CV will usually produce more reliable results in practice.

8. **Comments on different bootstrap variants.**
   Many variations on the algorithms presented in section 2.2 exists. We will here restrict ourselves to comparing the simple out-of-bag bootstrap, using only s^{out}, the older .632, which always sets $\hat{w} = 0.632$, and the .632+. While the out-of-bag bootstrap is pessimistically biased in the sense that it bases its performance values on models which use only about 63.2% of the data, .632 can be optimistic in a much worse way, as complex models can easily achieve $s^{in} = 0$ (or consider adding a "memory" of the training data to your model). Both estimators are known to have a low variance, and out-of-bag is especially good, when the sample size is small and the error or noise in the data is high [44]. .632+ combines the best properties of both estimators and can generally be trusted to achieve very good results with small sample sizes. Its main drawback are the difficult combination with tuning (see comment 10) and that it might result in an optimistic bias, when more complex models are considered [24, 27].

9. **Choosing the number of iterations in bootstrapping, subsampling, etc.**

These questions seem to remain an open research issue, and the general consensus seems to be to err on the safe by choosing a large k. 100, 250 and 500 are common values in published papers, [44] recommends at least 200 to obtain a good estimate. But a reasonable setting has always to be data-dependent, relying on the number of available samples, the complexity of the task and applied models. One should be aware that this will be computationally very expensive.

There exists a couple of heuristics to select the number of iterations adaptively, especially in model selection and optimization. For example consider the repeated testing performed in F-Races [7]. But for valid statistical inference one will have to employ methods from the sequential analysis. To our knowledge, this is not very often done in machine learning, and a comprehensive overview of these techniques is beyond the scope of this article. We refer the reader to [28] for an accessible introduction.

10. **Bootstrapping or subsampling? Repeated observations can be problematic.**
When combining model or complexity parameter selection with bootstrapped data sets, repeated observations can lead to a substantial bias towards more complex models. This stems from the fact that with a high probability measurements will occur both in the training set as well as the test set, so that more complex models "memorizing" the training data will seem preferable. This effect was documented in the setting of using boosting for high-dimensional microarray data [6] and subsampling was proposed and evaluated as a remedy.
11. **Independence, Confidence Intervals and Testing.**
In general, the generated training and test sets, and therefore the obtained performance statistics will not be independent when sampling from a finite data set. This has negative consequences if confidence intervals for the performance measure should be calculated. The dependence structure is especially complicated for the commonly used cross-validation, where the split-up of the data in one iteration completely depends on all other split-ups. It can be shown that in this setting no unbiased estimator of the variance exists [4] and pathological examples can be constructed, where the variance estimator performs arbitrarily bad. Extensions on corrections for subsampling exist which take the dependence between sampled data sets into account and provide a much better foundation for interval estimators and subsequent statistical tests regarding location parameters [30].
The focus in this paper has largely been on model selection and not necessarily on model comparison. For this several procedures using statistical hypothesis tests have been proposed by [1, 11, 20, 30].
12. **Software and illustrating examples.**
While many machine learning toolboxes contain at least some of the mentioned resampling strategies, we would like to point out that our own *R* package *mlr* [8] contains all of the methods mentioned in this paper, including generic nested resampling, and an interface to many regression, classification and optimization algorithms in *R*.

R examples to demonstrate specific issues from this section are made available at `http://statistik.uni-dortmund.de/~bischl/ppsn2010_res`.

4.1 Comparison to relevant work

To our best knowledge the mentioned methodology in this article is not very often used to assess and improve meta models in optimization. We are only aware of a paper by Tenne and Armfield [41], where a similar approach is followed and will compare to their work. The authors propose to use hold-out, 10-fold CV, LOO and the .632 bootstrap to estimate the prediction error of a quadratic, a kriging and and radial basis function network model on three different test functions. They do not include the .632+ bootstrap or repeated CV in their study, although these estimators might be more appropriate for their small sample sizes (less than 100). They also optimize the hyperparameters of the anisotropic kriging model on the full data set in the beginning, which might lead to a bias in the following resampling caused by overfitting.

Otherwise their findings agree with our comments above: they state that hold-out has an pessimistic bias, as fewer data is used for training, LOO has a large variance, the 10-fold CV also has a large variance due to the random splitting of the few samples and .632 has a low variance and an optimistic bias. However, they do no provide references for these facts, which have also been observed by other authors in comparison studies in machine learning [14, 24, 26, 27, 34, 35, 45]. They report LOO as the most suited but computationally expensive resampling method for model assessment, but without mentioning the many approaches for its efficient calculation (see 2.1).

5 Conclusions and Outlook

Meta modeling techniques play a prominent role in modern expensive optimization procedures. Much research has focused on the design of powerful modeling methods. In this paper we have given an overview of well-known resampling methods from statistics and how they can be used to choose or tune a meta model.

Proper model validation in this context is crucial. First of all, employing a meta-model which badly approximates the original objective function cannot lead to reliable optimization results. Secondly, hyperparameter tuning for a specific model class, selecting from a finite amount of different models and possibly choosing relevant features are important steps in practical optimization. It is well known that their correct application leads to an improvement in generalization performance. As in model validation, a properly evaluated performance measure is needed in order to compare among the potential models. But when very few sample points are combined with very powerful and flexible regression models like kriging, overfitting becomes likely. Therefore, the model validation procedure plays a key role for determining a useful model and subsequently performing efficient optimization.

After introducing the three basic strategies, cross-validation, bootstrapping and subsampling, nested resampling is discussed and common pitfalls and design mistakes are highlighted when using these methods. As a summarizing recommendation, we suggest to consider efficient Leave-One-Out cross-validation for fast model tuning, the .632+ bootstrap for small samples sizes with low complexity models and when no tuning is required, and subsampling or repeated cross-validation in all other cases, see Fig. 4.

In the future we will perform a similar benchmark study like [41]. We will include more advanced resampling methods, which are specifically tailored for the small sample sizes encountered in global optimization of expensive functions, like repeated cross-validation and the .632+ bootstrap. We will also investigate the question, whether the usual approach of selecting hyperparameters on the full sample in the beginning leads to overfitting and an optimistically biased estimator in the subsequent resampling, and if so, whether this can be avoided by nested resampling. From these results we will derive further practical recommendation for resampling evaluations in the field of optimization by meta modeling.

In addition, different facets of model quality with respect to optimization processes will be studied and incorporated into to the presented guidelines for model accuracy assessment.

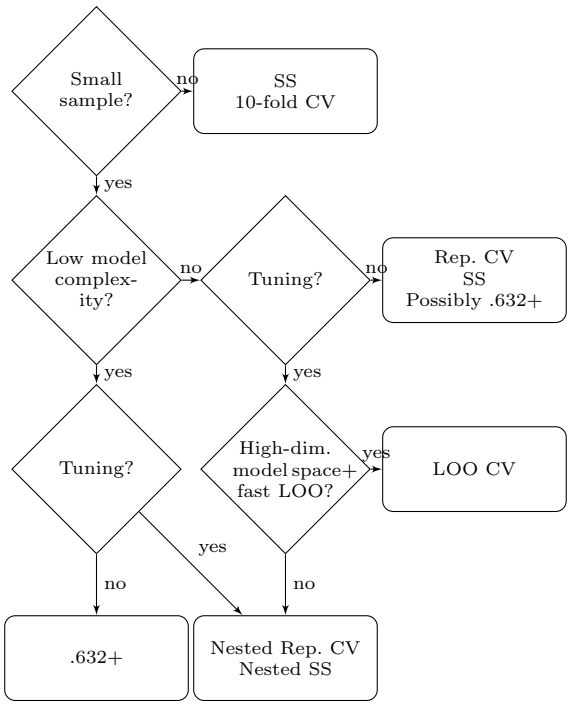

Fig. 4: Guidelines for selecting a resampling strategy.

Acknowledgements This work was partly supported by the Collaborative Research Center SFB 823 and the Research Training Group "Statistical Modelling" of the German Research Foundation.

References

1. Alpaydin, E.: Combined 5x2cv f test for comparing supervised classification learning algorithms. Neural Computation 11, 1885–1892 (1999)
2. Bartlett, P., Boucheron, S., Lugosi, G.: Model selection and error estimation. Machine Learning 48(1-3), 85–113 (2002)
3. Bartz-Beielstein, T.: Experimental Research in Evolutionary Computation – The New Experimentalism. Springer Natural Computing Series, Berlin (2006)
4. Bengio, Y., Grandvalet, Y.: No unbiased estimator of the variance of k-fold cross-validation. Journal of Machine Learning Research 5, 1089–1105 (2004)
5. Bhattacharya, M.: Meta model based ea for complex optimization. International Journal of Computational Intelligence 1(4), 36–45 (2008)
6. Binder, H., Schumacher, M.: Adapting prediction error estimates for biased complexity selection in high-dimensional bootstrap samples. Statistical Applications in Genetics and Molecular Biology 7(1), Article 12 (2008)
7. Birattari, M., Stutzle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 11–18. Morgan Kaufmann (2002)
8. Bischl, B.: mlr: Machine learning in R, http://mlr.r-forge.r-project.org (2010)
9. Cawley, G., Talbot, N.: Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. Neural Networks 17(10), 1467–1475 (2004)
10. Diamantidis, N., Karlis, D., Giakoumakis, E.: Unsupervised stratification of cross-validation for accuracy estimation. Artificial Intelligence 116(1-2), 1–16 (2000)
11. Dietterich, T.: Approximate statistical tests for comparing supervised classification learning algorithms. Neural Computation 10(7), 1895–1923 (1998)
12. Efron, B.: Bootstrap methods: Another look at the jackknife. The Annals of Statistics 7(1), 1–26 (1979)
13. Efron, B.: Estimating the error rate of a prediction rule: Improvement on cross-validation. Journal of the American Statistical Association 78(382), 316–331 (1983)
14. Efron, B., Tibshirani, R.: Improvements on cross-validation: The 0.632 + bootstrap method. Journal of the American Statistical Association 92(438), 548–560 (1997)
15. Emmerich, M., Giannakoglou, K., Naujoks, B.: Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels. IEEE Transactions on Evolutionary Computation 10(4), 421–439 (2006)
16. Fiebrink, R., Fujinaga, I.: Feature selection pitfalls and music classification. In: ISMIR. pp. 340–341 (2006)
17. Fu, W.J., Carroll, R.J., Wang, S.: Estimating misclassification error with small samples via bootstrap cross-validation. Bioinformatics 21(9), 1979–1986 (2005)
18. Good, P.: Resampling Methods: A Practical Guide to Data Analysis. Birkhauser (2005)
19. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer (2001)
20. Hothorn, T., Leisch, F., Zeileis, A., Hornik, K.: The design and analysis of benchmark experiments. Journal of Computational and Graphical Statistics 14, 675–699 (2005)

