Applied Soft Computing 29 (2015) 357–370

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Efficient multi-criteria optimization on noisy machine learning problems

Patrick Koch[a,*], Tobias Wagner[b], Michael T.M. Emmerich[c], Thomas Bäck[c], Wolfgang Konen[a]

[a] Cologne University of Applied Sciences, Germany
[b] Institut für Spanende Fertigung, TU Dortmund, Germany
[c] Leiden Institute of Advanced Computer Science, Netherlands

ARTICLE INFO

Article history:
Received 18 April 2013
Received in revised form 30 August 2014
Accepted 6 January 2015
Available online 20 January 2015

Keywords:
Machine learning
Multi-criteria optimization
Efficient Global Optimization
Kriging
Hypervolume indicator

ABSTRACT

Recent research revealed that model-assisted parameter tuning can improve the quality of supervised machine learning (ML) models. The tuned models were especially found to generalize better and to be more robust compared to other optimization approaches. However, the advantages of the tuning often came along with high computation times, meaning a real burden for employing tuning algorithms. While the training with a reduced number of patterns can be a solution to this, it is often connected with decreasing model accuracies and increasing instabilities and noise. Hence, we propose a novel approach defined by a two criteria optimization task, where both the runtime and the quality of ML models are optimized. Because the budgets for this optimization task are usually very restricted in ML, the surrogate-assisted Efficient Global Optimization (EGO) algorithm is adapted. In order to cope with noisy experiments, we apply two hypervolume indicator based EGO algorithms with smoothing and re-interpolation of the surrogate models. The techniques do not need replicates. We find that these EGO techniques can outperform traditional approaches such as latin hypercube sampling (LHS), as well as EGO variants with replicates.

1. Introduction

In machine learning (ML), and in particular in classification, the quality of a learning algorithm can be measured by the number of correctly predicted instances on an usually independent set of test instances. Most learning algorithms are highly influenced by the hyperparameter settings.[1] Thus, finding good hyperparameters is essential for training a precise classifier. For solving this problem, we can define and perform a single-criteria optimization (SCO), where the classification error or any other quality indicator is being minimized. Due to the large runtimes of ML algorithms, it has been shown earlier [1–3] that methods like Efficient Global Optimization (EGO) [4] can help to solve the problem, especially when budgets are limited. In EGO the optimization is internally performed on a surrogate model. The expected improvement (EI) [4] deals as *infill criterion* in an sequential optimization process, recommending which point to explore/to evaluate in the ML task.

However, in most cases the user is not only interested in high-quality prediction models, but also wants that model training and parameter tuning is performed in a reasonable amount of time. These objectives are usually conflicting, demanding for the concept of multi-criteria optimization (MCO). Nowadays, EGO-like algorithms are also available for multi-criteria optimization problems [5]. Up to now, this paradigm has received little attention in the ML community, and only little work has been spent to optimize the mentioned objectives at the same time.

One reason for this lies in the nature of ML parameter tuning, where the evaluation of the learning process is usually subject to noise. Besides the usual noise of labeling the ML data, it is mainly caused by the randomness in the selection of training, validation and test data which lead to different results, even for deterministic ML models. A surrogate model has to cope with such noisy responses. The noise in the quality response will increase if we look for solutions with low runtime, which is usually connected with smaller training set sizes.

Summarizing, the typical challenges of noisy MCO can be formulated as follows: (a) finding a good approximation of the Pareto front, (b) coping with the noise and (c) finding a good solution set with very restricted budgets of function evaluations.

1.1. Research questions

We aim at applying multi-criteria EGO variants to solve two criteria ML tasks. Therefore we can define the following research questions:

Q1 Is multi-criteria optimization with surrogate modeling (Kriging) possible for two criteria in the presence of strong noise?

* Corresponding author. Tel.: +49 226181966216.
E-mail addresses: patrick.koch@fh-koeln.de (P. Koch), wagner@isf.maschinenbau.uni-dortmund.de (T. Wagner), emmerich@liacs.nl (M.T.M. Emmerich), baeck@liacs.nl (T. Bäck), wolfgang.konen@fh-koeln.de (W. Konen).

[1] A hyperparameter is a parameter of the learning algorithm which is set to a specific value prior to the learning phase and controls the ability of the learning algorithm to adapt to new data (generalization).

http://dx.doi.org/10.1016/j.asoc.2015.01.005
1568-4946/

Q2 Is it also possible when the budget for function evaluations (i.e. ML training runs) is very restricted?
Q3 Is it necessary to dampen the noise by averaging over repeated function evaluations, at the price of fewer allowed infills under the given budget?
Q4 Are the multi-criteria EGO approaches better in finding good approximation sets than traditional design of experiments (DoE) techniques, e.g., LHS (see Section 2.3)? Are there significant differences between the different EGO approaches?

The paper is structured as follows: in Section 1.2 we highlight related approaches. The basic idea of surrogate-based optimization is described for single-criteria problems in Section 2.1, and followed by a general introduction of MCO in Section 2.2. In Section 3 we describe the setup of the study for efficient two criteria ML experiments. The experimental results are discussed in Section 4 and we give concluding remarks in Section 5.

1.2. Related work

Because most supervised ML models like Support Vector Machines (SVMs) [6,7] are sensitive to their hyperparameter settings, an optimization is required until an optimal behaviour of the models can be guaranteed. This problem became popular as *model selection* [8,9]. Often hyperparameters of models like SVMs are set by grid search or local search heuristics. E.g., Chapelle et al. [10] proposed an approach based on gradient descent, while Keerthi et al. [11] tuned parameters using a BFGS optimizer. Later, Keerthi et al. [12] also showed that hyperparameter optimization can be efficiently done with BFGS even for large-scale problems. Instead, global optimization heuristics were firstly proposed by Cohen et al. [13], who used Genetic Algorithms (GA) for selecting the best SVM model. Friedrichs and Igel [14] later optimized SVM hyperparameters with the CMA-ES [15,16] and demonstrated a superior performance compared with grid search. Also, Glasmachers and Igel [17,18] presented an improved approach for general Gaussian kernels and handling uncertainty.

Although global optimization methods like Evolutionary Algorithms are suitable for many different problems, they often suffer from requiring too many objective function calls. Alternative approaches are known as response surface methodology (RSM) [19] or model-assisted optimization. In model-assisted optimization a surrogate model of the objective function is learned during the optimization process, which can be used to replace evaluations on the real expensive function. Model-assisted optimization has received a lot of interest with the integration of Kriging surrogate models in the context of design and analysis of computer experiments (DACE) [20]. An overview about surrogate models in Evolutionary Computation has been given by Jin [21]. Kriging surrogate models for reducing the number of function evaluations in Evolutionary Computation have been proposed by Ratle [22], Emmerich et al. [23] and Zhou et al. [24]. Lim et al. [25] describe a generalized evolutionary framework using ensembles and smoothing of surrogate models to generate reliable fitness approximations. In their approach they compare Kriging, polynomial regression and radial basis functions.

In case of noisy evaluations, there is a large body of work on Noisy Kriging-based Optimization (NKO) for single-criteria optimization tasks. Forrester et al. [26] extended the deterministic DACE [20] method to noisy experiments using a method named re-interpolation, which is also used in this work. In the same year, Huang et al. [27] proposed an approach based on the so-called Augmented Expected Improvement (AEI) criterion for noisy evaluations. Picheny et al. [28] introduced the quantile-expected improvement and an online computation time allocation method. A comprehensive overview about these approaches is given in [29]. In this article a benchmark of several NKO-approaches on a variety of well-known hard optimization problems with a steerable amount of additive noise has been performed. In (noisy) ML parameter tuning Konen et al. [3] showed that a model-assisted tuning using Kriging performs better on a set of benchmark problems than other state-of-the-art optimization heuristics. On basis of these results Koch and Konen [2] discovered that tuning with small fractions of the available training data can lead to good parameter settings. But any a-priori setting of the training set size without special problem knowledge remained virtually impossible.

A solution to this issue can be to explore a set of solutions, representing alternatives between small and large training set sizes, so that a suitable size is finally delivered to the user. As a necessity, this approach requires the definition of multiple objectives. In earlier ML research, MCO was firstly proposed by Liu and Kadirkamanathan [30]. They optimized a radial basis function network, where two objectives functions were considered to optimize the differences between the real non-linear system and the non-linear model, and another function to emphasize on simpler models. Freitas [31] and Jin and Sendhoff [32] give comprehensive reviews about the employment of multi-criteria algorithms in ML. Jin [33] advocates to use Pareto-based approaches, covering both supervised and unsupervised learning. For MCO often set-based approaches based on evolutionary multi-objective algorithms (EMOA) are proposed. As a drawback, the required number of real function evaluations is considerably higher for these algorithms. This can be problematic, because the computation time is usually very limited. Instead Knowles and Nakayama [34] discuss the use of surrogate-modeling techniques also for multi-criteria optimization problems. The first EMOA using surrogate models was proposed by Giannakoglou et al. [35], whereas Emmerich and Naujoks [36] introduced Kriging models that make use of error prediction to EMOA. Ascia et al. [37] performed an extensive comparison of state-of-the-art EMOA with an approach based on fuzzy approximation to speed up evaluations. A popular variant of EGO for multi-criteria optimization was given by Knowles [38], the Pareto-EGO (Par-EGO). Later, EGO approaches using the hypervolume as infill criterion were introduced, e.g., the SMS-EGO by Ponweiser et al. [39], or the hypervolume-based EI criterion by Emmerich et al. [40], also referred to as $\mathcal{S}$-metric based Expected Improvement (SExI) [5]. Another approach based on decomposition is the MOEA/D by Zhang et al. [41]. Recently, Zaefferer et al. [42] compared SMS-EMOA, a well-known solver without surrogate modeling, and four cutting-edge multi-criteria solvers with surrogate modeling (SMS-EGO, SExI-EGO, MSPOT and MEI-SPOT) on an optimization task without noise. An overview of the properties of multi-criteria EGO variants was given by Wagner et al. [5]. As an alternative to Kriging-based methods some authors propose topology-based methods using Delaunay regression or SOM to capture the topology of the underlying data [43].

Only few authors, see Knowles et al. [44], address the topic of NKO for multi-criteria optimization tasks. The new contribution of the present paper is that we investigate for the first time (i) hypervolume-based expected improvement in surrogate models for MCO, and (ii) MCO for tuning noisy ML problems. We will show that the proposed NKO methods can achieve good solutions with relatively few function evaluations and that they can cope with the specific noise which arises in the field of ML mainly from the subsampling of the training data.

2. Methods

2.1. Efficient single-criteria optimization

In single-criteria optimization we seek an optimal vector $\vec{x}^* \in S \subseteq \mathbb{R}^n$ of a function $f(\vec{x}) \to \mathbb{R}$ which gives the minimal[2] function value f^* for that function:

$$\min_{\vec{x} \in S} f(\vec{x}) = f^* \tag{2.1}$$

Because in our case the evaluation of f includes a complete training and evaluation of a ML model, the direct optimization of f can be expensive. Instead of performing the optimization on f, we fit a surrogate function $\hat{f}$ approximating the real function f. The function is first evaluated at design points $\vec{x}^{(1)}, \ldots, \vec{x}^{(k)}$, in order to train the approximation function $\hat{f}$. We will denote the real evaluations – also called observations – by $y^{(1)}, \ldots, y^{(k)}$. These k observations are used to build a good start point for the following optimization on the model:

$$init_{des} := (\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \ldots, (\vec{x}^{(k)}, y^{(k)}) \tag{2.2}$$

This initial design consisting of k points is usually selected from a point set, which is uniformly distributed over the search space, e.g., by calculating an optimized LHS. Then, a regression function can be fitted based on the initial design. In general, any regression technique can be used, however we prefer to use Kriging surrogate models, which were proposed in the context of Design and Analysis of Computer Experiments (DACE) [20], because Kriging performed best in earlier studies. As another advantage, it can handle more complex fitness landscapes and augments predictions with an estimate of the corresponding uncertainty.

