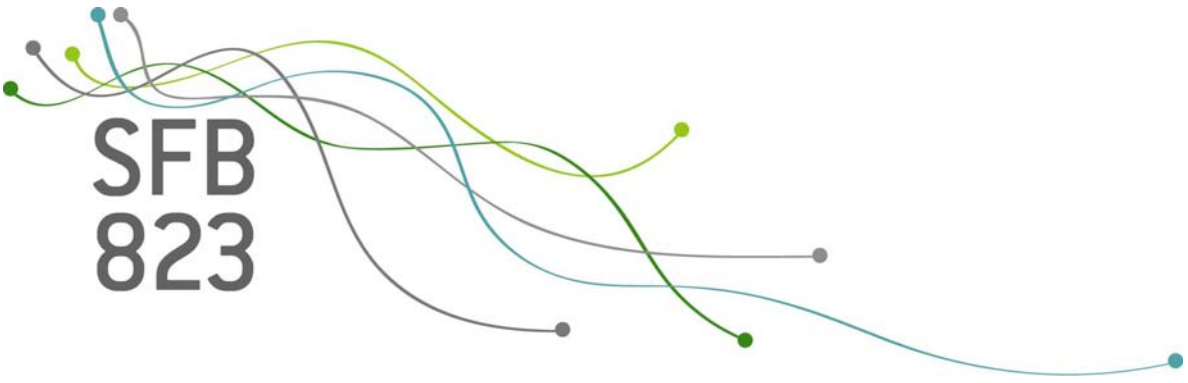
SFB 823

Multi-objective parameter configuration of machine learning algorithms using model-based optimization

Daniel Horn, Bernd Bischl

Nr. 68/2016

Discussion Paper



Multi-Objective Parameter Configuration of Machine Learning Algorithms using Model-Based Optimization

Daniel Horn[1]* and Bernd Bischl[2]

[1] TU Dortmund, Computational Statistics, 44227 Dortmund, Germany
daniel.horn@tu-dortmund.de
[2] LMU München, Computational Statistics, 80539 München, Germany
bernd.bischl@stat.uni-muenchen.de

Abstract. The performance of many machine learning algorithms heavily depends on the setting of their respective hyper-parameters. Many different tuning approaches exist, from simple grid or random search approaches to evolutionary algorithms and Bayesian optimization. Often, these algorithms are used to optimize a single performance criterion. But in practical applications, a single criterion may not be sufficient to adequately characterize the behavior of the machine learning method under consideration and the Pareto front of multiple criteria has to be considered. We propose to use model-based multi-objective optimization to efficiently approximate such Pareto fronts.

Furthermore, the parameter space of many machine learning algorithms not only consists of numeric, but also categorical, integer or even hierarchical parameters, which is a general characteristic of many algorithm configuration tasks. Such mixed and hierarchical parameter space structures will be created naturally when one simultaneously performs model selection over a whole class of different possible prediction algorithms. Instead of tuning each algorithm individually, our approach can be used to configure machine learning pipelines with such a hierarchical structure, and efficiently operates on the joint space of all considered algorithms, in a multi-objective setting.

Our optimization method is readily available as part of the mlrMBO R package on Github. We compare its performance against the TunePareto R package and the regular Latin Hypercube Sampling. We use a pure numerical setting of SVM parameter tuning and a mixed, hierarchical setting in which we optimize over multiple model spaces at once.

 We acknowledge partial support by the Collaborative Research Center "Statistical modeling of nonlinear dynamic processes" (SFB 823) of the German Research Foundation (DFG).

1 Introduction

Many machine learning problems, can be solved by a large number of different algorithms. Moreover, most of these algorithms have several associated hyper-parameters with a significant influence on their respective performances. The problem to efficiently obtain a well-performing algorithm configuration for a given machine learning problem is known as tuning, hyper-parameter optimization or model selection. In most applications it is sufficient to compare different algorithmic configurations with respect to a single performance measure and to choose the hyper-parameter setting according to the optimal performance value. However, in some applications multiple performance measures have to be considered, e.g. runtime vs. predictive power, model sparsity vs. predictive power or multiple measures from ROC analysis. A few most relevant references to multi-objective machine learning are [1–3]. Although the single-objective case of algorithm configuration and hyper-parameter tuning has been rather well-studied, there is considerably less work on multi-objective model selection. Advanced and efficient techniques for single-objective parameter optimization are often based on evolutionary algorithms, iterative racing procedures [4] (see, e.g., [5] where an approach is presented to configure modeling pipelines for survival analysis) or model-based optimization techniques [6–8], also known as Bayesian optimization. In the latter approach, a surrogate machine learning regression model, often a Gaussian process, is sequentially used to model the relation between parameters and performance output. One proposes new points by optimizing an infill criterion defined on the model. The proposed point(s) are subsequently evaluated, the model is updated, and the procedure iterates, often until the available budget is depleted or a predefined output or model-quality is reached. The approach is linked to the field of surrogate assisted optimization [9].