21. Huang, D., Allen, T., Notz, W., Zeng, N.: Global optimization of stochastic black-box systems via sequential kriging meta-models. J. of Global Optimization 34(3), 441–466 (2006)
22. Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing 9(1), 3–12 (2005)
23. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments – a survey. IEEE Transactions on Evolutionary Computation 9(3), 303–318 (2005)
24. Kim, J.H.: Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. Computational Statistics and Data Analysis 53(11), 3735–3745 (2009)
25. Knowles, J., Nakayama, H.: Multiobjective Optimization: Interactive and Evolutionary Approaches, chap. Meta-Modeling in Multiobjective Optimization, pp. 245–284. Springer (2008)
26. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: IJCAI. pp. 1137–1143 (1995)
27. Molinaro, A., Simon, R., Pfeiffer, R.: Prediction error estimation: a comparison of resampling methods. Bioinformatics 21(15), 3301–3307 (2005)
28. Mukhopadhyay, N., de Silva, B.M.: Sequential methods and their applications. Chapman & Hall/CRC (2009)
29. Myers, R., Montgomery, D.: Response Surface Methodology. Wiley, New York (1995)
30. Nadeau, C., Bengio, Y.: Inference for the generalization error. Machine Learning 52(3), 239–281 (2003)
31. Nakayama, H., Yun, Y., Yoon, M.: Sequential Approximate Multiobjective Optimization Using Computational Intelligence. Springer Publishing Company, Incorporated (2009)
32. Ong, Y., Nair, P., Keane, A.: Evolutionary optimization of computationally expensive problems via surrogate modeling. AIAA Journal 41(4), 687–696 (2003)
33. Salzberg, S.: On comparing classifiers: Pitfalls to avoid and a recommended approach. Data Mining and Knowledge Discovery 1(3), 317–328 (1997)
34. Shao, J.: Linear model selection by cross-validation. Journal of the American Statistical Association 88(422), 486–494 (1993)
35. Simon, R.: Fundamentals of Data Mining in Genomics and Proteomics, chap. Resampling Strategies for Model Assessment and Selection, pp. 173–186. Springer, US (2007)
36. Simon, R., Radmacher, M.D., Dobbin, K., McShane, L.M.: Pitfalls in the use of dna microarray data for diagnostic and prognostic classification. Journal of the National Cancer Institute 95(1), 14–18 (2003)
37. Smit, S., Eiben, A.: Comparing parameter tuning methods for evolutionary algorithms. In: IEEE Congress on Evolutionary Computation (CEC). pp. 399–406 (2009)
38. Stone, M.: Cross-validatory choice and assessment of statistical predictions. Journal of the Royal Statistical Society, Series B 36(1), 111–147 (1974)
39. Stone, M.: An asymptotic equivalence of choice of model by cross-validation and akaike's criterion. Journal of the Royal Statistical Society, Series B 39, 44–47 (1977)
40. Sundararajan, S., Keerthi, S.: Predictive approaches for choosing hyperparameters in gaussian processes. Neural Computation 13(5), 1103–1118 (2001)
41. Tenne, Y., Armfield, S.: Metamodel accuracy assessment in evolutionary optimization. In: IEEE Congress on Evolutionary Computation. pp. 1505–1512 (2008)

42. Wagner, T., Emmerich, M., Deutz, A., Ponweiser, W.: On expected-improvement criteria for model-based multi-objective optimization. In: Schaefer, R. (ed.) Parallel Problem Solving from Nature. Springer, Berlin (2010)
43. Wahba, G.: Spline bases, regularization, and generalized cross validation for solving approximation problems with large quantities of noisy data. In: Proceedings of the International Conference on Approximation Theory in Honour of George Lorenz. Academic Press (1980)
44. Weiss, S., Kulikowski, C.: Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1991)
45. Wu, C.: Jackknife, bootstrap and other resampling methods in regression analysis. Annals of Statistics 14, 1261–1295 (1986)
46. Younis, A., Dong, Z.: Metamodelling using search space exploration and unimodal region elimination for design optimization. Engineering Optimization 6(42), 517–533 (2010)
47. Yun, Y., Yoon, M., Nakayama, H.: Multi-objective optimisation based on meta-modelling using support vector regression. Optimization and Engineering 10(2), 167–181 (2009)

A Rank Transformation Can Improve Sequential Parameter Optimization

Simon Wessing[1] and Tobias Wagner[2]

[1] Chair of Algorithm Engineering, TU Dortmund
Otto-Hahn-Straße 14, 44227 Dortmund, Germany
simon.wessing@tu-dortmund.de, http://ls11-www.informatik.uni-dortmund.de
[2] Institute of Machining Technology (ISF), TU Dortmund
Baroper Straße 301, 44227 Dortmund, Germany
wagner@isf.de, http://www.isf.de

Abstract. Over recent years, parameter tuning of Evolutionary Algorithms (EAs) is attracting more and more interest. In particular, the sequential parameter optimization (SPO) framework for model-assisted tuning procedures resulted in established parameter tuning algorithms. Most variants of SPO apply Kriging for modeling the fitness landscape and, thereby, finding an optimal parameter configuration on a limited budget of EA runs. In this work, we enhance the SPO framework by introducing transformation steps before the aggregation and before the modeling. We empirically show that a rank transformation of the data improves the mean performance of SPO and is superior to other transformations, such as the Box-Cox and the logarithmic transformation.

Keywords: Data Transformation, Design and Analysis of Computer Experiments, Kriging, Sequential Parameter Optimization (SPO)

1 Introduction

Parameter tuning of evolutionary algorithms (EAs) is a noisy optimization problem with expensive evaluations. An approach to this problem is SPO by Bartz-Beielstein et al. [1], which usually uses Kriging models to predict new promising search points. Kriging is based on the assumption that the modeled data follows a deterministic, stationary multivariate Gaussian process (GP). This cannot be guaranteed to be fulfilled in general. For example, distributions of objective values often show an unwanted skewness. Hutter et al. [10] have demonstrated that it can be advantageous for SPO to apply a logarithmic transformation to the data before the modeling. In geostatistics, which is the original application area of Kriging, this technique is known as lognormal Kriging. Our goal is to investigate if other techniques considered in geostatistics are equally successful in our application. In particular, a rank transformation seems promising since it is established to process highly skewed data [12, 21].

The remainder of the paper is organized as follows. In Sect. 2, an introduction to the theoretical aspects of Kriging is given. Sect. 3 describes the SPO

framework with our proposed modifications. In Sect. 4, we present experiments to examine our hypotheses about the benefit of data transformations. Sect. 5 concludes the paper and provides an outlook on future research topics.

2 Basics

Parameter tuning of EAs represents an optimization problem $\min_{\mathbf{x}\in\mathbf{X}} y(\mathbf{x})$.[3] $\mathbf{X}$ is called design space. A vector $\mathbf{x} \in \mathbf{X}$ is denoted as design point. It contains the settings of the considered tuning parameters $x_i,\ i = 1, \ldots, n$. Usually, the design space $\mathbf{X}$ is bounded by box constraints $\mathbf{l}$ and $\mathbf{u}$, whereby $l_i \leq x_i \leq u_i$ define the region of interest (ROI). The performance of the algorithm $y(\mathbf{x})$ is evaluated by performing experiments on a representative test instance $\mathcal{T}$. In $\mathcal{T}$, information about the optimization task, i. e., the test function or simulator output of interest, and the desired target quality or an allowed runtime budget are encoded. Based on the outcomes of the experiments for $\mathbf{x}_i$, an aggregated response y_i can be derived. Based on stochastic effects in the algorithm or simulator, the actual objective value of a design can only be evaluated as $y_i = y(\mathbf{x}_i) + \epsilon_i$, where ϵ_i can be denoted as random error or noise of an evaluation.

In this paper, we particularly focus on Kriging models as used in the Design and Analysis of Computer Experiments [17, 18]. Kriging models $\mathcal{M}$ are based on a set of designs $\mathbf{D} = (\mathbf{x}_1^T, \ldots, \mathbf{x}_k^T)^T$ and the corresponding responses $\mathbf{y} = (y_1, \ldots, y_k)^T$, which approximate the actual objective function $y(\mathbf{x})$. They consider the k responses as $y_j = \mathbf{f}(\mathbf{x}_j)^T\boldsymbol{\beta} + Z(\mathbf{x}_j),\ j = 1, \ldots, k$, where the vector $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_p(\mathbf{x}))^T$ contains p monomial regression functions, e. g., x^2 or $\tan(x)$, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T$ is a p-dimensional vector of corresponding regression coefficients, and $\mathrm{Z}(\mathbf{x})$ is a zero mean (centered) stationary Gaussian process with dependencies specified by the covariance $Cov\{Z(\mathbf{x}_{j_1}), Z(\mathbf{x}_{j_2})\} = \sigma_Z^2 r(\mathbf{x}_{j_1}, \mathbf{x}_{j_2})$ for a known correlation function r and a process variance σ_Z^2. Consequently, it is assumed that the residuals to the regression function can be described as a multivariate Gaussian distribution.

It can be shown [18] that the best linear unbiased predictor $\hat{y}(\mathbf{x})$ with respect to the mean squared prediction error is

$$\hat{y}(\mathbf{x}) = \underbrace{\mathbf{f}(\mathbf{x})^T\hat{\boldsymbol{\beta}}}_{\text{regression function}} + \mathbf{r}(\mathbf{x})^T\mathbf{R}^{-1} \underbrace{(\mathbf{y} - \mathbf{F}\hat{\beta})}_{\text{observed residuals}}, \tag{1}$$

where $\hat{\boldsymbol{\beta}} = \left(\mathbf{F}^T\mathbf{R}^{-1}\mathbf{F}\right)^{-1}\mathbf{F}^T\mathbf{R}^{-1}\mathbf{y}$ are the regression coefficients estimated in the sense of least squares and $\mathbf{F} = (\mathbf{f}(\mathbf{x}_1) \cdots \mathbf{f}(\mathbf{x}_k))^T$ is the $k \times p$ matrix of regression function values for each of the k designs. Analogously, the vector of correlations between $\mathbf{x}$ and the already evaluated designs $\mathbf{r}(\mathbf{x}) = (r(\mathbf{x}, \mathbf{x}_1), \ldots, r(\mathbf{x}, \mathbf{x}_k))^T$ and the $k \times k$ intercorrelation matrix $\mathbf{R} = (\mathbf{r}(\mathbf{x}_1) \cdots \mathbf{r}(\mathbf{x}_k))$ are defined in terms of the correlation function r. Usually, a constant regression function $f(\mathbf{x}) = \beta_1$

[3] Minimization problems are considered in this paper. Maximization problems can be transformed to corresponding minimization problems $\min_{\mathbf{x}\in\mathbf{X}} -y(x)$.