Kriging. Kriging is a regression technique named after the geostatistician Krige [45]; the theory of Kriging was mathematically formalized by Matheron [46]. In Kriging the Best Linear Unbiased Predictor (BLUP) and Kriging variance or uncertainty are used for (error) prediction. This coincides with the conditional mean and variance, whenever the random process is Gaussian [47]. We will proceed here with this Bayesian interpretation also known as Ordinary Kriging (OK) [29]. In a first step the results $\vec{y} = (y_1, \ldots, y_k)^T = f(\vec{x}^{(1)}), \ldots, f(\vec{x}^{(k)})$ of an arbitrary initial design are taken as input to the OK model. For any design point $\vec{x}^{(i)}$ the approximation function

$$Y(\vec{x}^{(i)}) = \mu + Z(\vec{x}^{(i)}) \tag{2.3}$$

[2] Note that for simplicity we always refer to minimization problems in this paper, but maximization problems can easily be transformed to minimization problems. Moreover, we assume the existence of an optimum.

predicts values using an unknown constant trend $\mu \in \mathbb{R}$ and an n-dimensional stochastic Gaussian process $Z(\vec{x})$ with stationary covariance function of the form

$$Cov(Z(\vec{x}^{(s)}), Z(\vec{x}^{(t)})) = \sigma^2 K_\theta(\vec{x}^{(s)}, \vec{x}^{(t)}) \tag{2.4}$$

where σ^2 is the process variance, and K_θ is a correlation function with corresponding kernel parameter θ [48]. In case of noisy observations, the approximation function $Y(\vec{x}^{(i)})$ can be written as realization of random variables $Y(\vec{x}^{(i)}) := Y(\vec{x}^{(i)}) + \epsilon_i$ with measurement errors ϵ_i. To cope with the noise, Kriging conditions Y on the noisy observations $\vec{x}^{(1)}, \ldots, \vec{x}^{(n)}$.

Similar to other kernel methods (see Appendix C), any correlation function can be considered such as the Gaussian kernel function:

$$K_\theta(\vec{x}^{(s)}, \vec{x}^{(t)}) = \exp(-\theta||\vec{x}^{(s)} - \vec{x}^{(t)}||^2) \tag{2.5}$$

For simplicity we denote by

$$\vec{K}_\theta(\vec{x}) = \left(K_\theta(\vec{x}, \vec{x}^{(1)}), \ldots, K_\theta(\vec{x}, \vec{x}^{(k)})\right)^T \tag{2.6}$$

the correlations between $\vec{x}$ and the earlier evaluated design points $\vec{x}^{(1)}, \ldots, \vec{x}^{(k)}$.

Under standard assumptions[3] the Kriging model can be written in terms of conditional expectation and variance of $\hat{f}$ knowing the observations:

$$M(\vec{x}) = \mathbb{E}[Y(\vec{x}) \mid Y(\vec{x}^{(i)}) = y_i, 1 \le i \le k] \tag{2.7}$$

$$s^2(\vec{x}) = Var[Y(\vec{x}) \mid Y(\vec{x}^{(i)}) = y_i, 1 \le i \le k] \tag{2.8}$$

Having evaluated the first k design points, the mean can be written as

$$M_k(\vec{x}) = \hat{\mu}_k + \vec{K}_\theta(\vec{x})^T \mathbf{K}_k^{-1}(\vec{y} - \hat{\mu}_k \mathbf{1}_k) \tag{2.9}$$

with:

- $\mathbf{K}_k = (K_\theta(\vec{x}^i, \vec{x}^j))_{1 \le i,j \le k}$
- $\hat{\mu}_k = \mathbf{1}_k^T \mathbf{K}_k^{-1} \vec{y}_k / \mathbf{1}_k^T \mathbf{K}_k^{-1} \mathbf{1}_k$
- $\mathbf{1}_k$ is a $k \times 1$ vector of ones

Based on the correlations $\vec{K}_\theta$ and the process variance σ^2 the uncertainty of the prediction can be measured by:

$$\hat{s}_n(\vec{x}) = \sqrt{\sigma^2 - \vec{K}_\theta(\vec{x})^T \mathbf{K}_n^{-1} \vec{K}_\theta(\vec{x}) + \frac{(1 - \mathbf{1}_n^T \mathbf{K}_n \vec{K}_\theta(\vec{x}))^2}{\mathbf{1}_n^T \mathbf{K}_n^{-1} \mathbf{1}}} \tag{2.10}$$

The Kriging equations defined in Eqs. (2.9) and (2.10) are called the regressing model. Herewith we consider uncertainties of the objectives independently from each other and do not use correlations between the two objectives in the variance function. Svenson [49] has derived the following statements in that regard:

1 "In many examples, the dependence model actually performed worse, on average, than the independence model for several improvement functions."
2 "A general recommendation is to stick to with the independence model."

Although it is obvious that correlations between the objectives exist, the independence assumption is accepted throughout this paper in line with these results. An equivalent formulation for noisy Kriging based on variograms has been proposed by Sakata and Ashida [50].

[3] We assume that μ is independent of Z and uniformly but randomly distributed over $\mathbb{R}$.

With a Kriging model as a surrogate for the real objective function f, we can perform an optimization on this surrogate model. This can be done by generating candidate points for which an evaluation on the surrogate model is performed. New promising points are then evaluated on the real function, and thereafter the surrogate model is updated with the new information. Often the point considered for evaluation on the real function is selected using the EI infill criterion proposed by Mockus et al. [51] and introduced to DACE by Jones et al. [4] which is defined by the following function:

$$EI(\vec{x}) = E[\max(f_{\min} - Y(\vec{x}), 0)] \tag{2.11}$$

where f_{min} is the minimum of all previously obtained real evaluations, and $Y(\vec{x})$ is the random variable predicted by the Kriging surrogate model, which is usually assumed to be normally distributed with mean $M_k(\vec{x})$ and variance $\hat{s}_k^2(\vec{x})$. Jones et al. have shown in their seminal paper [4] that under this assumption the expected improvement can be calculated in closed form

$$EI_k(\vec{x}) = (f_{\min} - M_k(\vec{x}))\Phi\left(\frac{f_{\min} - M_k y(\vec{x})}{\hat{s}_k(\vec{x})}\right) + \hat{s}_k(\vec{x})\rho\left(\frac{f_{\min} - M_k y(\vec{x})}{\hat{s}(\vec{x})}\right) \tag{2.12}$$

In the above Φ and ρ are the cumulative density and the probability density of the standard normal distribution, respectively [4].

The EI deals as infill criterion, that is by maximizing the EI, new promising candidate points can be determined. While interpolating surrogate-models like Kriging assume uncertainties in undiscovered regions of the search space, an uncertainty of zero is inserted at evaluated points. Thereby the EI can support the exploration, because high variances in unknown regions can lead to points with a better $\hat{y}$ promising as the next infill candidate. A more detailed study on infill criteria has been given by Picheny et al. [29].

2.2. Multi-criteria optimization

In multi-criteria optimization, multiple objective functions are optimized in parallel:

$$\min_{\vec{x} \in S}(f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x})) \tag{2.13}$$

Note that the individual objective functions $f_1, \ldots, f_m$ can be conflicting, so that the sets of the optimal solutions for the functions cannot be unique. We say a point $\vec{x}_1$ *dominates* a point $\vec{x}_2$, if all functions values of $\vec{x}_1$ are smaller or equal than those of the corresponding objective function values of $\vec{x}_2$, and at least one objective function value of $\vec{x}_1$ is strictly better:

$$\vec{x}_1 \prec \vec{x}_2 \tag{2.14}$$

A point x^* is considered to be Pareto-optimal, if there does not exist any feasible point $\vec{x} \in S$ which would decrease one objective function value without a simultaneous increase in any other objective function value.

In the last years, set-based approaches have been established to approximate the Pareto-optimal front [52], and here especially evolutionary multi-objective algorithms (EMOA) have become popular methods in practice. The advantage of set-based approaches is that a group of points distributed over the Pareto front is approximated within a single run of the algorithm, while in single-solution approaches only a single trade-off between the objectives is returned in the end.

2.3. Model-assisted multi-criteria optimization

The most important weakness of the usage of evolutionary algorithms is probably that they suffer from large number of function

calls required for converging to good regions of the search space. Especially in multi-criteria optimization this is a clear disadvantage and can result in prohibitive computation times. A frequently shared paradigm in this case is to use surrogate-functions which model the real fitness landscape and can help in performing optimization runs with much fewer real function evaluations [34].

In MCO, surrogate models must be learned for each objective. This set of surrogate models is then used for the prediction of promising new design points using EI definitions for multiple criteria [5].

We are using the following hypervolume-based EGO methods, that are currently popular in MCO:

SMS-EGO ($\mathcal{S}$-Metric Selection EGO by Ponweiser et al. [39]): This EGO method uses an infill criterion based on the dominated hypervolume where the important part of the improvement function estimates the hypervolume increase due to a potential solution.

SExI-EGO ($\mathcal{S}$-Metric Expected Improvement EGO by Emmerich et al. [40]): This EGO method uses another infill criterion also based on the dominated hypervolume where for each point with objective values and predicted variances given by the surrogate models the exact computation of the expected improvement in the hypervolume is done. This indicator has recently been used in design optimization under the name "Hypervolume-based Expected Improvement" (HEI), cf. [53,54].

The hypervolume indicator or the $\mathcal{S}$-metric is defined as the Lebesgue measure of the subspace that is dominated by the approximation set. To make this measure finite, the subspace is cut from above by a reference point r. The reference point is chosen in such a way that all points in the approximation set dominate it. The $\mathcal{S}$-metric (or hypervolume) is a scalar indicator for measuring performance of an approximation set to the Pareto front. As such it takes into account both, diversity and convergence of the approximation set to the Pareto front. The spread of approximation sets that optimize the $\mathcal{S}$-metric was investigated by Fleischer [55] for discrete problems and Auger et al. [56] for finite point sets on continuous Pareto fronts. The hypervolume indicator prefers approximation sets which are spread along the Pareto front, thus maximizing the hypervolume inherently avoids the clustering of non-dominated solutions at the Pareto front. Maximization of the $\mathcal{S}$-Metric on continuous Pareto fronts leads to regularly spaced Pareto front approximations, where the spacing on the Pareto front depends on the deviation of the slope from a 45° angle. Increasing the number of points in the approximation sets yields increasingly dense covers of the Pareto front and for compact Pareto fronts lead in the limit to gap free approximations.

Since the $\mathcal{S}$-metric indicates good theoretical properties in [5], we consider both the $\mathcal{S}$-metric Selection-EGO (SMS-EGO) and the $\mathcal{S}$-Metric Expected Improvement-EGO (SExI-EGO) as model-assisted algorithms for our experiments on multi-criteria optimization of ML models.

We compare these two EGO-approaches with latin hypercube sampling as a baseline. For LHS the full budget of real function evaluations are spent to build an competing optimal LHS in the region of interest. The set of non-dominated solutions among the LHS points is determined and its hypervolume calculated.

2.4. Noise handling

Model evaluations in ML are usually subject to noise. Although models like SVM are deterministic, the random sampling of data leads to different model accuracies. This can lead to wrong decisions during tuning, because the returned objective function values are unstable and have variances significantly larger than zero.

2.4.1. Replicates

The usual procedure to cope with noise in DoE is to use replicated measurements and to calculate an aggregated value, e.g., the mean of all replicates. This will reduce the noise level and may avoid wrong tuning decisions. But the price for replicated measurements under a limited budget is that fewer infill points can be generated: e.g., if each design point is evaluated three times, we have only one third of the number of infills. We investigate below whether replicates have a beneficial effect on the overall quality.

In case of interpolating Kriging, it is necessary that replicated designs are aggregated. Here, standard aggregation procedures like calculating the mean of a series of evaluations can be taken as value for an interpolating Kriging model. Instead Kriging regression estimates the confidence intervals, whereas it is valuable to pass the replicates to the Kriging regression model to refine the uncertainty estimation of the model.