In many practical settings only a restricted budget is spendable. For example, the arise of Big Data confronts many machine learning techniques with new expensive parameter configuration problems. A single training of a Support Vector Machine (SVM) on a data-set containing less than a million observations can take several hours. Tuning its hyper-parameters on such data-sets using evolutionary or racing techniques is not feasible. Therefore we focus on model-based techniques, but extend them for multi-objective problems. In 2016 we used these techniques to compare the runtime and the predictive power of different approximate SVM solvers [10].

We are currently aware of two external approaches and packages to solve such problems in multi-objective hyper-parameter optimization: The TunePareto package [11] and the MSPOT approach from the SPOT package [12]. Moreover, [13] shows how to perform model-based multi-objective optimization on noisy machine learning problems.

TunePareto provides multiple multi-objective optimization strategies for both categorical and numeric parameter spaces. If the parameter space only contains a (small) finite number of possible values, TunePareto suggests to perform a full search of all possible combinations. In case of larger parameter spaces and numeric parameters, different sampling strategies ranging from uniform sampling

over latin hypercube sampling towards quasi-random low-discrepancy sequences can be applied to draw a subset of all possible parameter configurations. Furthermore, for numerical parameter spaces TunePareto provides a variation of the popular NSGA-II [14]. an evolutionary multi-objective optimization algorithm.

SPOT (Sequential Parameter Optimization Toolbox) provides different model-based approaches for single-objective optimization, including the handling of categorical parameter spaces. Furthermore a multi-objective method named MSPOT is included. Since [15] shows that MSPOT does not perform better than competing MBMO algorithms, we will not focus on MSPOT here.

Often, normal hyper-parameter tuning is run for each considered machine learning algorithm individually and the best algorithm and its configuration are selected at the end. One disadvantage of this approach is that a potentially large amount of runtime is spent on tuning inferior algorithms. This fact could have been automatically learned from the obtained data during the optimization much earlier. Current algorithm configuration approaches like IRACE [4] and SMAC [16] can naturally deal with this problem, as they allow the optimization over hierarchical parameter spaces. If the configuration problem contains discrete choices for the selection of certain methods, special categorical selection parameters are introduced that control which method should be used. In our case that might be the selected machine learning algorithm, but it could also refer to the kernel of an SVM or the applied pre-processing algorithm. The individual methods' parameters are made dependent on this choice and are usually called subordinate parameters. This results in a rather complex parameter space, with often continuous, integer, categorical and subordinate parameters. SMAC deals with this problem by giving up the Gaussian process as a surrogate model and instead uses a random forest. We follow the same approach here and propose to operate on the joint parameter space of all learners.

We propose to use sequential model-based multi-objective optimization (MBMO) algorithms for parameter configuration problems. Our contributions in this paper are as follows:

- We extend the MBMO procedure for mixed and hierarchical parameter spaces.
- We perform a benchmark study demonstrating that MBMO algorithms outperform pure sampling and evolutionary multi-objective techniques from the TunePareto package for both a standard numerical and a mixed hierarchical parameter space.
- We present a software framework written in R that allows a precise definition of the parameter configuration problem and its optimization using MBMO algorithms.

In this paper we will focus on the special case of classification in machine learning. However, all presented methods can easily be applied to any machine learning tuning problem where performance is empirically measurable or even general algorithm configuration in arbitrary contexts.

In section 2 we give a general introduction to MBMO and explain how to use MBMO algorithms in the context of parameter configuration. In section 3

we conduct two experiments to compare the performance of our MBMO toolbox against TunePareto. Section 4 demonstrates the usage of our R implementation, the paper is concluded in section 5.