Algorithm 7: Pseudocode of the considered SPO approach

Require: $\mathcal{T}$ {test instance}
$\mathbf{l}, \mathbf{u}$ {ROI}
N {experimental budget}
N_{init} {size of the initial design set}

1: $\mathbf{D} = LatinHypercubeSampling(\mathbf{l}, \mathbf{u}, N_{init})$ {generate initial design}
2: $\mathbf{Y} = runDesign(\mathbf{D})$ {perform experiments}
3: **while** $entries(\mathbf{Y}) < N$ **do**
4: $\quad \widetilde{\mathbf{Y}} = transformLocal(\mathbf{Y})$ {transformation of the responses}
5: $\quad \mathbf{y} = aggregateRuns(\widetilde{\mathbf{Y}})$ {calculate performance indices}
6: $\quad \widetilde{\mathbf{y}} = transformGlobal(\mathbf{y})$ {transformation of the performance indices}
7: $\quad \mathcal{M} = fitModel(\mathbf{D}, \widetilde{\mathbf{y}})$ {fit empirical model of the response}
8: $\quad \mathbf{d}_{new} = modelOptimization(\mathcal{M})$ {find promising design points}
9: $\quad \mathbf{Y} = \mathbf{Y} \cup runDesign(\mathbf{d}_{new})$ {perform experiments and add results}
10: **end while**
11: **return** $\mathcal{M}$, $\mathbf{d}^*$ {return final Kriging model and the best design}

and a nonisotropic Gaussian kernel $r(\mathbf{x}_{j_1}, \mathbf{x}_{j_2}) = \exp\left(-\sum_{i=1}^{n} \theta_i |x_{j_1,i} - x_{j_2,i}|^2\right)$ are used. The model parameters $\boldsymbol{\theta}$ control the activity of the Gaussian process in each dimension. The model $\mathcal{M}$ based on the Gaussian correlation kernel is infinite times differentiable.

Based on the strength of the correlations, also the corresponding prediction uncertainty $\hat{s}(\mathbf{x})$ can be computed [18]. The predictions can then be interpreted to follow a normal distribution $\mathcal{N}(\hat{y}(\mathbf{x}), \hat{s}(\mathbf{x}))$ with mean $\hat{y}(\mathbf{x})$ and standard deviation $\hat{s}(\mathbf{x})$. By these means, the incorporation of the accuracy of an evaluation is possible [8, 16].

3 Approach

The framework of our SPO approach is shown in Algorithm 7. As usual, the initial design $\mathbf{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{N_{init}}\}$ for the model generation is obtained by computing a Latin hypercube sample (LHS) [14] in the specified ROI. Then, R_{init} runs of the EA with different random seeds are performed for each design $\mathbf{x} \in \mathbf{D}$ on the test instance defined by $\mathcal{T}$.

In the known SPO approaches, the aggregation (AG) into a single performance value for each design is carried out directly. In contrast, we introduce a local transformation (LT) step before the results of the different runs are aggregated. The aim of LT is the preprocessing of the result distribution of each run in order to improve the estimation of the performance index in AG, e. g., a symmetric distribution for the estimation of the mean and the standard deviation. An approximately normal distribution of the results allows the uncertainty of the evaluation to be incorporated into Kriging models with nugget effect [6, 8, 16]. By these means, the robustness of a design point can also be considered.

In this paper, we empirically analyze the statistically established logarithmic, Box-Cox [4] and rank transformation [5]. All transformations are rank-

preserving, i. e., the ordering of the designs is not changed. The Box-Cox transformation is a power transformation

$$y_i^{(\lambda)} = \begin{cases} \frac{y_i^\lambda - 1}{\lambda(\mathrm{GM}(\mathbf{y}))^{\lambda-1}}, & \text{if } \lambda \neq 0 \\ \mathrm{GM}(\mathbf{y}) \log y_i, & \text{if } \lambda = 0 \end{cases}, \tag{2}$$

where $\mathrm{GM}(\mathbf{y})$ denotes the geometric mean of all responses. It allows a continuously defined change-over between no ($\lambda = 1$) and a logarithmic transformation ($\lambda = 0$). Nevertheless, the logarithmic transformation is also considered since it has already been used in the SPO framework before [10]. The exponent λ in the Box-Cox transformations is usually optimized using maximum likelihood estimation.

For a rank transformation, the responses are ordered and ranks are assigned starting with 1 for the best response. Consequently, the data is transformed to a uniform distribution given an complete order on the response. If ties occur, all responses are substituted by the averaged rank, i. e., the ordered responses $y_1 = 0.1$, $y_2 = 0.3$, $y_3 = 0.3$, and $y_4 = 1$ would become $\tilde{y}_1 = 1$, $\tilde{y}_2 = 2.5$, $\tilde{y}_3 = 2.5$, and $\tilde{y}_4 = 4$. Whereas the Box-Cox transformation can be performed design-wise using a fixed exponent λ estimated based on the whole data set, the rank transformation has to be computed based on ranks calculated from all available results. All these transformations can improve the characteristics of the result distribution, such as skewness, kurtosis, and the fit with the normal distribution.

After LT, the different results for each design are forwarded to AG. Usually, the mean of the runs is used in SPO. The mean is defined as average over all observed and transformed results making its estimation sensitive to outliers. Thus, the use of the median – as a more robust performance index for AG – is additionally analyzed in this paper. As already observed by Hutter et al. [10], the LT changes the actual aggregation of the individual responses. The combination of the mean and a logarithmic LT results in the logarithm of the geometric mean of the untransformed responses. This is particularly interesting with respect to often observed success or failure scenarios in multimodal optimization because the geometric mean is well suited to aggregate binary random variables. Moreover, a local Box-Cox transformation can provide a slight shift from the arithmetic to the geometric mean.

The global transformation (GT) is performed after the aggregated responses have been computed. In this step, two aims have to be achieved. For the modeling, the responses should follow a multivariate Gaussian distribution. For the sequential optimization, a clearer identification of the optimal basin is desired. Hutter et al. [10, 11] have already shown that a logarithmic transformation is able to achieve both aims for some selected test instances. In this paper, we consider a Box-Cox and a rank transformation as possible alternatives. The Box-Cox transformation is known for the ability to transform skewed, but still unimodal, distributions to normality, whereas the rank transformation will result in almost uniform distributions. Consequently, these transformations may act as extreme solutions to achieve the before-mentioned aims.

Finally, a Kriging model $\mathcal{M}$ is computed from the transformed results, as described in the previous Sect. 2. In an optimization loop, the model is then used to predict promising parameter configurations based on the second moment of the expected improvement [19]. The new candidates are evaluated and the data is fed back into the model.

4 Experiments

Research Question: Is there a static combination of LT, AG, and GT that is superior in general?
Preexperimental planning: Since we are interested in the effect of the transformations in a practical parameter tuning scenario, the enhanced SPO is used to tune standard EAs on established test problems. Moreover, we do not require to define a theoretical noise distribution. The distribution is created by the characteristics of the EA. To achieve general results, we are using two single-objective and two multi-objective test instances, where the one is uni- and the other is multimodal. The performance of a configuration is measured by the mean y^* of the responses of the EA for the design $\mathbf{d}^*$ returned as the best design by SPO. The mean has been chosen since it penalizes outliers with bad results and corresponds to the natural aggregation used in a Kriging model with nugget effect [6, 8]. This study is a first step to analyze the effect of transformations in SPO. Consequently, the estimation of the robustness of a design point, i. e., the corresponding standard deviation of the results, is not considered. In line with this decision, also the use of the nugget effect will be omitted in this study.
Task: The task is to analyze the effect of local and global transformation functions when applied in SPO on different kinds of test instances $\mathcal{T}$. In particular, the general and the problem-specific appearance of effects has to be distinguished.
Setup: The considered test cases $\mathcal{T}$ are summarized in Table 1. The multi-objective problems S_ZDT1 and R_ZDT4 are taken from the CEC 2007 competition [9]. SPO is given a budget of $N = 500$ EA runs. As single-objective EA, the $(\mu\ ,\ \lambda)$-Evolutionary Strategy (ES) [20] is chosen. It is using Gaussian mutation with self-adapted step sizes. For the multi-objective problems, the S-metric-selection-based multi-objective EA (SMS-EMOA) [2] is used. In this algorithm, polynomial mutation with fixed step size $\eta_m = 15$ and probability $p_m = 1/n$ and simulated binary crossover (SBX) [7] are applied. The corresponding tuning parameters and their ROI are defined in Table 2. For each EA, three parameters are to be tuned: population size μ, selection pressure λ/μ, and initial step size σ_{init} in the single-objective case and μ, distribution index η_c, and crossover probability p_c in the multi-objective case. The logarithmic transformations are performed since DACE models are relying on a stationary GP, i. e., a constant activity of the response over the considered domain (see, e. g. [3]). In contrast, a change of the population size from $\mu = 1$ to $\mu = 10$ surely has a bigger effect on the response compared to a change from $\mu = 100$ to $\mu = 110$. The change of the magnitude of the parameter is more important than the change in the absolute

value. The same holds for the the selection pressure λ/μ and the initial step size σ_{init}.

For the initial design $\mathbf{D}$, $N_{init} = 30$ design points and $R_{init} = 4$ runs of each design point are used. Consequently, less than a quarter of the experimental budget is utilized for the initial design, thus focusing on the sequential model-based optimization. On single-objective problems, the objective value can be directly used as response of the corresponding run, whereas in the multi-objective case the difference in the hypervolume indicator [13] with respect to a Pareto-optimal reference set is computed. This is related to the measure internally optimized in the SMS-EMOA.

Log and Box-Cox transformations are only valid in the domain $\mathbb{R}^+$. To ensure this constraint to be met, we transform the responses shifted in the positive domain $\widetilde{\mathbf{Y}} = transformLocal(\mathbf{Y} - \min\{\mathbf{Y}\} + \epsilon)$, where ϵ denotes the machine precision. The effects of AG, LT, and GT are analyzed within a threeway analysis of variance (ANOVA) [15]. In this ANOVA, the effects of the factors (grouping variables) are evaluated by comparing the variance explained by the factors with the variance that cannot be explained. The factors and factor levels considered are summarized in Table 3.

Results/Visualization: The ANOVA tables for the test functions are shown in Tables 4, 5, 6, and 7. The confidence intervals of selected multiple comparison tests for interaction effects of LT and GT based on the results of the ANOVAs are shown in Fig. 1 and Fig. 2. Histograms of the aggregated responses' distributions are provided in Fig. 3. Moreover, the effects of the global transformation on the input data of the Kriging model are visualized. In Fig. 4 and 5, the distibutions of the data before and after LT, AG, and GT are analyzed using histograms based on the results of successful combinations. A combination of LT and GT that leads to undesired results is shown in Fig. 6.

Observations: The most important main effect, which is significant at the 0.01 level in the ANOVAs of all considered test functions, is the one of AG. The effect of GT is significant at the 0.01 level on Rastrigin and S_ZDT1, but can only be confirmed at the 0.05 level on the sphere function. In contrast, S_ZDT1 is the sole test case where LT shows a significant effect. The most important interaction effect is the one of LT and GT, which is significant at the 0.01 level for S_ZDT1 and at the 0.05 level for the sphere problem and Rastrigin. Consequently, we focus on the main effect of AG and GT and the interaction between LT and GT in the following.