2.4.2. Re-interpolation

An elegant method for noisy Kriging optimization (NKO) is the re-interpolation (RI) method by Forrester et al. [26]. At first, m non-interpolating Kriging models are created for all m objective functions using the actual design in the ith step (des_i). Then the actual design is re-interpolated to induce m *interpolating* Kriging models. The new interpolating Kriging models are finally used to optimize the EI infill criterion as usual. The corresponding pseudocode of the RI procedure is shown in Algorithm 1.

Algorithm 1. Re-interpolation procedure for noisy Kriging optimization

Set initial design $des_i := \{\vec{x}^{(1)}, \vec{x}^{(2)}, \ldots, \vec{x}^{(k)}\}$
Fit non-interpolating model $mod:1$ using nugget estimation for design des_i
Evaluate des_i using $mod:1$
Fit interpolating model $mod:2$ for predicted responses from previous step
Maximize expected improvement on all $mod:2$

3. Experimental analysis

We performed an experimental study using two objectives f_1 and f_2 for parameter tuning in ML. In a preliminary experiment we compared the algorithms on two synthetic test functions ZDT1 and ZDT2. Finally, in the ML experiments, we measured the *classification gain* for the first objective and the *training time* of the learning algorithm for the second objective. It is very likely that these objectives are conflicting: the runtime increases, when more data is used for the learning. However, the classification result should improve in this case.

Different variants of model-assisted MCO algorithms are analyzed to measure the performance of the EGO approach. All algorithms are implemented in Matlab version 7.12.0.635 (R2011a) and Kriging is used together with different EI criteria based on the hypervolume. The learning itself was implemented in R using the TDMR framework [57]. A flow chart of the tuning process is displayed in Fig. 1.

3.1. Problem definition

The goal of the following experiments is to show how two criteria NKO works for selected ML tasks. Given a certain classification problem, the result of the experiment shall be an approximation to the Pareto front for objectives f_1 and f_2. This approximation is presented to the ML-user and allows him to decide about the right balance between classification accuracy and ML runtime.

For the ML tasks, we used SVM as learning algorithm and the available data were split prior to tuning in 20% test data and 80% for training-validation data. The tuning goal is to find the best

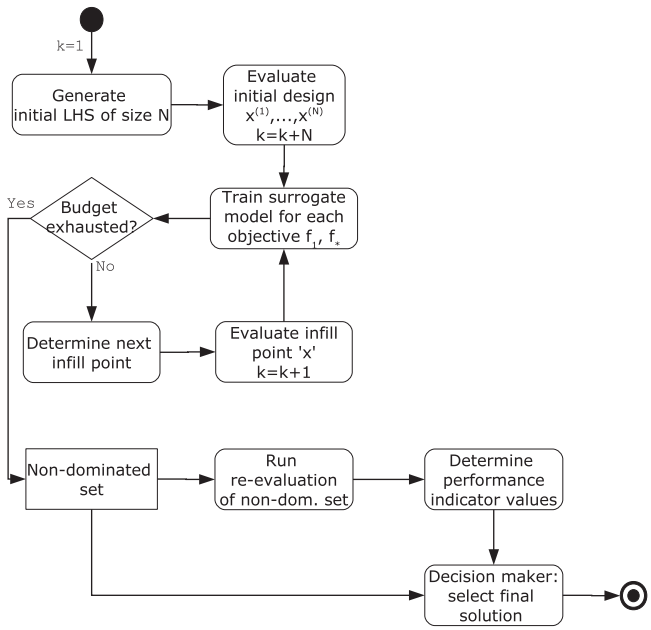

Fig. 1. Flow chart of the two criteria optimization process. At first, all algorithm variants create an initial parameter design based on a LHS. This design is evaluated on the real objective function (e.g., the learning algorithm, performing a complete training and a prediction of the SVM using parameters obtained from the LHS setting). After that, m surrogate functions are trained using results from the objective functions as information. Based on these surrogate models, a new sequential design point (infill point) is determined (e.g., via the expected improvement criterion). This point is as well evaluated on the real objective function and the result is used to refine the surrogate models. This sequential procedure is repeated until the computational budget is exhausted. Finally, the non-dominated set is returned and re-evaluated, to obtain an unbiased solution set. The performance of the re-evaluated non-dominated set is used for measuring the performance indicators (hypervolume and R2).

parameters (design variables). The set of design variables is task-dependent (and defined in detail below in Section 3.2 for each task), but for the real-world problems we always consider the parameter *trnFrac*. The tuning parameter *trnFrac* defines the fraction of records from the train-validation-set used solely for training. The remaining records are used for validation purposes. The objective *classification gain* is always evaluated on the independent test set (20%).

The noise observed in our experiments has its main source in the subsampling procedure. A smaller *trnFrac* will result in a higher noise level. We want to show with our experiments how well different multi-criteria NKO methods can cope with this form of noise and which variant works best. The goal is to get with a limited number of function evaluations a good approximation of the 'best' Pareto front (details are explained in Section 3.3).

3.2. Experimental setup

As initial design we always used a LHS consisting of n_{init} parameter design points, which were evaluated on the real objective function. In the preliminary experiments on synthetic test functions we spent $n_{init} = 20$ evaluations for the 3-dimensional ZDT2 function, and $n_{init} = 50$ evaluations for the 8-dimensional ZDT1 function. In case of the real-world problems we likewise spent $n_{init} = 20$ evaluations for the Sonar dataset from the UCI library and $n_{init} = 50$ evaluations for an engineering problem named AppAcid. For Sonar 130 sequential steps were performed, while we spent 150 evaluations for the AppAcid problem. The maximum number of surrogate model evaluations was restricted to 50,000 evaluations. As an additional stopping criterion we forced the algorithms to stop when the infill function value does not give more than $\varepsilon = 10^{-7}$ improvement. This restriction was necessary, because although evaluations on the surrogate model are usually cheap, some EGO implementations tend to calculate the dominated hypervolume very often, which can be time-consuming also for small objective dimensions m.

The SVM implementation was taken from the *e1071* R package. As kernel for the SVM we chose the radial basis kernel. It is known that this learning algorithm is sensitive to the parameter settings of the kernel parameter γ. For the Sonar benchmark we tuned γ together with the parameter *trnFrac* as another tuning parameter, having a somehow comparable setup as in [2]. For the more difficult AppAcid problem, we tuned 7 parameters. Detailed information about the region of interest and the parameters for AppAcid can be found in Table B1 We did not tune the SVM regularization parameter C for these two datasets because in earlier experiments [2] the model was rather insensitive to its settings.

We employed two objectives for the problems consisting of $f_1 = gain\ on\ test\ set$ and $f_2 = training\ time$. The parameter *trnFrac* should directly affect objective f_2, while the other parameters should mainly affect function f_1. The region of interest (RoI) was chosen according to settings from preliminary runs and the settings used in [3]. Since the parameters partly affect functions f_1 and f_2, it is of interest, if the optimization methods are capable to learn the parameter relevances.

All methods employed seek for the Pareto optimal set (SMS-EGO, SExI-EGO or LHS). For every recommended parameter setting the learning algorithm is called. Thus in each call the learning algorithm takes as input a subset of the data for building a prediction model (both for Sonar and AppAcid) and the hyperparameters set by the optimizer. For each algorithm, ten runs were performed with different randomly drawn test and training-validation data samples to get statistically sound results.

3.3. Benchmarking

Benchmarking MCO algorithms usually includes the distance of the approximation set to the real Pareto front and the spread of the points over the front. Since the real Pareto front is unknown in our case, we try to introduce a fair benchmarking which nevertheless respects both performance measures.

At first we validate the performance of the non-dominated set of each algorithm variant by re-evaluating these solutions ten times. Then we select the non-dominated subset from this re-evaluated set of all algorithms. We call this resulting set the *reference set*. The distance of the non-dominated sets of each run and this reference set then describes the approximation quality. In the best case an algorithm would produce exactly this reference set, so that the total distance would become zero.

Now we only have to define how the distance between the single algorithm runs and the reference set are measured. One option is the difference in hypervolumes (HV). Since HV and other quality indicators are controversially discussed in MCO, we additionally incorporate the R_2 indicator by Hansen and Jaszkiewicz [58] (also termed R2 in the following), which is described in Appendix D.

3.4. Synthetic test functions

We analyze the behaviour of the hypervolume-based expected improvement infill criterion on two synthetic test functions, ZDT1 and ZDT2, from the benchmark suite by Zitzler et al. [59]. For both problems the true Pareto front is known, which enables a comparison of the solution set of the tuning algorithm and the real Pareto optimal set. Because originally both problems are deterministic functions, we added a normal distributed noise term $N(0, \sigma)$ with a fixed value of $\sigma = 0.1$.

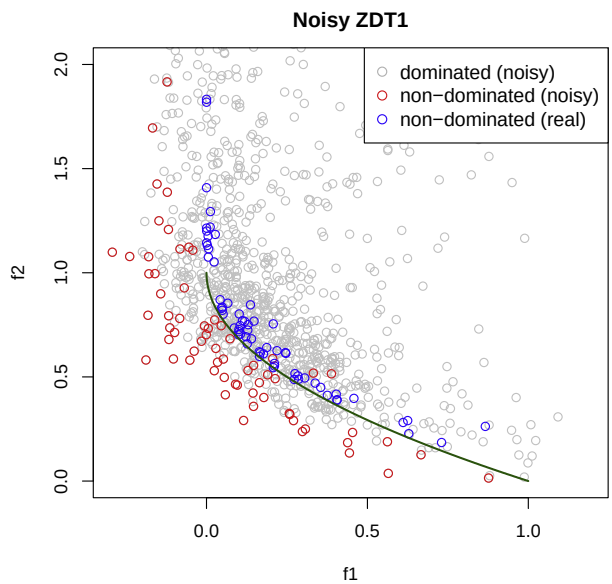

Fig. 2. Pareto plot for the 8-dimensional noisy ZDT1 function. The plot shows the solutions of ten runs using the re-interpolation approach by Forrester et al. with the SMS infill criterion. In total, we spent 200 function evaluations for the tuning, with 50 evaluations given for the initial LHS.

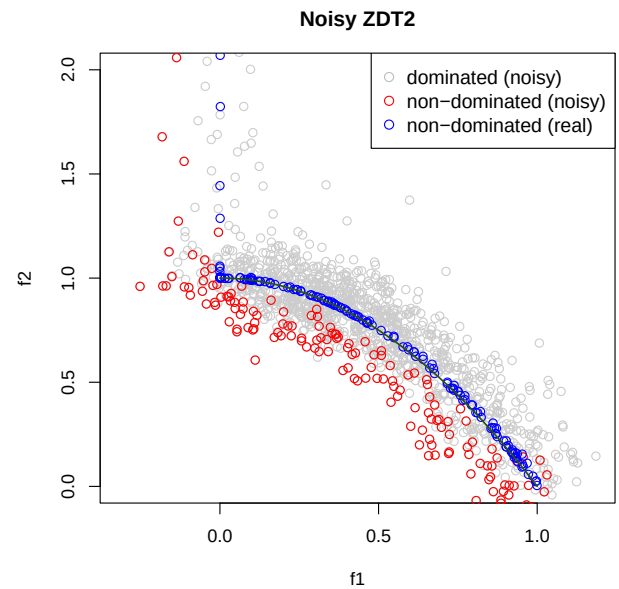

Fig. 3. Pareto plot for the 3-dimensional, noisy ZDT2 function. The plot shows the solutions of ten runs using the re-interpolation by Forrester et al. with the SMS infill criterion. We spent 150 function evaluations for the tuning, where 20 evaluations were given to the initial LHS.