2 Model-Based Multi-Objective Optimization for Mixed Parameter Spaces

Multi-objective (blackbox) optimization refers to an optimization setting with a vector-valued objective function $\mathbf{f} = (f_1, f_2, \ldots, f_m) : \mathcal{X} = \mathcal{X}_1 \times ... \times \mathcal{X}_d \to \mathbb{R}^m$, with the usual deal that each f_i should be minimized. Here, we do not only allow $\mathcal{X}_i$ to be numeric, i.e., $\mathcal{X}_i = [x_i^{(l)}, x_i^{(u)}] \subset \mathbb{R}$, but also to be integer ($\mathcal{X}_i = \{x_i^{(l)}, ..., x_i^{(u)}\} \subset \mathbb{Z}$) or categorical ($\mathcal{X}_i = \{v_i^{(1)}, ..., v_i^{(s)}\}$).

In general, multiple objectives are contradicting, and we are interested in computing the Pareto front of all possible trade-offs. A solution $\mathbf{x} \in \mathcal{X}$ is said to dominate solution $\mathbf{x}' \in \mathcal{X}$ if $\mathbf{x}$ is at least as good as $\mathbf{x}'$ in all and strictly better in at least one objective. This relation defines only a partial order, allowing the case of incomparable solutions. The dominance relation is sufficiently strong for a definition of optimality: a solution $\mathbf{x}$ is called Pareto optimal if and only if it is not dominated by any other solution $\mathbf{x}'$. The set $\left\{\mathbf{f}(\mathbf{x}) \,\middle|\, \mathbf{x} \in \mathcal{X} \text{ is Pareto optimal}\right\}$ of all non-dominated solutions is called the Pareto front.

In the last 10 years many MBMO algorithms have been proposed. Most of these approaches are based on ideas of the popular Efficient Global Optimization (EGO) procedure for expensive black-box optimization [17]. The EGO algorithm works by sequentially fitting a surrogate regression model on all so far obtained experimental design points. In each iteration the model is refitted and optimized with regard to a so-called infill criterion. A new candidate point is proposed and evaluated with the true, expensive objective and the whole procedure iterates until a stopping criterion is reached, usually a constraint on time or number of function evaluation.

Two popular MBMO algorithms are ParEGO [18] and SMS-EGO [19]. An overview of alternative MBMO algorithms including their taxonomy is available in [15]. Due to the general availability of parallel computing power and the advantages of performing experiments in batches, allowing more than one point evaluation per sequential iteration (batch processing) is of great interest. Especially in context of machine learning with the usage of parallel computing it is easy to evaluate many parameter settings at the same time. As most MBMO algorithms lack a mechanism to propose more than one point per iteration, we also proposed multi-point strategies for some algorithms, including ParEGO and SMS-EGO [15].

Algorithm 1 shows the pseudo code of a general MBMO algorithm allowing the proposal of multiple points in parallel. First, an initial design of size n_{init} is evaluated on the actual, expensive objective function. Afterwards a set of l surrogate models is fitted on the design. A set of t new candidates is generated by optimizing a set of infill criteria based on the surrogate models. Finally the

Algorithm 1 Model-based multi-objective optimization.

Require: expensive function $\mathbf{f} : \mathcal{X} \to \mathbb{R}^m$, batch size t, size of initial design n_{init}, evaluation budget

1: generate *initial design* $\mathcal{D} := \{\mathbf{x}_1, ...\mathbf{x}_{n_{init}}\} \subset \mathcal{X}$
2: compute $Y := \mathbf{f}(\mathcal{D}) = [\mathbf{f}(\mathbf{x}_1), ..., \mathbf{f}(\mathbf{x}_{n_{init}})]^T$
3: **while** evaluation budget not exceeded **do**
4: fit *surrogate models* $\hat{\mathbf{f}} = \{\hat{f}_1, ..., \hat{f}_l\}$ on $\mathcal{D}$ and Y
5: get $\{\mathbf{x}_1^*, ..., \mathbf{x}_t^*\}$ by *optimizing* some *infill criteria* on $\hat{\mathbf{f}}$
6: evaluate $\mathbf{y}_i^* := \mathbf{f}(\mathbf{x}_i^*), i = 1, ..., t$
7: $\mathcal{D} := \mathcal{D} \cup \{\mathbf{x}_1^*, ..., \mathbf{x}_t^*\}$
8: $Y := [Y^T, \mathbf{y}_1^{*T}, ..., \mathbf{y}_t^{*T}]^T$
9: **return** Pareto front and set of Y and $\mathcal{D}$

new candidates are evaluated on the real objective function and added to the design.