Table 1: Overview of the test instances $\mathcal{T}$ used in the experiments.

Problem	Variables	Objectives	Algorithm	Evaluations	Reference point
Sphere	30	1	$(\mu\ ,\ \lambda)$-ES	10000	–
Rastrigin	10	1	$(\mu\ ,\ \lambda)$-ES	50000	–
S_ZDT1	30	2	SMS-EMOA	10000	$(2.05, 10.45)^T$
R_ZDT4	10	2	SMS-EMOA	20000	$(4.15, 524.95)^T$

Table 2: Summary of tuning parameters considered in the experiments including their respective ranges and additional transformations.

	(μ , λ)-ES			SMS-EMOA		
Parameter	μ	λ/μ	σ_{init}	μ	η_c	p_c
ROI	$\{1,\ldots,100\}$	$[2,100]$	$[0.001,(\mathbf{u}-\mathbf{l})/2]$	$\{1,\ldots,1000\}$	$[1,50]$	$[1/n,1]$
Transf.	$\log_{10}$	$\log_{10}$	$\log_{10}$	$\log_{10}$	none	none

Table 3: The factors of the full factorial design and their levels.

Factor	LT	AG	GT
Default	none	mean	none
Levels	{none, Box-Cox, log, rank}	{mean, median}	{none, Box-Cox, log, rank}

A multiple comparison test based on the results of the ANOVA shows that the mean aggregation always shows a significant improvement ($\alpha = 0.05$) compared to the median. Considering GT, a rank transformation is significantly superior to the Box-Cox transformation on the sphere problem, a logarithmic or no transformation on Rastrigin, and to all other transformations on S_ZDT1. In return, the rank transformation is not significantly outperformed by any other GT. The confidence intervals with respect to the different combinations of LT and GT are shown in Fig. 1 and Fig. 2. For the sphere problem and R_ZDT4, no plots are shown since no significant differences can be detected. On Rastrigin, the Box-Cox transformation is superior to no GT when LT is omitted or a rank transformation is performed. On S_ZDT1, all pairwise combinations of Box-Cox and logarithmic transformations result in a bad performance of SPO. This leads to an outperformance of the logarithmic transformation by the approach using no GT based on a grouping of all possible local transformations. Nevertheless,

Table 4: ANOVA table for the full factorial experiment on the sphere problem.

Source	Sum Sq.	d.f.	Mean Sq.	F	Prob> F	
Local Trans. (LT)	2.2e-5	3	7.4e-6	2.00	0.1119	
Aggregation (AG)	2.7e-5	1	2.7e-5	7.40	0.0066	< 0.01
Global Trans. (GT)	3.0e-5	3	1.0e-5	2.76	0.0412	< 0.05
LT*AG	1.2e-5	3	3.9e-6	1.06	0.3655	
LT*GT	7.6e-5	9	8.5e-6	2.30	0.0146	< 0.05
AG*GT	1.7e-5	3	5.5e-6	1.50	0.2133	
LT*AG*GT	2.2e-5	9	2.5e-6	0.68	0.7316	
Error	5.8e-3	1568	3.7e-6			
Total	6.0e-3	1599				

Table 5: ANOVA table for the full factorial experiment on Rastrigin.

Source	Sum Sq.	d.f.	Mean Sq.	F	Prob> F	
Local Trans. (LT)	1.4e+1	3	4.7e+0	1.94	0.1213	
Aggregation (AG)	3.6e+1	1	3.6e+1	14.92	0.0001	< 0.01
Global Trans. (GT)	3.7e+1	3	1.2e+1	5.02	0.0018	< 0.01
LT*AG	4.1e+1	3	1.4e+1	5.57	0.0008	< 0.01
LT*GT	4.3e+1	9	4.7e+0	1.94	0.0429	< 0.05
AG*GT	1.2e+1	3	4.0e+0	1.64	0.1780	
LT*AG*GT	2.9e+1	9	3.3e+0	1.34	0.2095	
Error	3.8e+3	1568	2.4e+0			
Total	4.0e+3	1599				

Table 6: ANOVA table for the full factorial experiment on S_ZDT1.

Source	Sum Sq.	d.f.	Mean Sq.	F	Prob> F	
Local Trans. (LT)	4.0e-4	3	1.3e-4	33.47	0.0000	< 0.01
Aggregation (AG)	2.7e-5	1	2.7e-5	6.77	0.0093	< 0.01
Global Trans. (GT)	5.8e-4	3	1.9e-4	48.21	0.0000	< 0.01
LT*AG	5.9e-6	3	2.0e-6	0.49	0.6872	
LT*GT	1.3e-3	9	1.5e-4	37.26	0.0000	< 0.01
AG*GT	1.2e-5	3	4.1e-6	1.02	0.3835	
LT*AG*GT	2.4e-5	9	2.6e-6	0.66	0.7476	
Error	6.2e-3	1568	4.0e-6			
Total	8.6e-3	1599				

the worst results are obtained when performing no transformations at all, which corresponds to the classical SPO approach.

In the histgrams of Fig. 3, all nontransformed distributions are heavily skewed to the right. By applying a logarithmic transformation, the distribution becomes left-skewed on the sphere and approximately symmetric on the Rastrigin problem, whereas the response distributions of the multi-objective problems are still right-skewed. The Box-Cox transform is able to reduce the skewness on all considered test instances. Only on R_ZDT4, the distribution is still slightly skewed to the right. Based on the plots of transformed responses, it can be seen that the distributions are not unimodal. Both the logarithmic and the Box-Cox transformation are rank-preserving. Consequently, they cannot change the modality of the distribution.

In Fig. 4 and Fig. 5, which visualize the effect of the different transformation steps for successful combinations of LT and GT, an improvement of the residuals' fit with the normal distribution can be observed after the LT. Moreover, the skewness of the original distribution of the aggregated responses can be reduced by the GT. Nevertheless, no approximately normal distribution of the aggregated responses can be obtained due to the multimodality. As shown Fig. 6, a

Table 7: ANOVA table for the full factorial experiment on R_ZDT4.

Source	Sum Sq.	d.f.	Mean Sq.	F	Prob> F	
Local Trans. (LT)	3.6e+1	3	1.2e+1	1.17	0.3201	
Aggregation (AG)	6.3e+2	1	6.3e+2	61.28	0.0000	< 0.01
Global Trans. (GT)	2.0e+1	3	6.7e+0	0.65	0.5840	
LT*AG	1.5e+1	3	4.9e+0	0.48	0.6998	
LT*GT	9.3e+1	9	1.0e+1	1.01	0.4317	
AG*GT	4.6e+1	3	1.5e+1	1.51	0.2112	
LT*AG*GT	5.8e+1	9	6.4e+0	0.63	0.7739	
Error	1.6e+4	1568	1.0e+1			
Total	1.7e+4	1599				

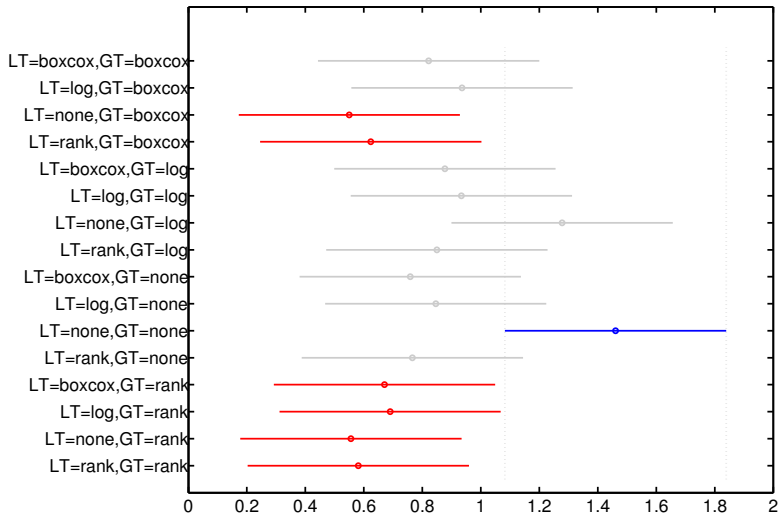

Fig. 1: Confidence intervals of the multiple comparison test for the interaction effect of LT and GT based on the results of the ANOVA on Rastrigin. The colors indicate significant differences (red outperforms blue) at the $\alpha = 0.05$ level. The gray confidence intervals are overlapping, thus, marking incomparable results.

combination of equal or similar, e. g., Box-Cox and logarithmic, transformations can result in a severe change of the response distribution that deteriorates the performance of SPO.

Discussion: Of course, the significance of the effect of AG is biased by the decision for using the mean over all runs as performance criterion. This decision was made based on practical considerations, such as the robustness with respect to negative outliers and the interpretation of a Kriging prediction as normal distri-

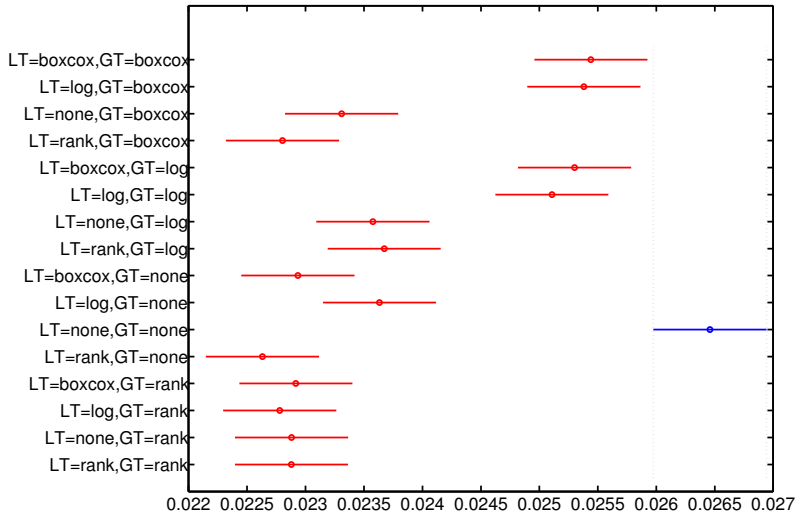

Fig. 2: Confidence intervals of the multiple comparison test for the interaction effect of LT and GT based on the results of the ANOVA on S_ZDT1. The colors indicate significant differences (red outperforms blue) compared to the version without transformations at the $\alpha = 0.05$ level.

bution defined by the predicted mean $\hat{y}(\mathbf{x})$ and standard deviation (uncertainty) $\hat{s}(\mathbf{x})$. This interpretation also motivates the use of LT for improving the fit of the results' empirical distribution for each run with a normal distribution, which is shown to be successful in Fig. 4 and Fig. 5. However, the effect of LT in the experiments is low since SPO uses a Kriging model without nugget effect, which results in an uncertainty prediction of zero for the observed designs. It can be assumed that the effect of LT will gain importance when using Kriging models with heterogenous nugget effect [16], because these models can incorporate uncertainty estimates from the data.