3.4.1. ZDT1

ZDT1 is a two-criteria test function with a convex shaped Pareto front. The RoI for each parameter is given by the interval [0, 1]. The number of dimensions is scalable from at least two parameters, making it possible to de- or increase the complexity of the problem. In this experiment, we chose 8 dimensions to be comparable with the AppAcid application described later in this section. Besides that, the artificial noise term added to each objective makes the complexity of the problem more related to the noise in ML. As a consequence, the noisy objective function values can dominate the real Pareto front, which is given by $(x_1, 1-\sqrt{x_1})^T$ with $x_1 \in [0, 1]$.

In Fig. 2 we present the results of ten independent runs on ZDT1. The grey points show the solutions on the noisy problem (ZDT1 with additive noise term) in the objective space. The red points depict the final noisy non-dominated solution set of each run. Note that in the plot a non-dominated solution from the ith run can dominate a solution from the jth run, with $i \neq j$. This causes some red points to be dominated by other solutions. The blue solutions again depict the non-dominated solutions (red points), after an evaluation on the non-noisy ZDT1.

It can be seen from the plot that the real Pareto front (green curve) can be attained by the model-assisted optimization. Although the majority of the non-dominated solutions (red points) are visibly biased by the noise term (they even seem to dominate the true Pareto front), they are close to the real Pareto front, after the noise is subtracted (blue points). Therefore, we can conclude that the additive noise term does not lead to a remarkable deterioration of the solution set.

A negative effect of the additive noise term becomes visible in the plot by an increased density of solutions in the left region ($f_1 \leq 0.5$). This is due to the rather simple optimization of the first objective (only the first parameter x_1 affects objective f_1, while f_2 depends on all other parameter settings) together with the additive noise for this objective. In this specific case (negative) noise can easily lead to a dominance of the real Pareto optimal solutions.

3.4.2. ZDT2

Similar to ZDT1, ZDT2 is a two-criteria test function from the benchmark suite by Zitzler et al. [59]. Again the RoI is defined by the interval [0, 1] for each parameter $x_i \in \vec{x}$. The real Pareto front is analytically given by $(x_1, 1-x_1^2)^T$ with $x_1 \in [0, 1]$. We present the results for ZDT2 as a plot in the solution space in Fig. 3. Again the real Pareto front could be attained by solutions, although the additive noise term was present during the optimization. The distribution of solutions over the front is better than for ZDT1, which is probably due to the lower dimension considered for ZDT2 (3 compared with 8 for ZDT1).

Although the concentration of solutions is slightly increased again in the left part of the plot, this effect is not equally visible as in the case of the higher-dimensional ZDT1 function. A possible reason for this is the lower dimensionality of the ZDT2 function in our experiment, making the complexity of the two objectives more comparable to each other. We think that without the artificial additive noise term, or with a relative noise term it is very likely, that the observed effect would not be visible at all.

3.5. Sonar

Because Sonar is a small dataset with only 208 records, a parameter tuning can be performed quickly to test the performance of tuning algorithms. Sonar is a binary classification problem and therefore ideally suited to be solved with SVM. The data itself can be downloaded from the UCI website.[4] For simplicity we performed a tuning with only two parameters and spent 150 evaluations for the tuning. As input parameters *gamma* and *trnFrac* were defined as described in Table A1, parameter *gamma* defines the bandwidth for the SVM classifier, and *trnFrac*, sets the training set size used for the model training (random subsampling). Both parameters are varied in the tuning method and passed to the machine learning process, which is initiated in the objective function.

In Fig. 4 we show the solution space (the first axis shows the (negative) prediction accuracy of the SVM, and the second axis the runtime of the learning process) of ten independent runs of SMS-EGO on the Sonar dataset. In the upper plot ($r = 1$: one evaluation

[4] http://archive.ics.uci.edu/ml/datasets.html

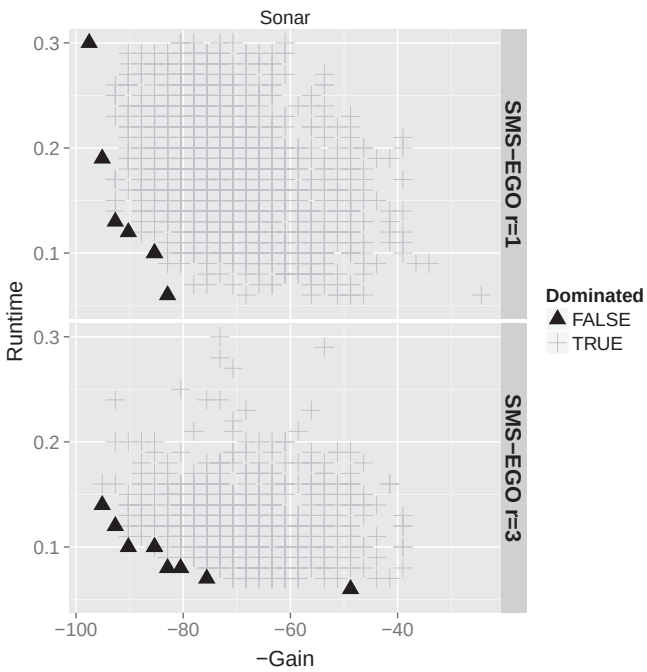

Fig. 4. Tuning results with 150 real function evaluations on the Sonar dataset. In the plot the aggregated solutions of ten runs in the objective space are displayed. The initial design size (LHS) was 20. Only a single evaluation was performed in the first experiment (top), while we spent three repeated evaluations for the second experiment (bottom). SMS-EGO was run using a Gaussian kernel together with re-interpolation. All solutions were combined and a non-dominated filtering was applied to these points. The triangles represent the non-dominated solutions of this combined filtering, while the crosses represent the dominated points of the combined filtering.

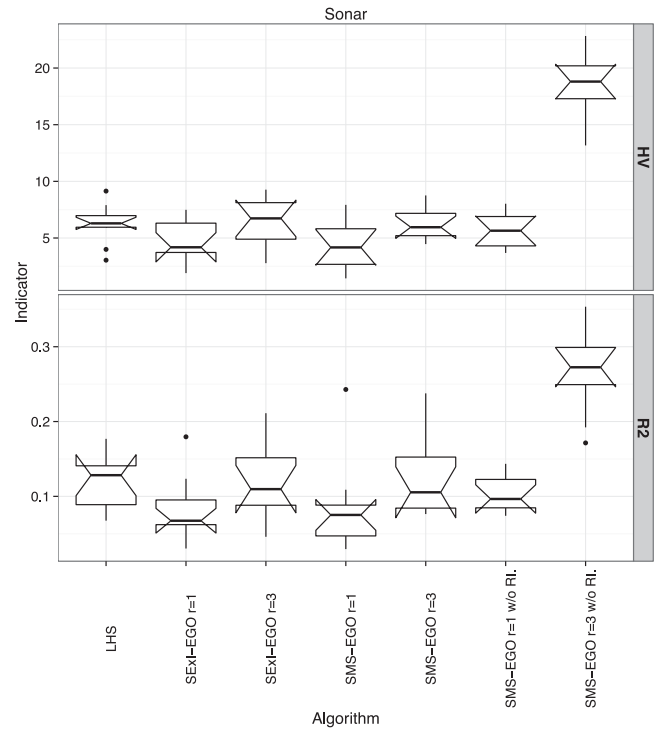

Fig. 5. R2 and HV quality indicators for ten independent runs on task Sonar. The non-dominated solutions obtained after 150 function evaluations were evaluated again ten times, each time using different training/validation splittings to remove biased evaluations. For each quality indicator the mean value of these re-evaluations was calculated. Then we determined the differences between the reference set (set of non-dominated solutions of all runs) and the mean quality indicator values. Thus values of zero would be optimal. We compare the EGO variants both with one function evaluation ($r = 1$) and three function evaluations ($r = 3$). Additionally, SMS-EGO was run without re-interpolation (SMS-EGO w/o RI, variant (c)).

for each sequential design point) we used the re-interpolation procedure by Forrester et al. with the SMS infill criterion.

It can be seen from the plot, that promising non-dominated solutions could be attained for both criteria. For the gain criterion (prediction accuracy) solutions with a classification accuracy of better than 90% were obtained, while for the second criterion very fast runtimes of a few milliseconds are available after the tuning.

Surprisingly for SMS-EGO with $r = 1$ the approximation quality appears to be better than in the lower plot ($r = 3$: three evaluations for each sequential design point). As a consequence using replicates with a reduced total number of iterations results in worse approximation, compared to having more exploration through spending only a single evaluation for each design point.

It is possible, that the solutions with a gain value better than −90% are biased due to the random resampling. However, all solutions in ML are subject to noise, and we evaluated solutions carefully in our final evaluation of the algorithm variants. Therefore we re-evaluated all non-dominated solutions of all tuning algorithm variants ten times after the budget was fully spent (here: 150 trainings and evaluations of the Sonar dataset). This was done to remove possible bias in the non-dominated sets. Then a reference set for all algorithms was formed, consisting of the union of all non-dominated solutions from all variants and runs of the re-evaluation. The differences of the obtained quality indicator values for dominated hypervolume (HV) and R2 between each run and this reference set are shown in Fig. 5. Small values indicate good performance of the algorithm, while large differences to the reference set indicate poor performance.

It can be seen from the boxplots that all EGO variants are better in median than the LHS solutions. Based on a t-test we can give evidence that SExI-EGO with three evaluations ($r = 3$) is significantly better than the simple LHS baseline approach (p-values of 0.037 for the HV indicator and 0.027 for the R2 indicator). All other variants did not show significant differences to LHS, but as the boxplot reveal, the variants with $r = 1$ and re-interpolation reach better values than LHS. Another thing to note is, that although sometimes LHS performs good, this will be in general only true for small parameter spaces. In the Sonar experiment, where we have only two tunable parameters, the performance of the LHS is not so bad when enough evaluations are available. But if we further limit the budget, LHS performs worse. Besides that, it is obvious that as soon as the dimension of the input space increases, LHS will probably fail, or will need a tremendous number of real evaluations. Results for larger parameter spaces are shown later in this article when tuning the parameters for the AppAcid dataset.

The behaviour of SMS-EGO, SExI-EGO and LHS during the optimization process is shown as a line plot in Fig. 6. It can be seen from the plot that the EGO variants achieve a very fast decrease of the difference to the attainment surface, especially in the first 50 sequential iterations. Although LHS does not include a sequential process, we plot it as a line here. This line was produced by evaluating the quality indicators, when 51, . . ., 150 LHS points are considered in the approximation set. It can be concluded that LHS converges more slowly to the reference set than SMS-EGO or SExI-EGO.

To highlight the amount of noise of the ML problem (we term the observed noise of the problem *empirical variance*), we analyzed the variances of the obtained parameter settings. Therefore we re-evaluated each solution obtained from the SMS-EGO runs ten times, each time using a new random sample for the training set. We show the results of the *empirical variance* for the two objectives in Fig. 7,

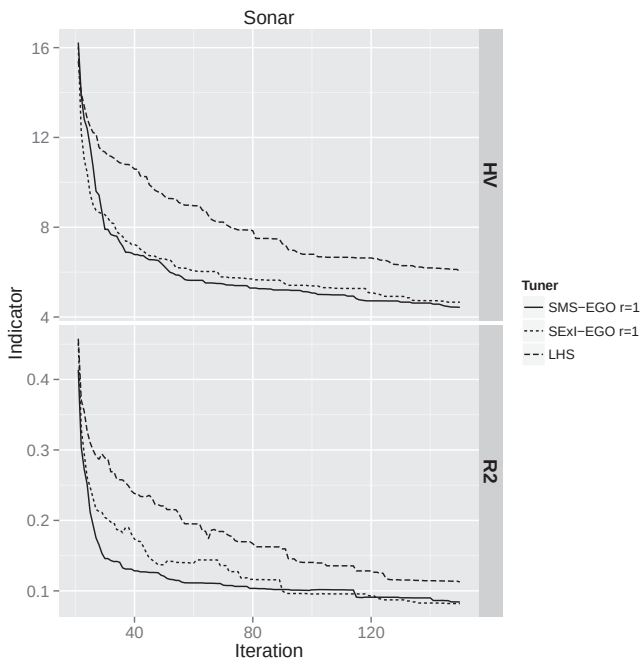

Fig. 6. Sonar: Mean HV and R2 development of ten runs of SExI-EGO, SMS-EGO and LHS. The plot shows that the EGO variants approximate quicker to good HV and R2 approximations compared to LHS. The initial design of size 20 is not plotted for the variants, but we show the situation directly after the initial evaluations.

where the plot of Fig. 4 is extended with the variances obtained in the re-evaluation procedure.

We also investigated the quality of the surrogate model predictions. In Fig. 8 we compare the objective 'gain' obtained during tuning with ten repeated evaluations on the real objective function. Furthermore we compare the Kriging re-interpolation approach (smoothing) with the observed solutions. In the optimal case all points would lie on the diagonal of the plot, symbolized by a red diagonal line. However, a perfect fit is unrealistic, because a certain amount of noise is always present in the data and causes a more or less fluctuation of the results around a mean value. But as can be seen from the plot, the smoothing with one evaluation for each design point leads to predicted solutions which tend to be close to the diagonal with some outliers. Performing more than one evaluation for the design points (repeated evaluations, Fig. 8 bottom right) does seem to have a negative effect on the estimated accuracy.

3.6. AppAcid

AppAcid is a dataset for classifying UV/vis spectrography measurements of a biogas plant [60]. It contains $C = 5$ classes and the class patterns are unevenly distributed. It belongs to the nature of the problem that we are not only interested in maximizing the overall classification accuracy, but to get a good classifier for all five classes. This is also known as cost-sensitive learning, and can be integrated in arbitrary learning algorithms, e.g., by using class weights. A popular method in this research area is the MetaCost algorithm by Domingos [61], which has been used here. We denote the class weight parameters by cutoff1, ..., cutoff4. The cutoff parameter for class 5 is then automatically calculated in relation to the other 4 values.

Besides the integration of the class weight parameters into the tuning, the objective function for the classification accuracy has to be changed. E.g., we now aim at minimizing the *mean error rate* or maximizing the *mean classification accuracy*, where the latter is defined as follows:

$$MCA = 1 - \frac{1}{C}\sum_{i=1}^{C} e_i \tag{3.1}$$

where $C = 5$ denotes the number of classes and e_i is the class error rate obtained on class i. Earlier SCO tuning with model-assisted optimization techniques [3] showed performances in mean classification accuracy of up to 88%.

It was our goal to show the effect of balancing runtime and prediction accuracy. Our approach shows an interesting behaviour of the runtime, because SCO in ML is known to be very time-consuming when state-of-the-art models are considered. It can be seen from Fig. 9 that the two criteria optimization has no negative effect to the overall model accuracies. As in the single-criteria task before, the optimized SVM classifiers reached a high mean

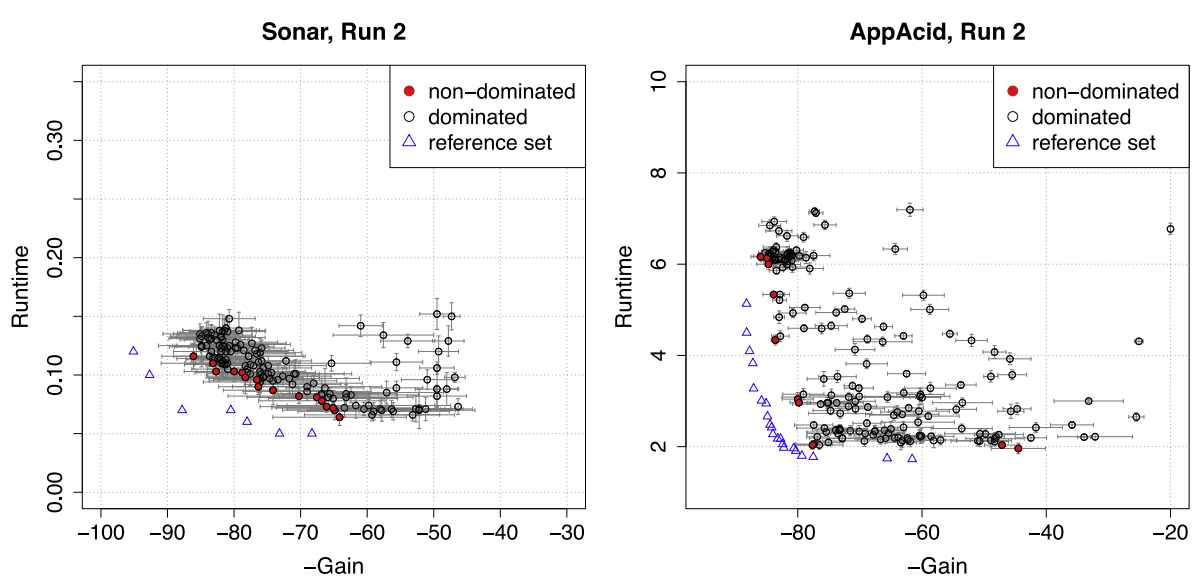

Fig. 7. Objective space plots showing the empirical variances for the two objectives – *Gain* (%) and *Runtime* (s). The error bars represent the standard deviations from 10 repeated evaluations (re-training on a new training sample) of a certain parameter point. The gain is obtained on an independent test set. Shown are the variances for Sonar and AppAcid. The blue triangles denote the reference sets from Fig. 4 and Fig. 9, respectively. The variances are bigger for Sonar since the number of records in a sample is smaller. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

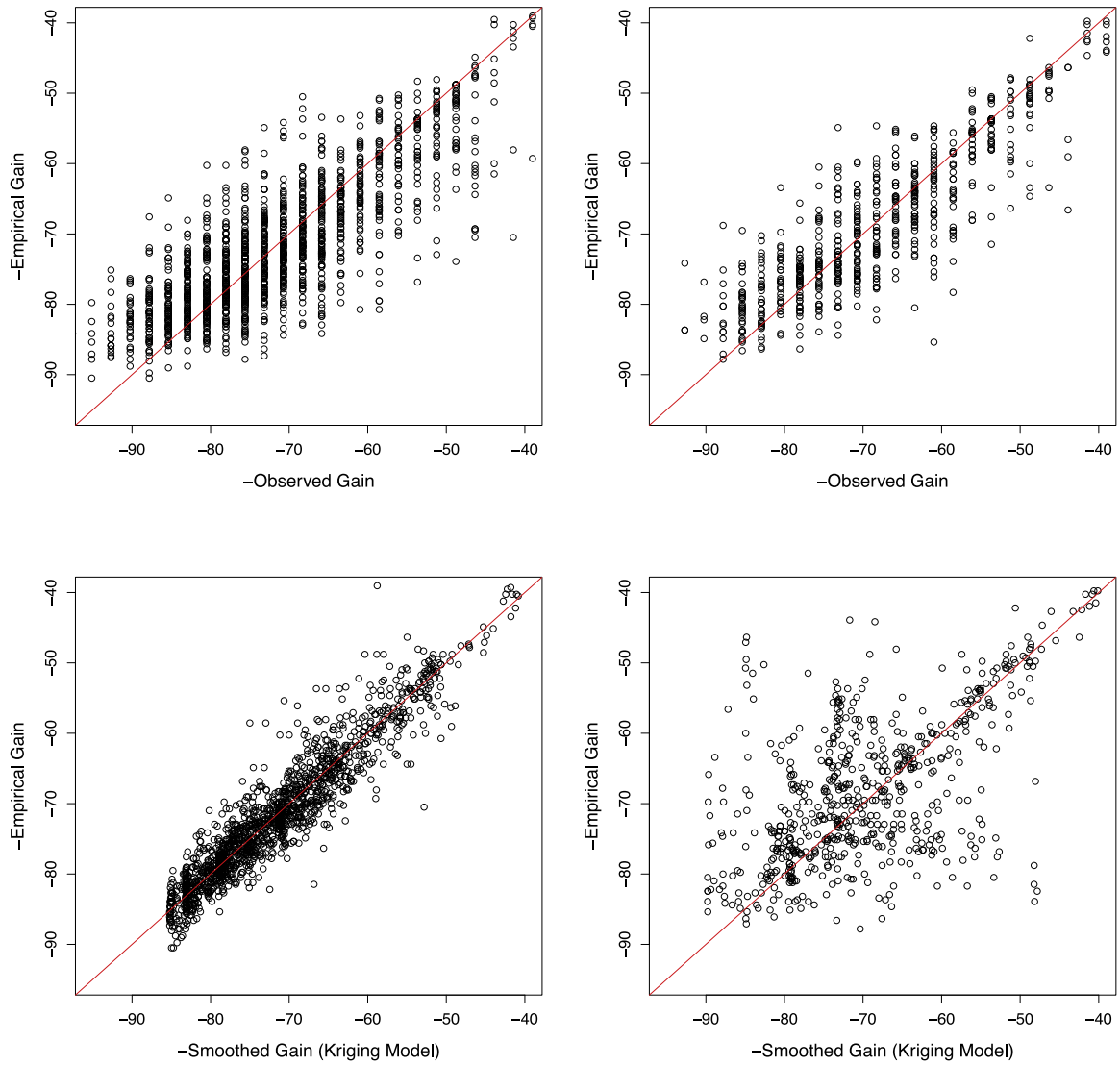

Fig. 8. Tuning result of SMS-EGO on the Sonar dataset for the first objective *gain*. All plots in this figure are based on the results of ten runs using SMS-EGO as a tuning algorithm. In the two upper plots the observed results seen by SMS-EGO are displayed on the *x*-axis, while the results of a re-evaluation is shown on the *y*-axis, where each design point was evaluated again ten times (yielding the empirical gain). In the plots the red diagonal line denotes possible situations where the empirical gain would be equal to the observed gain and smoothed gain respectively. The upper left plot shows the observed points of ten runs of SMS-EGO using a single evaluation for each design point (coefficient of determination $R^2 = 0.65$). In the upper right plot the results of three evaluations for each design point in the sequential part of the algorithm are displayed (coefficient of determination $R^2 = 0.74$). The two bottom plots then show the same re-evaluated points, but the smoothed predictions of the Kriging surrogate model are used for comparison instead of the observed points (*x*-axis). Again we differ between one evaluation for each design point (bottom left) and three evaluations for each design point (bottom right). While for one evaluation the smoothing leads to a very high coefficient of determination $R^2 = 0.89$, the repeated evaluations seem to have a negative influence on the smoothing ($R^2 = 0.32$ in the case of three evaluations for each design point). As a conclusion the best estimation is obtained with a smoothed model (Kriging re-interpolation) and one evaluation for each design point. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

classification accuracy of up to 88%. Interestingly the plot also shows that high computation times do not necessarily lead to good prediction models in terms of accuracy. With wrong settings for *xperc*, the percentage of selected features, even a high runtime can lead to disappointing low gain as the crosses around *Gain* = 0.2 in Fig. 9 show. On the other hand it can be also seen from the plot, that solutions with small fractions of the total training size do not lead to much worse classification performances. Although there is a steep slope in the area of 85% classification accuracy and higher, the runtime criterion for non-dominated solutions seems to be stable for smaller classification accuracies.

Similar to the Sonar task, we re-evaluated the non-dominated sets of each run ten times for all variants considered in this study. The differences of the re-evaluated points and the corresponding reference set are shown in Fig. 11. For the AppAcid task 200 ML trainings were spent, since more parameters had to be tuned compared to the Sonar task. The different EGO variants are discussed in more detail in Section 4.1.