The improvements obtained by the rank transformation on most of the considered problems are stunning. In particular with regard to the smooth and continously defined Kriging models with Gaussian kernel, the good performance is surprising since the rank transformation introduces a stair structure to the modeled response. An explanation of this effect may be based on the skewness of the aggregated responses. In Fig. 3 it is shown that the results' distribution over the runs in SPO is highly skewed. Continuous transformations can reduce the skewness, but as shown on R_ZDT4, they are not always successful. The transformation to a uniform distribution, which is performed by a rank transformation, directly results in a non-skewed distribution (cf. Fig. 4). Consequently, the center of mass is shifted to the center of the Gaussian process used for the

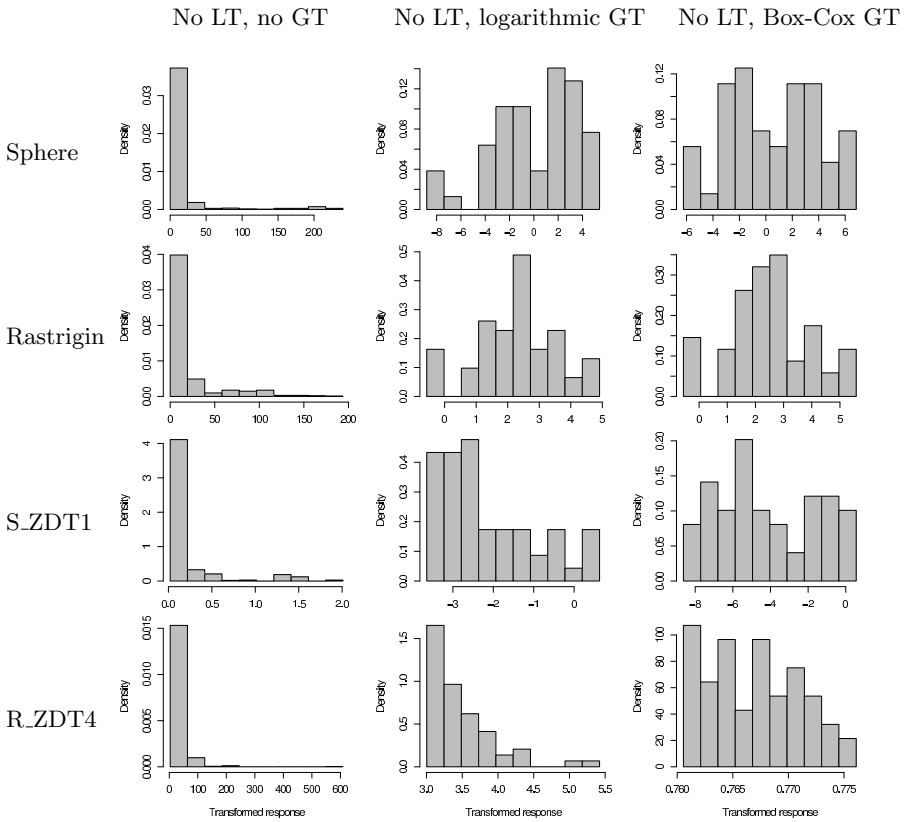

Fig. 3: Histograms of the aggregated responses' distributions. In the left column, the original data obtained by SPO runs with the default configuration are shown. In the middle and right column, the histograms of the respective transformations of these responses are presented.

Kriging prediction. This results in a higher efficiency of the sequential optimization in SPO.

The absence of significant results on R_ZDT4 is based on a floor effect. The test function is to hard to be solved by any parameterization of the SMS-EMOA. Thus, a better setup of SPO cannot provide significant improvements.

5 Conclusion and Outlook

Based on four test instances representing different problem classes, we analyzed the utility of applying different AG, LT, and GT functions within the SPO framework. Given the mean performance as quality criterion, the mean was superior to the more robust median estimate. Furthermore, a rank-transformation of the aggregated data before the modeling improved the results of SPO. An additional

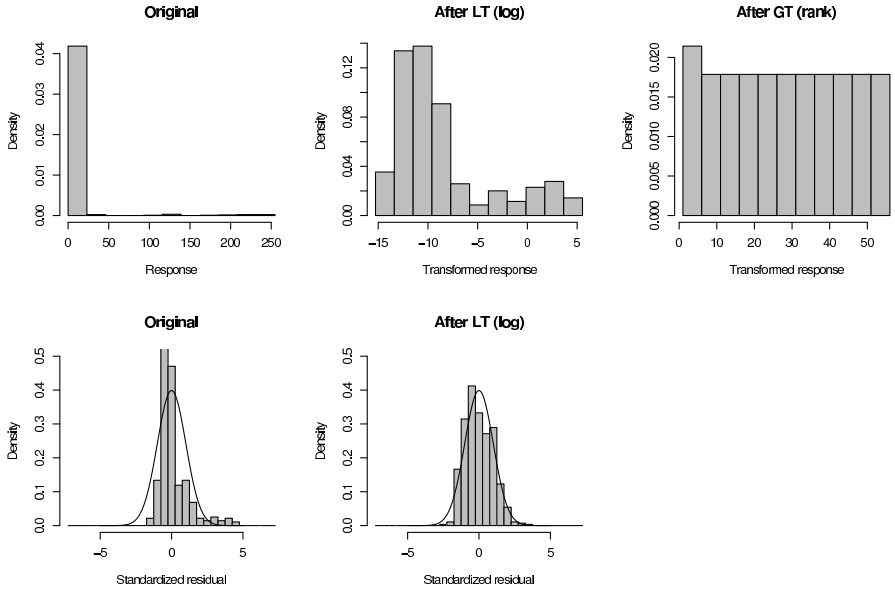

Fig. 4: Histograms of the complete set of observed responses before (left) and after (center) the LT and of the final aggregated responses after GT (right) for the successful combination of a logarithmic LT and a rank GT on the sphere problem. At the bottom, the effect of the LT on the normality of the residuals, which are computed based on the estimated mean and standard deviation of each parameter vector, is shown.

transformation – rank, logarithmic, or Box-Cox – prior to the aggregation resulted in an improved fit of the residuals with the normal distribution, but did not show significant improvements for all considered test instances. Nevertheless, this LT has to be in accordance with the GT after the aggregation since a strong interaction between both transformations exists.

The superiority of the rank transformation has been shown based on an established and well-chosen, but small set of test instances. All these test instances result in heavily right-skewed distributions of aggregated response values. By transforming the results to a uniform distribution, the rank GT obtains a non-skewed training sample for Kriging. Additional experiments on further test instances should be performed in order to support this result's generality. The weak effect of the LT may be related to the chosen Kriging approach without nugget effect. Thus, the analysis of LT should also be performed in the context of Kriging models with homo- and heterogeneous nugget effect. Based on the variety of different kinds of problems, an adaptive approach that selects the transformation function based on a data analysis of the observed responses is still desired. We have developed first adaptive approaches we are analyzing right now.

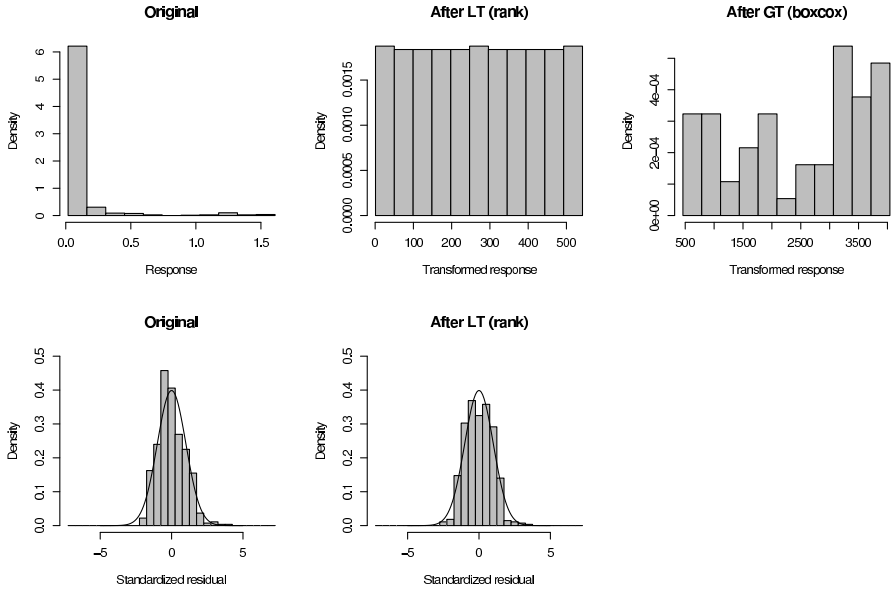

Fig. 5: Histograms of the complete set of observed responses before (left) and after (center) the LT and of the final aggregated responses after GT (right) for the successful combination of a rank LT and a Box-Cox GT on S_ZDT1. At the bottom, the effect of the LT on the normality of the residuals, which are computed based on the estimated mean and standard deviation of each parameter vector, is shown.

Acknowledgments This paper is based on investigations of the collaborative research center SFB/TR TRR 30, which is kindly supported by the Deutsche Forschungsgemeinschaft (DFG).

References

1. Bartz-Beielstein, T., Lasarczyk, C., Preuss, M.: Sequential parameter optimization. In: McKay, B., et al. (eds.) Proc. 2005 IEEE Congress on Evolutionary Computation (CEC 2005). pp. 773–780. IEEE press, Los Alamitos, CA (2005)
2. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. European Journal of Operational Research 181(3), 1653–1669 (2007)
3. Biermann, D., Joliet, R., Michelitsch, T., Wagner, T.: Sequential parameter optimization of an evolution strategy for the design of mold temperature control systems. In: Fogel, G., Ishibuchi, H., et al. (eds.) Proc. 2010 IEEE Congress on Evolutionary Computation (CEC 2010). IEEE Press, Las Alamitos, CA (2010)
4. Box, G.E.P., Cox, D.R.: An analysis of transformations. Royal Statistical Society, Series B 26(2), 211–252 (1964)
5. Conover, W.J., Iman, R.L.: Rank transformations as a bridge between parametric and nonparametric statistics. American Statistician 35(3), 124–129 (1981)

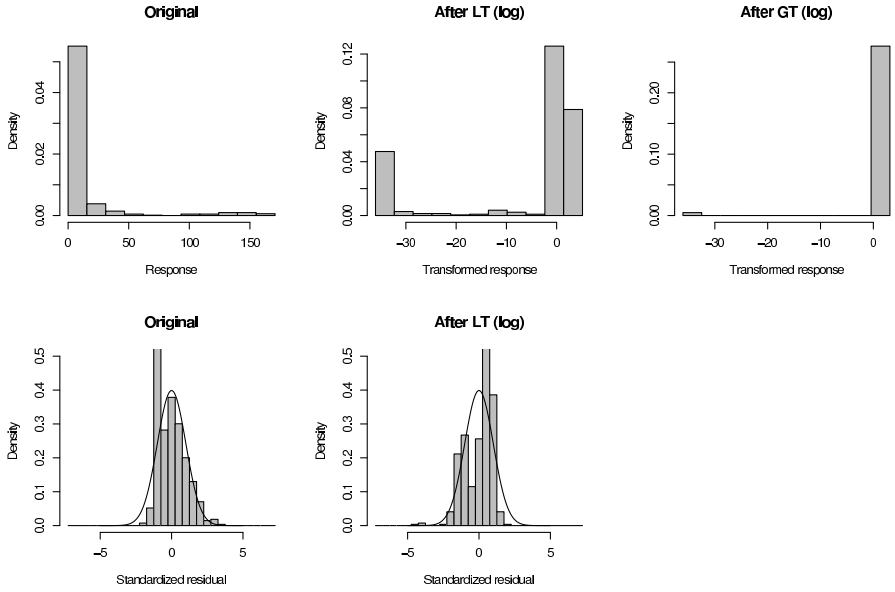

Fig. 6: Histograms of the complete set of observed responses before (left) and after (center) the LT and of the final aggregated responses after GT (right) for the poorly performing combination of a logarithmic LT and GT on Rastrigin. At the bottom, the effect of the LT on the normality of the residuals, which are computed based on the estimated mean and standard deviation of each parameter vector, is shown.