4. Discussion

4.1. Noise handling

In ML it is difficult to obtain a good and reliable estimator of the real generalization error of a prediction model. In our MCO experiments we evaluated the models using the holdout set error (the classification error obtained on a validation set). Although this gives a better estimate than just to consider the training error (which would rather lead to overfitting issues), the objective

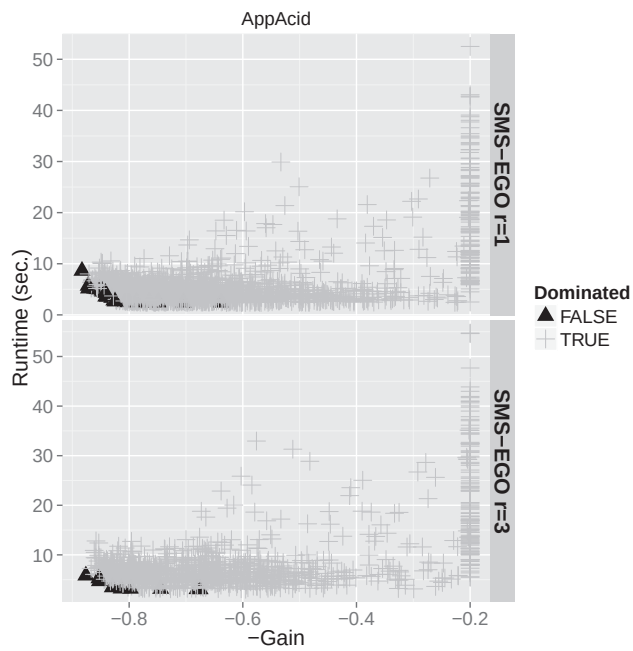

Fig. 9. Plot of objectives for AppAcid. The mean classification accuracy (*gain*) is plotted on the *x*-axis and the runtime in seconds on the *y*-axis. Note, that the negative value for the gain is taken, because all objectives are being minimized.

function values are still subject to high noise levels, because they depend on the randomly sampled training data. Therefore we need appropriate methods to cope with the high level of noise in these ML evaluations. We compared three variants for our optimization under restricted budgets:

(a) the RI method by Forrester et al. [26], one evaluation of each design point ($r = 1$),
(b) replicates: repeated evaluations of each design point without RI ($r = 3$, this allows only one third of the infills), and
(c) no RI and no replicates ($r = 1$).

The goal is to avoid too early convergence of the surrogate models to misleading solutions caused by noise.

Variant (a) tries to avoid selecting biased solutions by an internal estimation of the noise. Variant (b) instead explicitly performs re-evaluations to obtain more stable results during the optimization and to give better estimates for the surrogate model predictions. Variant (c) dealt as baseline comparison to the special noise handling approaches. In our experiments, the variant using one evaluation ($r = 1$) and noise estimation within the Kriging procedure (re-interpolation) performed worse once on task Sonar (SExI-EGO using three evaluations performed better than SExI-EGO ($r = 3$), Fig. 5). Also, SMS-EGO ($r = 3$) performed better on task AppAcid (compared to SMS-EGO ($r = 1$), Fig. 11). But it has to be noted, that in both cases the significance level of the *t*-test did not reach the 95% confidence level for both quality indicators R2 and dominated HV. This means that we cannot give evidence for using $r = 3$ rather than $r = 1$ on any of these tasks or vice versa.

All EGO variants are significantly better than LHS (*t*-tests with $p < 0.029$) on task AppAcid for the R2 indicator, with the exception of SExI-EGO ($r = 3$). Interestingly, the behaviour for different quality indicators is not necessarily the same, which can be seen from the result for SExI-EGO ($r = 1$) on task AppAcid. In this boxplot a very large box was obtained for indicator HV, while the box for the R2 indicator is rather small, leading to a significant better result of the EGO approach (*t*-test for R2 with $p = 0.029$, while the *p*-value for HV is 0.977).

Very high noise levels were observed for the Sonar task, leading to rather unstable results. As a positive result in Fig. 5 the EGO variants reveal smaller boxes, indicating a higher robustness of the optimization compared to LHS. However, in our statistical test (*t*-test, confidence level 95%) only SExI-EGO ($r = 3$) was significantly better than LHS. The good result of SExI-EGO ($r = 3$) can be justified by making use of repeated evaluations, when the task is very noisy. Instead when the parameter space becomes larger (e.g., task AppAcid), more exploration is required, and the algorithms using replicates do not perform as well (Fig. 11). It is also an important achievement that all EGO variants result in more stable quality indicator measures as can be derived from the smaller box sizes in the boxplot of Fig. 5.

A point of discussion is the setting of the reference point for the hypervolume indicator. We noticed that this point plays an important role, because it can lead to a perturbation of the results. This most likely occurs, when the reference point is set too close to the points in the approximation set. As a consequence some points do not contribute much to the hypervolume indicator, resulting in a too pessimistic evaluation of the approximation set. Therefore special care must be given to the corner solutions. Such solutions will not contribute to the dominated hypervolume if the reference point is set too close. Out of this reason we always set the reference point

$$r = \left(\max_{\vec{x} \in N} f_1(\vec{x}) + C_1, \max_{\vec{x} \in N} f_2(\vec{x}) + C_2 \right) \tag{4.1}$$

where N denotes the set of non-dominated solutions from the reference set and C_i are additional constants for each objective. Here we chose $C_i = 1$. For the R2 indicator we set an utopian point as $(-1, 0)^T$ which is guaranteed to dominate every real solution.

Summarizing we find no significant difference between the noise-dampening variant with $r = 3$ and the other variant $r = 1$ which has more noise but also allows more infills.

RI deals successfully with the observed noise at the ($r = 1$)-level. SMS-EGO with RI performs always slightly better than SMS-EGO without RI, the difference are however not significant in our current experiments. It is matter of future research to reveal whether RI makes a difference compared to simple interpolating Kriging strategies without special noise handling.

4.2. Parameter space dimension

It is interesting to note the influence of the parameter space dimension on the relative performance of LHS against the MC-EGO variants. In the Sonar case (dimension 2) LHS is very close to the EGO results (Fig. 5). This is not too surprising, because we spent a large number of 150 function evaluations for the Sonar task. The line plot in Fig. 6 shows that the EGO variants could obtain good approximations of the reference set after 50 iterations. This indicates that EGO would also produce good results with fewer function evaluations, and thereby would probably outperform the LHS. But finally, after a sufficient number of function evaluations (here, after 150 iterations), all algorithms achieve a comparable attainment of the reference set.

In contrast to this, the AppAcid task (dimension 7) shows that LHS cannot keep pace with EGO, because, as Fig. 10 shows, the MC-EGO indicators now drop much faster: Even at iteration 100 (50 initial design + one third of 150), the HV-indicators for both EGO variants are already better than LHS at iteration 200.

4.3. Single- and two criteria optimization compared

MCO spends the budget for examining different regions of the Pareto front and may thus not find the best parameters in the high-gain region of the front. SCO on the other hand concentrates solely

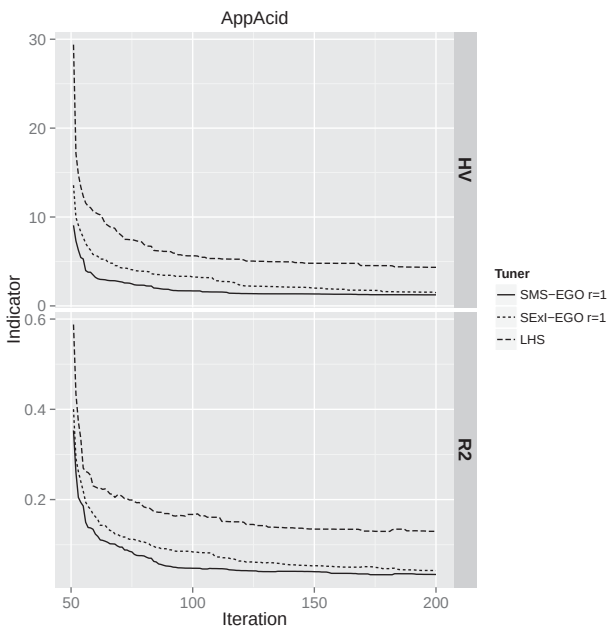

Fig. 10. AppAcid: Mean HV and R2 development of ten runs of SExI-EGO, SMS-EGO and LHS. The plot shows that the EGO variants approximate quicker to good HV and R2 approximations compared to LHS. The initial design of size 50 is not plotted for the variants, but we show the situation directly after the initial evaluations.

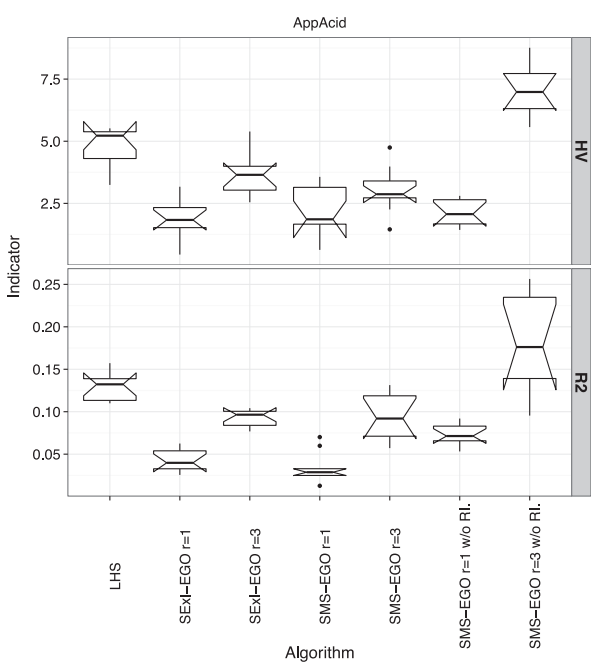

Fig. 11. R2 and HV quality indicators for ten independent runs on task AppAcid. The non-dominated solutions obtained after 200 function evaluations were evaluated again ten times, each time using different training/validation splittings to remove biased evaluations. For each quality indicator the mean value of these re-evaluations was calculated. Then we determined the differences between the reference set (set of non-dominated solutions of all runs) and the mean quality indicator values. Thus values of zero would be optimal. We compare the EGO variants both with one function evaluation ($r = 1$) and three function evaluations ($r = 3$). Additionally, SMS-EGO was run without re-interpolation and a single evaluation for each design point (SMS-EGO w/o RI, variant (c)).

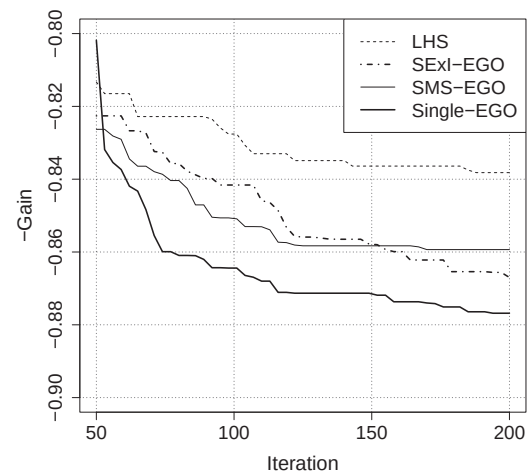

Fig. 12. AppAcid: Development of the objective – *Gain* during single-criteria EGO (thick line), SMS-EGO and SExI-EGO as two multi-criteria EGO approaches (thin and dash-dotted lines) and LHS as baseline approach. Shown is the average of ten independent runs with different training samples.

on the gain and thus will be in general better with respect to this gain. The question is: how much better?

To answer this question we performed for task AppAcid ten SCO-runs with the same parameters to optimize, the same RoI and the same budget (50 initial design points, 150 sequential infills) as in the MCO runs. As SCO tuning algorithm we choose the well-known sequential parameter optimization (SPO) [62] in the TDMR-framework [57]. SPO uses Kriging as well, but in this case only for single-criteria optimization of the objective *gain*. We show in Fig. 12 the average of ten runs. It is interesting to note that the MCO-gain is only slightly below the SCO-gain which is quite surprising because MCO operates on two objectives simultaneously.[5] At the same time, MCO gives the user more information with respect to the tradeoff between runtime and quality (*gain*). This might be of interest for the ML-practitioner.

4.4. EGO runtimes

A disadvantage of EGO are the high computation times, which can exceed the computation times of the objective function evaluations. E.g., the total runtime of the AppAcid optimization was 15.8 h with SMS-EGO, while the runtime of SExI-EGO was 143.6 h (almost 6 days). This result was determined with all settings as described before, having 200 evaluations in total, an initial design size of 50, and one evaluation per design point ($r = 1$). Because the largest runtime for the function evaluations is below one minute, we can conclude that the real objective function cannot be the source for the long runtime. Hence, the optimization on the surrogate model must be responsible for the high computation times. The behaviour can be constituted by the frequent calculation of the hypervolume indicator. Since the calculation of the hypervolume is NP-hard, this bares a real disadvantage for the model-assisted MCO approach. The reason why SExI-EGO has almost a ten times higher runtime, is caused by the underlying implementation. Recently, faster algorithms for the computation of the hypervolume-based expected improvement used in SExI-EGO have been proposed in Hupkens [63] and in Couckuyt et al. [64]. This will probably lead to reduced runtimes of SExI-EGO, but an implementation of this approach was

[5] We note in passing, that an earlier SCO-experiment [3] on task AppAcid with method SVM and a slightly different RoI had a mean gain of 86.1%, which is on the same level as the current MCO-results.

not available at the time of our experiments, so our results are based on the original implementation. Nevertheless one has to note, that the MCO approach makes sense when the runtime of the EGO is significantly lower than the runtime of the learning algorithm. In our benchmark this was not the case, but for really large datasets this will be usually true.

5. Conclusion

Summarizing we showed that two criteria optimization can help to offer the user a variety of possible solutions to select from. In earlier SCO approaches the selection was restricted to use the best generalizing model delivered by the optimization procedure. Now, with the two criteria approach presented in this paper, the user is able to select the best models spending a certain limited time budget for the training process.

We applied for the first time two hypervolume-based EGO variants, namely SMS-EGO and SExI-EGO, to noisy ML tasks. It was found that they operate well, even in the presence of considerable noise. Both algorithms deliver comparable quality in terms of the covered hypervolume and the R2-indicator. SMS-EGO in its current implementation is considerably faster than SExI-EGO.

To operate well under heavily restricted budgets, both MCO-approaches use Kriging models to select the next infill point. Those Kriging models have to work robustly under the presence of noise, noise being inevitable in most ML tasks due to the variations of randomly drawn training samples and the characteristics of the data. We found it to be crucial to use the re-interpolation technique to cope with the noise. Plain Kriging models without re-interpolation (variant (c), boxplot *SMS-EGO w/o RI* in Fig. 11) perform worse, because they tend to overfit the noise.

With the re-interpolation in place, we found that it is well capable to deal with the noise in our tasks. An additional noise reduction by aggregating repeated evaluations (see Section 4.1) can be necessary, when really high noise levels occur. For larger dimensions, repeated evaluations diminish the exploration of the search space, deteriorating the approximation quality due to a reduced number of infill steps.

On the MCO tasks, both Kriging-based EGO techniques performed better than the baseline LHS approach. The advantage increases with the number of dimensions in the parameter space: The difference between LHS quality indicator values and EGO quality indicator values is much higher in task AppAcid (parameter space of dimension 7) than in task Sonar (dimension 2).

Acknowledgements

This work has been supported by the Bundesministerium für Bildung und Forschung (BMBF) under the grant SOMA (AiF FKZ 17N1009) and by the Cologne University of Applied Sciences under the research focus grant COSA.

The SMS-EGO variants are based on investigations of the project D5 “Synthesis and multi-objective model-based optimization of process chains for manufacturing parts with functionally graded properties” as part of the collaborative research center SFB/TR TRR30 which is kindly supported by the Deutsche Forschungsgemeinschaft (DFG).

Appendix A. Region of interest for Sonar

See Table A1.

Table A1
Region of interest (RoI) for Sonar tuning experiments. See main text in Section 3.2 for explanation of parameters.

Parameter	Lower	Upper
gamma	0.01	0.7
trnFrac	0.05	0.7

Appendix B. Region of interest for AppAcid

See Table B1.

Table B1
RoI for AppAcid tuning experiments. Parameters *gamma* and *trnFrac* have the same meaning as in the Sonar task, see Section 3.2. Parameter *xperc* controls the feature selection, the algorithm selects a number of the most important features to capture a fraction *xperc* of the overall importance. The parameters *CUTOFFi*, $i = 1, \ldots, 4$ control the weight for each class.

	Lower	Upper
gamma	0.001	0.8
trnFrac	0.2	0.7
xperc	0.050	1.0
CUTOFF1	0.010	0.4
CUTOFF2	0.010	0.4
CUTOFF3	0.010	0.4
CUTOFF4	0.010	0.4

Appendix C. Machine learning and SVM

Machine learning is a sub-field of artificial intelligence and can be divided into supervised and unsupervised learning. The goal of supervised learning is to construct a method, which is capable to learn from observed data. The observations are usually represented as tuples of the input attributes (or features) $\mathcal{X}$ and their corresponding outputs $\mathcal{Y}$[6]:

$$(\vec{X}_1, Y_1), (\vec{X}_2, Y_2), \ldots, (\vec{X}_n, Y_n) \in \mathcal{X} \times \mathcal{Y} \tag{C.1}$$

Algorithms for supervised learning include amongst others tree-based classifiers such as random forests [65], artificial neural networks (ANN), and Support Vector Machines (SVMs) [7]. We focus on SVM and classification in this article.

SVM have been originally introduced as learning algorithms for binary classification and regression tasks [7]. Because SVM are known to be sensitive to computation times and parameter settings, we use them in our experiments. In binary classification, SVM seek the maximal margin classifier, which best separates the two classes. In the simplest case this means to search for the optimal separating hyperplane by minimizing the empirical risk. However, if SVM could only classify separable data, the applicability would be very limited, since most real-world problems are not linearly separable. Therefore, SVM performs classification with the following extensions:

(1) Kernel-induced feature space: the input space is implicitly mapped to a higher-dimensional space using a kernel function

$$K(\vec{X}, \vec{Z}) : \langle \phi(\vec{X}_i) \cdot \phi(\vec{X}_j) \rangle \tag{C.2}$$

where ϕ defines a mapping from the input space. Thus, the kernel function denotes the similarity of two observations $\vec{X}_i, \vec{X}_j \in \mathcal{X}$. Because the kernel function can be interpreted as a dot product in a high-dimensional space, the computation is feasible

[6] For a better differentiation between optimization and ML, we write observations in ML with capital letters, while we denote candidate solutions in optimization with lower case letters.

also for very high dimensions. When the patterns cannot be separated by a linear classifier in the original feature space, this can be still possible in the kernel-induced feature space. The kernel function can be defined anew for each task, or pre-defined kernel functions can be chosen. A necessary requirement of kernel functions is that they have to be symmetric and positive semi-definite. In our experiments we used the RBF kernel, because it outperformed other kernels in preliminary experiments:

$$K(\vec{X}_i, \vec{X}_j) = \exp(-\gamma \cdot ||\vec{X}_i - \vec{X}_j||^2) \tag{C.3}$$

where γ is a parameter for the kernel function.

(2) Regularized risk minimization: If some observations are still not classified correct in the kernel-induced feature space, SVM can make use of the soft-margin concept by Cortes and Vapnik [66]. In soft-margin SVM a regularization term is introduced, which penalizes wrongly classified patterns. For simplicity, we write the optimal SVM classification model in the unconstrained dual form, denoting H as the reproducing kernel Hilbert space for the kernel function $K(\cdot, \cdot)$, in the following optimization problem:

$$\hat{F} = arg \inf_{F \in H, b \in \mathbb{R}} ||F||_H^2 + C \sum_{i=1}^{n} L(Y_i, F(\vec{X}_i) + b) \tag{C.4}$$

Here f is the real-valued target function, and $\hat{F}$ the regularized target function. In case of binary classification we would discretize the real-valued output of $\hat{F}$ by mapping it to $\{-1, 1\}$. The first term is called a smoothness penalty, as the 2-norm can be used to penalize non-smooth functions. The second term measures the closeness of our predictions to the true outputs by means of a loss function. In classification, we usually select the hinge loss $L(Y, t) = L_h(Y, t) = \max(0, 1 - Yt)$ for an intended output $t = \pm 1$ and a classifier score Y. The two terms are balanced by the parameter C, sometimes also referred to as *Cost*. In recent SVM implementations, a value of $C = 1$ is taken as default, equally weighting loss function and smoothness penalty.[7]

Although SVM have been originally designed for binary classification, they can also be used to solve multi-class problems, e.g., by incorporating multiple "one-against-all classifiers" (see [69] for a comprehensive overview).

Appendix D. The R_2 indicator

For a given approximation set A and a set of arbitrary utility functions U, the R_2 indicator is defined as follows

$$R_2(A, U) := -\frac{1}{|U|} \sum_{u \in U} \max_{a \in A} u(a) \tag{D.1}$$

As utility function we used the weighted Tchebycheff distance, but in fact also other metrics (e.g., the Minkowski distance) could be used here:

$$u(z) = u_\lambda(\vec{z}) = -\max_{j \in \{1, \ldots, m\}} \lambda_j |z_j^* - z_j| \tag{D.2}$$

with z^* the utopian point and $\vec{\lambda} = (\lambda_1, \ldots, \lambda_m)^T$ a weight vector for the objective functions. Different, randomly chosen $\vec{\lambda}$ constitute the utility function set U. In our experiments we sampled U uniformly at random from the interval [0, 1] as proposed by Hansen and Jaszkiewicz [58]. The size of U is heuristically defined in Eq. (D.3):

$$|U| = \binom{s + m - 1}{m - 1} \tag{D.3}$$

with

$$s = \begin{cases} 500 & m = 2 \\ 30 & m = 3 \\ 12 & m = 4 \\ 8 & m = 5 \\ 3 & else \end{cases} \tag{D.4}$$

Other performance indicators are not considered here, but with similar advantageous properties the averaged Hausdorff distance [70] to the reference attainment surface could be considered.

[7] In the *R* implementations contained in the *e1071* package [67] and in the *kernlab* package [68], the default values for C are set to 1.

References

[1] P. Koch, B. Bischl, O. Flasch, T. Bartz-Beielstein, C. Weihs, W. Konen, Tuning and evolution of support vector kernels, Evol. Intell. (2011) 1–18 (Special Issue on Evolutionary Kernel Machines).

[2] P. Koch, W. Konen, Efficient sampling and handling of variance in tuning data mining models, in: V. Cutello, M. Pavone (Eds.), PPSN 2012: 12th International Conference on Parallel Problem Solving from Nature, Springer, Berlin, Taormina, Italy, 2012, pp. 195–205.

[3] W. Konen, P. Koch, O. Flasch, T. Bartz-Beielstein, M. Friese, B. Naujoks, Tuned data mining: a benchmark study on different tuners, in: Proceedings of Genetic and Evolutionary Computation Conference (GECCO) 2011, ACM, Dublin, Ireland, 2011, pp. 1995–2002.

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

[5] T. Wagner, M. Emmerich, A. Deutz, W. Ponweiser, On expected-improvement criteria for model-based multi-objective optimization, in: Proceedings of the International Conference on Parallel Problem Solving from Nature (PPSN XI), Springer, 2010, pp. 718–727.

[6] V. Vapnik, Statistical Learning Theory, John Wiley and Sons Inc., New York, 1998.

[7] B. Schölkopf, A. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond, The MIT Press, Cambridge, MA, USA, 2002.

[8] K. Burnham, D. Anderson, Model Selection and Multi-model Inference: A Practical Information-theoretic Approach, Springer, New York, 2002.

[9] W. Zucchini, An introduction to model selection, J. Math. Psychol. 44 (1) (2000) 41–61.