6. Cressie, N.A.C.: Statistics for Spatial Data. John Wiley and Sons, New York (1993)
7. Deb, K., Agrawal, R.B.: Simulated binary crossover for continuous search space. Complex Systems 9, 115–148 (1995)
8. Huang, D., Allen, T.T., Notz, W.I., Zheng, N.: Global optimization of stochastic black-box systems via sequential kriging meta-models. Global Optimization 34(4), 441–466 (2006)
9. Huang, V.L., Qin, A.K., Deb, K., Zitzler, E., Suganthan, P.N., Liang, J.J., Preuss, M., Huband, S.: Problem definitions for performance assessment of multi-objective optimization algorithms. Tech. rep., Nanyang Technological University, Singapore (2007), `http://www3.ntu.edu.sg/home/epnsugan/index_files/CEC-07/CEC07.htm`
10. Hutter, F., Bartz-Beielstein, T., Hoos, H.H., Leyton-Brown, K., Murphy, K.: Sequential model-based parameter optimisation: an experimental investigation of automated and interactive approaches. In: Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuss, M. (eds.) Empirical Methods for the Analysis of Optimization Algorithms, pp. 361–411. Springer, Berlin Heidelberg (2010)
11. Hutter, F., Hoos, H.H., Leyton-Brown, K., Murphy, K.P.: An experimental investigation of model-based parameter optimisation: SPO and beyond. In: Raidl, G., et al. (eds.) Proc. Genetic and Evolutionary Computation Conf. (GECCO 2009). pp. 271–278. ACM, New York, NY (2009)

12. Journel, A.G., Deutsch, C.V.: Rank order geostatistics: A proposal for a unique coding and common processing of diverse data. In: Baafi, E., Schofield, N. (eds.) Geostatistics Wollongong '96. vol. 1, pp. 174–187. Kluwer Academic (1997)
13. Knowles, J.D., Thiele, L., Zitzler, E.: A tutorial on the performance assessment of stochastic multiobjective optimizers. Tech. rep., Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich (2005)
14. McKay, M.D., Conover, W.J., Beckman, R.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21(1), 239–245 (1979)
15. Montgomery, D.C.: Design and Analysis of Experiments. John Wiley and Sons, New York, 4. edn. (1997)
16. Picheny, V., Ginsbourger, D., Richet, Y.: Noisy expected improvement and on-line computation time allocation for the optimization of simulators with tunable fidelity. In: Rodrigues, H., et al. (eds.) Proc. 2nd Int'l Conf. Engineering Optimization (EngOpt 2010) (2010)
17. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Statistical Science 4(4), 409–435 (1989)
18. Santner, T.J., Williams, B.J., Notz, W.: The Design and Analysis of Computer Experiments. Springer, New York, NY (2003)
19. Schonlau, M., Welch, W.J., Jones, D.R.: Global versus local search in constrained optimization of computer models. In: Rosenberger, W.F., Flournoy, N., Wong, W.K. (eds.) New Developments and Applications in Experimental Design, vol. 34, pp. 11–25. Institute of Mathematical Statistics, Hayward, CA (1997)
20. Schwefel, H.P.: Evolution and Optimum Seeking. Wiley-Interscience, NY, USA (1995)
21. Singh, A.K., Ananda, M.M.A.: Rank kriging for characterization of mercury contamination at the East Fork Poplar Creek, Oak Ridge, Tennessee. Environmetrics 13(5–6), 679–691 (2002)

Optimizing Support Vector Machines for Stormwater Prediction

Patrick Koch, Wolfgang Konen, Oliver Flasch, and Thomas Bartz-Beielstein

Department of Computer Science, Cologne University of Applied Sciences,
51643 Gummersbach,Germany {patrick.koch, wolfgang.konen, oliver.flasch,
thomas.bartz-beielstein}@fh-koeln.de

Abstract. In water resource management, efficient controllers of stormwater tanks prevent flooding of sewage systems, which reduces environmental pollution. With accurate predictions of stormwater tank fill levels based on past rainfall, such controlling systems are able to detect state changes as early as possible. Up to now, good results on this problem could only be achieved by applying special-purpose models especially designed for stormwater prediction. The question arises whether it is possible to replace such specialized models with state-of-the-art machine learning methods, such as Support Vector Machines (SVM) in combination with consequent parameter tuning using sequential parameter optimization, to achieve competitive performance. This study shows that even superior results can be obtained if the SVM hyperparameters and the considered preprocessing is tuned. Unfortunately, this tuning might also result in overfitting or oversearching – both effects would lead to declined model generalizability. We analyze our tuned models and present possibilities to circumvent the effects of overfitting and oversearching.

1 Introduction

Environmental engineering offers important concepts to preserve clean water and to protect the environment. Stormwater tanks are installed to stabilize the load on the sewage system by preventing rainwater from flooding the sewage network and by supplying a base load in dry periods. Mostly, heavy rainfalls are the reason for overflows of stormwater tanks, causing environmental pollution from wastewater contaminating the environment. To avoid such situations, the effluent of the stormwater tanks must be controlled effectively and possible future state changes in the inflow should be detected as early as possible. The time series regression problem of predicting a stormwater tank fill level at time t from a fixed window of past rainfall data from time t back to time $t - W$ will be referred to as the *stormwater problem* in the remainder of this paper.

A model that predicts fill levels by means of rainfall data can be an important aid for the controlling system. Special sensors (Fig. 1) record time series data which can be used to train such a model.

Although many methods exist for time series analysis [4], ranging from classical statistical regression to computational statistics, such methods often require

Fig. 1: Left: rain gauge (pluviometer). Right: stormwater tank.

time-consuming investigations on the hyperparameter selection and preprocessing of the data. Besides that, the results are often worse than special-purpose models which are designed from scratch for each new problem. This situation is of course very unsatisfying for the practitioner in environmental engineering, because new models have to be created and parameters have to be tuned manually for each problem.

For this reason, it would be an advantage to have some standard repertoire of methods which can be easily adapted to new problems. In this paper we use support vector machines (SVMs) [6] for Support Vector Regression as a state-of-the-art method from machine learning and apply them to the stormwater problem. SVMs are known to be a strong method for classification and regression. It has to be noted here, that because of the data sets are time series, records are not necessarily independent of each other, as in normal regression. Therefore we investigate generic preprocessing operators to embed time series data and to generate new input features for the SVM model. In addition, we use sequential parameter optimization (SPOT) [2] and a genetic algorithm (GA) to find good hyperparameter settings for both preprocessing and SVM parameters. We analyze the robustness of our method against overfitting and oversearching of hyperparameters. Preliminary work in stormwater prediction has been done by Hilmer [9], Bartz-Beielstein *et al.* [3] and Flasch *et al.* [7]. A conclusion of these previous publications is that good results can be obtained with specialized models (which are 'hand-crafted' and carefully adapted to the stormwater problem). The main hypotheses of this paper are:

H1 It is possible to move away from domain-specific models without loss in accuracy by applying modern machine learning algorithms and modern parameter tuning methods on data augmented through generic time-series preprocessing operators.

H2 Parameter tuning for stormwater prediction leads to oversearching, yielding too optimistic results on the dataset during tuning.

Hypothesis **H2** puts emphasis on the fact that a distinction between validation set (used during tuning) and test set (used for evaluation) is essential to correctly quantify the benefits from parameter tuning in data mining. The oversearching

issue is prevalent in data mining since the output function to tune shows often a high variance when the data used for training or tuning are selected differently.

2 Methods

2.1 Stormwater Tank Data

Time series data for this case study are collected from a real stormwater tank in Germany and consists of 30,000 data records, ranging from April to August 2006. Rainfall data are measured in three-minute intervals by a pluviometer as shown in Fig. 1. As training set we always used a 5,000 record time window (Set 2, Fig. 2) in order to predict another 5,000 record time window for testing (Set 4) which is not directly successive with regard to time to the training period (see Tab. 1). Later, we conducted a hyperparameter tuning with SPOT and feature selection where we used all 4 different datasets {Set1, Set2, Set3, Set4}, each containing 5,000 records to analyze the robustness of our approach against oversearching and overfitting.

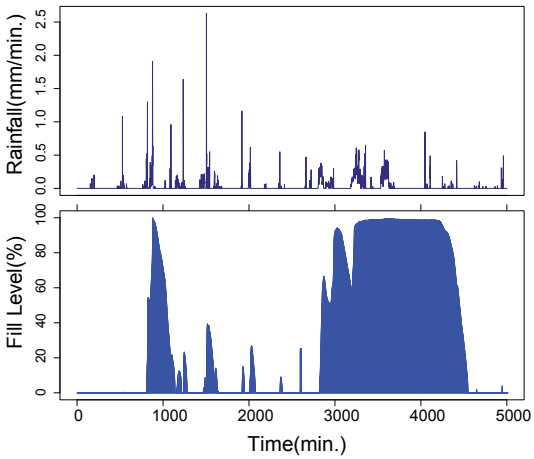

Fig. 2: Time window used for training (Set 2) containing rainfall and fill level of the stormwater tank.

Table 1: Real-world time series data from a stormwater tank in Germany.

Set	Start Date	End Date
Set 1	2006-04-28 01:05:59	2006-05-15 09:40:59
Set 2	2006-05-15 09:40:59	2006-06-01 18:20:59
Set 3	2006-06-19 03:01:00	2006-07-06 11:41:00
Set 4	2006-07-23 20:21:00	2006-08-10 05:01:00

2.2 Evaluation of Models

The prediction error on the datasets is taken as objective function for SPOT and for the GA. For comparing models, we calculate the root mean squared error (RMSE) as a quality measure, where $\langle\cdot\rangle$ denotes averaging over all measurements:

$$RMSE = \sqrt{\langle (Y_{predicted} - Y_{true})^2 \rangle} \tag{1}$$

We also incorporate a naïve prediction, always predicting the mean value of the training period.

2.3 Sequential Parameter Optimization

The main purpose of SPOT is to determine improved parameter settings for search and optimization algorithms and to analyze and understand their performance.

During the first stage of experimentation, SPOT treats an algorithm A as a black box. A set of input variables $\boldsymbol{x}$, is passed to A. Each run of the algorithm produces some output $\boldsymbol{y}$. SPOT tries to determine a functional relationship F between $\boldsymbol{x}$ and $\boldsymbol{y}$ for a given problem formulated by an objective function $f : \boldsymbol{u} \rightarrow \boldsymbol{v}$. Since experiments are run on computers, pseudorandom numbers are taken into consideration if:

- the underlying objective function f is stochastically disturbed, e.g., measurement errors or noise occur, and/or
- the algorithm A uses some stochastic elements, e.g., mutation in evolution strategies.

SPOT employs a sequentially improved model to estimate the relationship between algorithm input variables and its output. This serves two primary goals. One is to enable determining good parameter settings, thus SPOT may be used as a tuner. Secondly, variable interactions can be revealed for helping in understanding how the tested algorithm works when confronted with a specific problem or how changes in the problem influence the algorithm's performance. Concerning the model, SPOT allows for insertion of virtually any available model. However, regression and Kriging models or a combination thereof are most frequently used. The Kriging predictor used in this study uses a regression constant λ which is added to the diagonal of the correlation matrix. Maximum likelihood estimation was used to determine the regression constant λ [1, 8].

2.4 The INT2 Model for Predictive Control of Stormwater Tanks

In previous works [3, 12], the stormwater tank problem was investigated with different modeling approaches, among them FIR, NARX, ESN, a dynamical system based on ordinary differential equations (ODE) and a dynamical system based on integral equations (INT2). All models in these former works were systematically optimized using SPOT [2]. Among these models the INT2 approach turned out to be the best one [3]. The INT2 model is an analytical regression

model based on integral equations. Disadvantages of the INT2 model are that it is a special-purpose model only designed for stormwater prediction and that it is practically expensive to obtain an optimal parameter configuration: the parameterization example presented in [3] contains 9 tunable parameters which must be set. In this paper we compare hand-tuned INT2 parameters with the best parameter configuration found by SPOT in former study [3].

2.5 Support Vector Machines

Support Vector Machines have been successfully applied to regression problems by Drucker *et al.* [6], Müller *et al.* [14], Mattera and Haykin [13], and Chan and Lin [5]. In these studies the method has been shown to be superior to many other methods especially when the dimensionality of the feature space is very large.

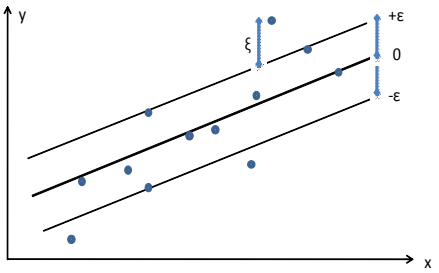

Fig. 3: Example for Support Vector Regression. A tube with radius ϵ is learned to represent the real target function. Possible outliers are being regularized considering a positive slack variable ξ.

Support Vector Machines transform the input space of the training data to a higher-order space using a non-linear mapping, e.g. a radial basis function. In this higher order space a linear function is learned, which has at most ϵ deviation from the real values in general and outliers are regularized by means of the parameter ξ. An example for Support Vector Regression and its parameters ϵ and ξ is depicted in Fig. 3. The transformation to a higher-order space by a non-linear kernel function with parameter γ is not shown here. For a more detailed description of SVM we therefore refer to Smola and Schölkopf [16].

2.6 Preprocessing for Time Series Modeling

In an accompanying work [11] we have analyzed the effects of different preprocessing methods for the stormwater problem. Time series prediction models can benefit from preprocessing operators which generate new features based on the input data. It turns out that best results were obtained with the following preprocessing ingredients:

– Embedding of the input data [10]. Here, the fill level of stormwater overflow tanks $l(t)$ can be represented by a function F on past input features, more

precisely by the rainfall $r(t)$ up to $r(t-W)$, where t indicates time and $W \in \mathbb{N}^+$ is the embedding dimension:

$$l(t) := F(r(t), r(t-1), r(t-2), ..., r(t-W)) \tag{2}$$

- Leaky integration of rainfall (termed *leaky rain*) which is computed as follows:

$$L(t) = \sum_{i=0}^{T} \left(\lambda^i \cdot r(t-i)\right) \tag{3}$$

  where $\lambda \in [0,1]$ and $T \in \mathbb{N}^+$ is the length of the integration window. Best results with SVM and leaky rain preprocessing can be obtained with two different leaky rain functions using two different values λ_1, λ_2 and T_1, T_2, resp (see Fig. 4).
- More details on the preprocessing are given in [11]. Sequential Parameter Optimization (SPOT) was used to tune the SVM and the preprocessing parameters.

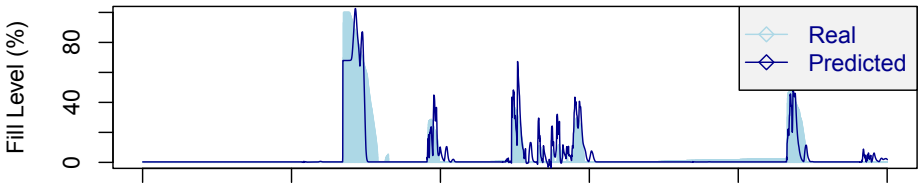

Fig. 4: Prediction for stormwater set 4: the dark line represents the predicted fill level of a SVM model against the shaded area as real fill level. The result was obtained by the best SPOT-tuned SVM model using two leaky rain kernels.

The results of the SPOT tuning are shown in Tab. 2 and Tab. 3. It might be surprising at first sight that the RMSE for Set 2, the data set on which each SVM was trained, is considerably larger than for all other test sets. This is in contrast to normal overfitting where one would expect the training set error to be lower than all other errors. But here Set 2 (see Fig. 2) is considerably more difficult than all other data sets since it contains long time intervals with 100% fill level which are difficult to reproduce from the observed rainfall alone for any predictive model.

In Tab. 3 the evaluation of the SPOT-tuned SVM model, a naïve predictor (predicting the mean value of the training set) and the special-purpose model INT2 on data sets 1-4 is shown. The worst result is obtained with the naïve prediction, which is no surprise, because no learning is included in this approach. *SVM (default)* gives mediocre results, while *SVM (SPOT-tuned)* is comparable to INT2. Tuning was done using the RMSE on Set 4. As a first conclusion drawn out of these results it seems to be possible to achieve competitive results to special-purpose models using our tuned Support Vector Regression model for the fill level problem.

Table 2: Best SPOT parameter configuration for all tuned parameters evaluated on set 4. The region of interest (ROI) bounds are the final values after preliminary runs.

Type	Parameter	Best Value found	ROI	Remark
Embedding	W_1	39	$[2, 60]$	embed. dimension 1
	W_2	16	$[2, 60]$	embed. dimension 2
	T_1	114	$[50, 120]$	leaky window size 1
	T_2	102	$[50, 120]$	leaky window size 2
	λ_1	0.0250092	$[0.00001, 0.3]$	leaky decay 1
	λ_2	0.225002	$[0.00001, 0.3]$	leaky decay 2
SVM	γ	0.0116667	$[0.005, 0.3]$	RBF kernel width
	ϵ	0.0116667	$[0.005, 0.3]$	ϵ-insensitive loss fct.
	χ	1.25	$[0, 10]$	penalty term

Table 3: Evaluation of all models on time windows 1-4. Shown are the RMSE values, when trained on set 2. *SVM (default)* means results with default SVM kernel parameters, while *SVM (SPOT-tuned)* represents mean values obtained in five runs of SPOT.

Model type	Set 1	Set 2	Set 3	Set 4
Naïve prediction	33.56	42.39	34.63	33.34
INT2 (hand-tuned)	7.74	24.27	12.39	10.61
SVM (default)	11.81	44.82	18.55	14.33
SVM (SPOT-tuned)	10.45	43.92	12.93	7.69

3 Experiments: The Effects of Tuning and Selection

The experimental analysis here and in [11] is based on the radial basis SVM kernel from the *e1071* SVM-implementation in R, since we achieved best results with this kernel choice.

All SVM models obtained in this work are sensitive to the parameters for preprocessing and modeling. It has been shown in [11] that a too large number of irrelevant input features can lead to significantly worse results. Although the input features have been extended first by leaky rain embedding and then tuned by a model-based parameter optimization technique (SPOT), it is not clear if

1. the model is generic enough to perform well on different test data and
2. all input features determined by SPOT are really relevant to achieve a good prediction accuracy.

For clarifying the first point, we investigate if our models lead to misleading fits when we evaluate them on different test sets. A misleading model in machine learning can be characterized by the terms *oversearching* and *overfitting* [15]. Oversearching occurs when the learned concept is misleading due to its tailoring to the training goal caused by a too extensive search. In contrast to the well-known concept of overfitting, an oversearched model must not necessarily be "overcomplex", but rather misleading due to a too extensive search for the best hyperparameters, i.e. model and preprocessing parameters.

The second point can be clarified by a consequent feature selection mechanism, that creates subsets of input features which are used to build models and then evaluated. Here we perform a model optimization by GA feature selection, where a binary string defines the feature set of the embedded leaky rain input features.

The following list of experiments describes our route for a critical investigation of issues related to overfitting and oversearching caused by parameter tuning and feature selection:

T1 Parameter tuning using SPOT for SVM parameters and preprocessing parameters, where the objective function for SPOT is evaluated on different validation sets;
T2 Feature Selection by Genetic Algorithms applied to the optimal parameter configurations of **T1**;
T3 Final parameter tuning using SPOT for parameter configuration obtained on the reduced feature set from step **T2**;

3.1 T1: Oversearching by SPOT

In our last models we used the prediction error gathered on the test data set as the objective function value for the hyperparameter tuning with SPOT. In the real world this value is unknown and when available during optimization it adds unrealistic benefits to the tuned model. In order to perform a fair comparison and to show the benefits of parameter tuning in a more realistic setting, we should use a different objective function. Otherwise the test set error might be too optimistic due the model being tuned and tested on the same set. In Tab. 4, we present the mean results of five SPOT runs for the SVR model to determine optimal parameter settings which are then alternately evaluated on the sets 1,3,4. Again, data set 2 has been used for training. The best configuration found by SPOT is then applied in turn to the other sets (columns) resulting in 3 RMSE values for each parameter configuration. We used the Matlab version of SPOT allowing a total budget of 200 SVM models to be built and a maximum number of 500 samples in the metamodel.

A good indicator for oversearching is when best values are often present in the diagonal of the table. It can be seen that this is the case for all validation sets of Tab. 4. Besides this, standard deviations of the offdiagonal values are also larger than the values on the diagonal.