[10] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, Choosing multiple parameters for support vector machines, Mach. Learn. 46 (1) (2002) 131–159.

[11] S. Keerthi, Efficient tuning of SVM hyperparameters using radius/margin bound and iterative algorithms, IEEE Trans. Neural Netw. 13 (5) (2002) 1225–1229.

[12] S. Keerthi, V. Sindhwani, O. Chapelle, An efficient method for gradient-based adaptation of hyperparameters in SVM models, in: Schölkopf B., et al. (Eds.), Advances in Neural Information Processing Systems 19, 2007, pp. 673–680.

[13] G. Cohen, M. Hilario, A. Geissbuhler, Model selection for support vector classifiers via genetic algorithms. An application to medical decision support, Biol. Med. Data Anal. (2004) 200–211.

[14] F. Friedrichs, C. Igel, Evolutionary tuning of multiple SVM parameters, Neurocomputing 64 (2005) 107–117.

[15] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evol. Comput. 9 (2001) 159–195.

[16] N. Hansen, The CMA evolution strategy: a comparing review, in: J. Lozano, P. Larranaga, I. Inza, E. Bengoetxea (Eds.), Towards a New Evolutionary Computation. Advances on Estimation of Distribution Algorithms, Springer, Berlin, Heidelberg, 2006, pp. 75–102.

[17] T. Glasmachers, C. Igel, Gradient-based adaptation of general Gaussian kernels, Neural Comput. 17 (10) (2005) 2099–2105.

[18] T. Glasmachers, C. Igel, Uncertainty handling in model selection for support vector machines, in: Parallel Problem Solving from Nature – PPSN X, 2008, pp. 185–194.

[19] G. Box, K. Wilson, On the experimental attainment of optimum conditions, J. R. Stat. Soc. 13 (1) (1951) 1–45.

[20] J. Sacks, W. Welch, T. Mitchell, H. Wynn, Design and analysis of computer experiments, Stat. Sci. 4 (4) (1989) 409–423.

[21] Y. Jin, A comprehensive survey of fitness approximation in evolutionary computation, Soft Comput. 9 (1) (2005) 3–12.

[22] A. Ratle, Kriging as a surrogate fitness landscape in evolutionary optimization, Artif. Intell. Eng. Des. Manuf. 15 (1) (2001) 37–49.

[23] M. Emmerich, K. Giannakoglou, B. Naujoks, Single-and multiobjective evolutionary optimization assisted by Gaussian random field metamodels, IEEE Trans. Evol. Comput. 10 (4) (2006) 421–439.

[24] Z. Zhou, Y.S. Ong, P.B. Nair, A.J. Keane, K.Y. Lum, Combining global and local surrogate models to accelerate evolutionary optimization, IEEE Trans. Syst. Man Cybern. C: Appl. Rev. 37 (1) (2007) 66–76.
[25] D. Lim, Y. Jin, Y.-S. Ong, B. Sendhoff, Generalizing surrogate-assisted evolutionary computation, IEEE Trans. Evol. Comput. 14 (3) (2010) 329–355.
[26] A. Forrester, A. Keane, N. Bressloff, Design and analysis of "noisy" computer experiments, AIAA J. 44 (10) (2006) 2331.
[27] D. Huang, T.T. Allen, W.I. Notz, N. Zeng, Global optimization of stochastic black-box systems via sequential Kriging meta-models, J. Global Optim. 34 (3) (2006) 441–466.
[28] V. Picheny, D. Ginsbourger, Y. Richet, Noisy expected improvement and on-line computation time allocation for the optimization of simulators with tunable fidelity, in: 2nd International Conference on Engineering Optimization, 2010, pp. 1–10.
[29] V. Picheny, T. Wagner, D. Ginsbourger, A benchmark of Kriging-based infill criteria for noisy optimization, Struct. Multidis. Optim. 48 (3) (2013) 607–626.
[30] G. Liu, V. Kadirkamanathan, Learning with multi-objective criteria, in: Proceedings of the 4th International Conference on Artificial Neural Networks, IET, 1995, pp. 53–58.
[31] A. Freitas, A critical review of multi-objective optimization in data mining: a position paper, ACM SIGKDD Explor. Newsl. 6 (2) (2004) 77–86.
[32] Y. Jin, B. Sendhoff, Pareto-based multiobjective machine learning: an overview and case studies, IEEE Trans. Syst. Man Cybern. C: Appl. Rev. 38 (3) (2008) 397–415.
[33] Y. Jin, Pareto-based multi-objective machine learning, in: Proceedings of the 7th International Conference on Hybrid Intelligent Systems (HIS 2007), IEEE, 2007, p. 2.
[34] J. Knowles, H. Nakayama, Meta-modeling in multiobjective optimization, Multiobjective Optimization (2008) 245–284.
[35] K. Giannakoglou, A. Giotis, M. Karakasis, Low-cost genetic optimization based on inexact pre-evaluations and the sensitivity analysis of design parameters, Inverse Probl. Eng. 9 (4) (2001) 389–412.
[36] M. Emmerich, B. Naujoks, Metamodel assisted multiobjective optimisation strategies and their application in airfoil design, in: Adaptive Computing in Design and Manufacture VI, 2004, pp. 249–260.
[37] G. Ascia, V. Catania, A. Di Nuovo, M. Palesi, D. Patti, Performance evaluation of efficient multi-objective evolutionary algorithms for design space exploration of embedded computer systems, Appl. Soft Comput. 11 (1) (2011) 382–398.
[38] J. Knowles, ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems, IEEE Trans. Evol. Comput. 10 (1) (2006) 50–66.
[39] W. Ponweiser, T. Wagner, D. Biermann, M. Vincze, Multiobjective optimization on a limited budget of evaluations using model-assisted $\mathcal{S}$-metric selection, in: Proceedings of the International Conference on Parallel Problem Solving from Nature (PPSN X), Springer, 2008, pp. 784–794.
[40] M. Emmerich, A. Deutz, J. Klinkenberg, Hypervolume-based expected improvement: monotonicity properties and exact computation, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC), IEEE, 2011, pp. 2147–2154.
[41] Q. Zhang, W. Liu, E. Tsang, B. Virginas, Expensive multiobjective optimization by MOEA/D with Gaussian process model, IEEE Trans. Evol. Comput. 14 (3) (2010) 456–474.
[42] M. Zaefferer, B. Naujoks, T. Bartz-Beielstein, M. Emmerich, T. Wagner, A case study on multi-criteria optimization of an event detection software under limited budgets, in: Proceedings of the 7th Evolutionary Multi-Criterion Optimization Conference (EMO 2013), 2013, pp. 756–770.
[43] F. Corona, M. Mulas, R. Baratti, J. Romagnoli, On the topological analysis of industrial process data using the SOM, in: C.O.d.N.R. de Brito Alves, E. Biscaia (Eds.), Computer Aided Chemical Engineering: Proceedings of PSE 2009 International Symposium on Process Systems Engineering, Salvador Bahia, Brazil, vol. 27 of Computer Aided Chemical Engineering, Elsevier, 2009, pp. 1173–1178.
[44] J. Knowles, D. Corne, A. Reynolds, Noisy multiobjective optimization on a budget of 250 evaluations, in: Evolutionary Multi-Criterion Optimization, Springer, 2009, pp. 36–50.
[45] D. Krige, A statistical approach to some basic mine valuation problems on the Witwatersrand, J. Chem. Metal Mining Soc. South Africa 52 (1951) 119–139.
[46] G. Matheron, Traité de géostatistique appliquée, vol. 1, Editions Technip, Paris, 1962.
[47] M. Stein, Interpolation of Spatial Data: Some Theory for Kriging, Springer, New York, 1999.
[48] M. Emmerich, A. Giotis, M. Özdemir, T. Bäck, K. Giannakoglou, Metamodel-assisted evolution strategies, in: Proceedings of the Conference on Parallel Problem Solving from Nature (PPSN VII), Springer, 2002, pp. 361–370.
[49] J. Svenson, Computer experiments: multiobjective optimization and sensitivity analysis (Ph.D. Thesis), Ohio State University, 2011.
[50] S. Sakata, F. Ashida, Ns-kriging based microstructural optimization applied to minimizing stochastic variation of homogenized elasticity of fiber reinforced composites, Struct. Multidis. Optim. 38 (5) (2009) 443–453.
[51] J. Mockus, V. Tiesis, A. Zilinskas, The application of Bayesian methods for seeking the extremum, Towards Global Optim. 2 (1978) 117–129.
[52] E. Zitzler, L. Thiele, J. Bader, On set-based multiobjective optimization, IEEE Trans. Evol. Comput. 14 (1) (2010) 58–79.
[53] K. Shimoyama, K. Sato, S. Jeong, S. Obayashi, Comparison of the criteria for updating Kriging response surface models in multi-objective optimization, in: IEEE Congress on Evolutionary Computation (CEC), IEEE, 2012, pp. 1–8.
[54] K. Shimoyama, K. Sato, S. Jeong, S. Obayashi, Updating kriging surrogate models based on the hypervolume indicator in multi-objective optimization, J. Mech. Des. 135 (9) (2013).
[55] M. Fleischer, The measure of Pareto optima applications to multi-objective metaheuristics, in: Evolutionary Multi-Criterion Optimization (EMO), Springer, 2003, pp. 519–533.
[56] A. Auger, J. Bader, D. Brockhoff, E. Zitzler, Theory of the hypervolume indicator: optimal μ-distributions and the choice of the reference point, in: Proceedings of the 10th ACM SIGEVO Workshop on Foundations of Genetic Algorithms, ACM, 2009, pp. 87–102.
[57] W. Konen, P. Koch, The TDMR package: tuned data mining in R, CIOP Technical Report 02/2012, Cologne University of Applied Sciences (November 2012). http://gociop.de/ciop-reports
[58] M. Hansen, A. Jaszkiewicz, Evaluating the Quality of Approximations to the Non-dominated Set, IMM, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 1998.
[59] E. Zitzler, K. Deb, T. Thiele, Comparison of multiobjective evolutionary algorithms: empirical results, Evol. Comput. 8 (2) (2000) 173–195.
[60] C. Wolf, D. Gaida, A. Stuhlsatz, T. Ludwig, S. McLoone, M. Bongards, Predicting organic acid concentration from UV/vis spectrometry measurements – a comparison of machine learning techniques, Trans. Inst. Meas. Control 19 (2011) 1–11.
[61] P. Domingos, Metacost: a general method for making classifiers cost-sensitive, in: Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 1999, pp. 155–164.
[62] T. Bartz-Beielstein, C. Lasarczyk, M. Preuss, The sequential parameter optimization toolbox, in: T. Bartz-Beielstein, et al. (Eds.), Experimental Methods for the Analysis of Optimization Algorithms, Springer, Berlin, Heidelberg, New York, 2010, pp. 337–360.
[63] I. Hupkens, Complexity reduction and validation of computing the expected hypervolume improvement (Master's thesis), University of Leiden, The Netherlands, 2013.
[64] I. Couckuyt, D. Deschrijver, T. Dhaene, Fast calculation of multiobjective probability of improvement and expected improvement criteria for Pareto optimization, J. Global Optim. 60 (3) (2014) 575–594.
[65] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[66] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[67] E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer, A. Weingessel, e1071: Misc functions of the department of statistics (e1071), TU Wien, 2006, pp. 1–60, e1071 R package (2006).
[68] A. Karatzoglou, A. Smola, K. Hornik, A. Zeileis, kernlab – an s4 package for kernel methods in R, J. Stat. Softw. 11 (2004) 1–20.
[69] G. Bo, H. Xianwu, SVM multi-class classification, J. Data Acquisit. Process. 21 (3) (2006) 334–339.
[70] O. Schütze, X. Esquivel, A. Lara, C. Coello, Using the averaged Hausdorff distance as a performance measure in evolutionary multiobjective optimization, IEEE Trans. Evol. Comput. 16 (4) (2012) 504–522.