We quantify the oversearching effect by evaluating the following formula: let R_{vt} denote the RMSE for row v and column t of Tab. 4. We define

$$V_t = \frac{S_t - R_{tt}}{R_{tt}} \quad \text{with} \quad S_t = \frac{1}{3}\left(\sum_{v=1}^{4} R_{vt} - R_{tt}\right) \tag{4}$$

With S_t we evaluate the mean off-diagonal RMSE for the columns $t = \{1, 2, 3\}$ which is an indicator of the true strength of the tuned model on independent test data. The diagonal elements R_{tt} are considerably lower in each column of

Table 4: Results of SPOT tuning on the stormwater problem. In each row 1-3 of the table, SPOT tunes the RMSE on validation set 1,3,4 leading to different SPOT-tuned parameter configurations. These configurations were applied to the test sets (columns) to make the results comparable. Each experiment was repeated five times with different seeds and we show the mean RMSE; the numbers in brackets indicate the standard deviations.

		Test		
		Set 1	Set 3	Set 4
Validation	Set 1	**9.11** (0.56)	16.40 (6.42)	12.88 (5.50)
	Set 3	10.82 (1.55)	**12.78**(0.34)	12.36 (3.46)
	Set 4	10.45 (0.28)	12.93 (0.35)	**7.69** (0.48)
	S_t	10.64	14.67	12.62
	V_t	16.7%	14.7%	64.1%

Tab. 4. In case of no oversearching, a value of V_t close to zero would be expected, whereas values larger than zero indicate oversearching.

In summary, a consequent tuning is beneficial but the tuned RMSE is often subject to oversearching effects. E.g. in our case the RMSE on a certain test set was on average 32% higher[1] when the tuned model had not seen the test data before (the realistic case) as compared to the lower value when the test data were used during tuning (T=V).

3.2 T2: Feature Selection

A Genetic Algorithm (GA) is used to determine good feature subsets for the SVM regressor. We rely here on the GA approach because it has some advantages compared to other feature selection methods: Iterative search algorithms can be used to determine feature subsets, where more features are added or eliminated to build the final feature set (Feature Forward Selection and Feature Backward Elimination). Unfortunately these methods often get stuck in locally optimal feature subsets where they finally converge. GAs offer the possibility to flee out of such local optima and find the global optimum given enough iterations.

Experimental Setup In our experimental analysis we started five GA runs, each with a population size of 100, elitist selection strategy (e.g. the best 20% of total population were definitely survivors) and termination after 100 generations. GA parameters where chosen by means of preliminary runs. Each GA individual has N genes, each of which representing whether a certain feature should be included in the model or not. The basis input feature set consisted of all features drawn from a sample SPOT-tuned configuration set as described in Sec. 3.1. Here, the gene length N equals the sum of the embedding dimensions for the two leaky rain functions, ranging from 55 to 92. The candidate solution is mapped to a feature vector which is passed to the feature selection preprocessing script before

[1] average of all V_t in Tab. 4

the SVM model is built. This process has an overall runtime of about 17 hours on a 2.4 GHz Intel Xeon CPU.

Results In each objective function, the RMSE was calculated on the validation sets as defined in Sec. 2.1. This resulted in different feature vectors, which were evaluated again on each validation set. The number of selected features ranges from a minimal feature set of 5 (mean value of GA runs when set 1 was used for evaluation) up to a maximum feature set of 50 (mean value of 5 GA runs). The number of features only vary slightly for runs of the same configuration, but usually differ for different configurations.

The evaluation is presented in Tab. 5. Again it has to be noted, that all configurations seem to suffer from oversearching, when the validation set V (the set on which the GA was performed) is equal to the test set T: the diagonal in Tab. 5 shows always the seemingly best values. Compared with the results gathered by the SPOT tuning (Tab. 4), GA feature selection leads to a slightly better predictive performance if we look at S_t, the mean off-diagonal RMSE.

Table 5: Mean results of five runs using feature selection by genetic algorithms. SVM and preprocessing parameters were obtained using the SPOT configurations 1,3,4 (see Sec. 3.1). The table shows the RMSE values for feature subsets on the validation sets leading to different feature configurations (rows). These configurations were evaluated on the test sets (columns); values in brackets indicate the standard deviation over five runs.

		Test		
		Set 1	Set 3	Set 4
Validation	Set 1	**9.36 (0.11)**	15.48 (0.90)	11.44 (0.91)
	Set 3	10.80 (0.19)	**12.11(0.59)**	7.78 (0.30)
	Set 4	10.99 (0.07)	13.04 (0.04)	**7.36 (0.03)**
S_t		10.90	14.26	9.61
V_t		16.40%	17.75%	30.57%

Even when feature selection does not produce much better results than SPOT tuning alone, it has an obvious positive effect on the RMSE ranges: the standard deviations of the configurations are considerably smaller with feature selection than without, leading to better generalizing models. Also the variance between the three off-diagonal RMSE's is lower than the high off-diagonal variance observed in experiment **T1**. A reason for this might be the complexity decrease of the models due to the lower number of input features. In addition to this, the runtime for model-building is also reduced.

3.3 T3: Final Optimization Using SPOT

One might expect that the best parameter configuration of SVM and embedding parameters has changed with the reduced feature subset found by GA. To check this hypothesis we have conducted SPOT again for two runs of the GA feature

configurations, leading to tuned parameter configurations for the reduced feature sets. Although we observed slight improvements on the validation data set used for tuning, the results got worse on the test sets which indicates overtuning caused by this optimization. As a consequence we claim that the parameter configurations of **T1** are still valid after GA optimization. Nevertheless we cannot exclude that this final step can be important in other applications. We suggest to perform a parameter tuning, as soon as there is a change in the input to the regression model, at least for a few runs.

4 Conclusion and Outlook

We analyzed different predictive models based on Support Vector Machines (SVM) for a practical application named stormwater prediction. The results gathered by the general SVM method are in most cases better than the best-known special-purpose model (INT2). This can be seen as a confirmation of our hypothesis **H1**. It might have a great impact for applications which need a lot of similar models to be built since with our approach most of the time-consuming work of defining and tuning domain-specific models can be replaced by automatic processes.

Our results have also shown that one has to be careful when optimizing data-mining models by means of parameter tuning, e.g. with SPOT and GA: parameter tuning will often lead to oversearching and to too optimistic error estimates on the data sets used for tuning (as measured by V_t in Tab. 4 and Tab. 5), which was the statement of our hypothesis **H2**. Therefore the distinction between validation data sets (used for tuning) and independent test sets is essential to obtain a realistic estimate on the improvement reached by tuning. Nevertheless, our results have shown that tuning leads to better models as measured by independent test set RMSE. Also feature selection led to more stable results in our case study, which indicates better-generalizing models. In a nutshell, feature selection and SPOT tuning can help to improve results, but must always be validated on different test sets to recognize possible overfitting and oversearching effects.

Although the user is satisfied with the current results, we plan to extend and validate our study on other datasets, first by applying our methodology to different stormwater tanks and more comprehensive data (time periods stretching over several years). Also we want to increase the prediction horizon to larger values, enabling longer response times for the effluent control. Furthermore, other time series problems will be investigated, especially problems where no good preprocessing is known yet.

Acknowledgements This work has been supported by the Bundesministerium für Bildung und Forschung (BMBF) under the grants FIWA and SOMA (AiF FKZ 17N2309 and 17N1009, "Ingenieurnachwuchs") and by the Cologne University of Applied Sciences under the research focus grant COSA. We are grateful

to Prof. Dr. Michael Bongards, his research group and to the Aggerverband Gummersbach for discussions and for the stormwater tank data.

References

1. Bartz-Beielstein, T.: SPOT: An R package for automatic and interactive tuning of optimization algorithms by sequential parameter optimization. Tech. Rep. arXiv:1006.4645. CIOP TECHNICAL REPORT 05-10. COLOGNE UNIVERSITY OF APPLIED SCIENCES (Jun 2010), `http://arxiv.org/abs/1006.4645`, comments: Article can be downloaded from: http://arxiv.org/abs/1006.4645. Related software can be downloaded from http://cran.r-project.org/web/packages/SPOT/index.html
2. Bartz-Beielstein, T.: Experimental Research in Evolutionary Computation—The New Experimentalism. Natural Computing Series, Springer, Berlin, Heidelberg, New York (2006)
3. Bartz-Beielstein, T., Zimmer, T., Konen, W.: Parameterselektion für komplexe modellierungsaufgaben der wasserwirtschaft – moderne CI-verfahren zur zeitreihenanalyse. In: Mikut, R., Reischl, M. (eds.) Proc. 18th Workshop Computational Intelligence. pp. 136–150. Universitätsverlag, Karlsruhe (2008)
4. Brockwell, P.J., Davis, R.A.: Time series: theory and methods. Springer Verlag (2009)
5. Chang, C., Lin, C.: IJCNN 2001 challenge: Generalization ability and text decoding. In: Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on Neural Networks. vol. 2 (2001)
6. Drucker, H., Burges, C., Kaufman, L., Smola, A., Vapnik, V.: Support vector regression machines. Advances in neural information processing systems pp. 155–161 (1997)
7. Flasch, O., Bartz-Beielstein, T., Koch, P., Konen, W.: Genetic programming applied to predictive control in environmental engineering. In: Hoffmann, F., Hüllermeier, E. (eds.) Proceedings 19. Workshop Computational Intelligence. pp. 101–113. KIT Scientific Publishing, Karlsruhe (2009)
8. Forrester, A., Sobester, A., Keane, A.: Engineering Design via Surrogate Modelling. Wiley (2008)
9. Hilmer, T.: Water in Society – Integrated Optimisation of Sewerage Systems and Wastewater Treatment Plants with Computational Intelligence Tools. Ph.D. thesis, Open Universiteit Nederland, Heerlen (2008)
10. Kantz, H., Schreiber, T.: Nonlinear time series analysis. Cambridge Univ. Press (2004)
11. Koch, P., Flasch, O., Konen, W., Batz-Beielstein, T.: Predicting fill levels: Generic preprocessing and tuning of support vector regression models. in preparation for `http://arxiv.org` (2010)
12. Konen, W., Zimmer, T., Bartz-Beielstein, T.: Optimierte Modellierung von Füllständen in Regenüberlaufbecken mittels CI-basierter Parameterselektion. at – Automatisierungstechnik 57(3), 155–166 (2009)
13. Mattera, D., Haykin, S.: Support vector machines for dynamic reconstruction of a chaotic system. In: Advances in kernel methods. pp. 211–241. MIT Press (1999)
14. Müller, K., Smola, A., Rätsch, G., Schölkopf, B., Kohlmorgen, J., Vapnik, V.: Predicting time series with support vector machines. Artificial Neural Networks–ICANN'97 pp. 999–1004 (1997)

15. Quinlan, J.R., Cameron-jones, R.M.: Oversearching and layered search in empirical learning. In: In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence. pp. 1019–1024. Morgan Kaufmann (1995)
16. Smola, A., Schölkopf, B.: A tutorial on support vector regression. Statistics and Computing 14(3), 199–222 (2004)