Now we will further explain the cursive components in Algorithm 1. They can be instantiated to form variants of MBMO algorithms, as we will demonstrate for ParEGO and SMS-EGO. Moreover, most MBMO algorithms were originally proposed only for numeric parameter spaces. With our here proposed adjustments we enable every MBMO algorithm to work on mixed, hierarchical parameter spaces.

Initial Design In principle, all available design-of-experiment techniques can be used. Very often, space filling techniques like Latin Hypercube Sampling (LHS) are applied, as we also do for both ParEGO and SMS-EGO here.

Since LHS itself is only defined for numeric parameter spaces, it has to be extended towards mixed, hierarchical parameter spaces. We propose to use a thinning approach, based on distances, to generate an initial design of size n_{init}: A purely random design of size $n_{exh} \gg n_{init}$ is sampled and reduced by removing a random point with minimal distance to another point until only n_{init} points remain. We use the Gower distance function [20] for mixed parameter spaces.

Surrogate Models Although any regression model could be used, almost all MBMO algorithms use Kriging, including ParEGO and SMS-EGO. Different from EGO, most MBMO algorithms have to fit more than one model per iteration since multiple objective have to be taken into account. For example, SMS-EGO fits an individual Kriging model for each objective function (here: $t = m$).

ParEGO follows a different approach. The algorithm fits a single Kriging model to the scalarized objective functions. This is done by applying the Augmented Tschebbyscheff norm with a uniformly sampled weight vector w. In order to propose a batch of size t, we suggested to sample t different weight vectors per iteration [15] in a stratified manner. New design points are selected by optimizing the infill criteria on the different scalarizations separately (here: $t = l$).

Kriging itself is defined for numeric parameter spaces only. Although there are several approaches how to generalize Kriging towards mixed parameter spaces

(e.g., see [21]), they are not frequently used. For the single objective case the SMAC toolbox uses a random forest for mixed parameter spaces [16]. A random forest is an intuitive choice since it has the ability to learn complex non-linear relationships. It also offers several possible estimators for the uncertainty of new observations, which is needed for the calculation of most infill criteria. One simple approach is to use the standard deviation of the mean proposals of all trees in the forest. We adopt this idea for the multi-objective case.

Due to the possibly hierarchical structure of the parameter space, some parameters may be inactive for a specific design point. We simply treat these inactive values as missing and use standard missing value imputation techniques to handle them with the random forest. Since each missing value refers to an area in the parameter space, where the behavior of the target function is rather different (since a different method is used), we replace missing values with values that indicate this different behavior. Missing values in numeric and integer parameters $\mathcal{X}_i$ are replaced by $\max + 2(\max - \min)$, where max and min are the maximal and minimal values of all already evaluated configurations of $\mathcal{X}_i$. This encodes the missing values outside of the range of normal parameters and therefore preserves the information that the values were indeed missing. Missing values in categorical parameters are simply coded as a new level missing.

Infill Criterion Although there are many different single objective infill criteria, only 2 are frequently used: the expected improvement (EI) and the (lower) confidence bound (CB) [6]. Since ParEGO performs a single objective (scalarized) optimization, it can use all of these criteria. Although [18] proposed to use the EI, we showed that it is more promising to use the CB [15]. In addition to those single objective criteria there are several specialized multi-objective infill criteria like the expected hyper-volume improvement [22] and the direct indicator based (DIB) approaches. SMS-EGO is one of the DIB approaches: New design points are chosen by maximizing the contribution of their CB-value to the current Pareto front approximation. The contribution is measured using the non-dominated hypervolume.

There is no need to adapt the infill criteria to the case of mixed, hierarchical parameter spaces, since all existing criteria depend only on the set of surrogate models and the corresponding uncertainty estimators.

Infill Optimization In each iteration of each MBMO algorithm the respective infill criterion has to be optimized. Since it is cheap to evaluate the infill criterion (it is only based on model prediction), a large budget of evaluations can be invested. Therefore the choice of the infill optimizer does not seem critical. Both ParEGO and SMS-EGO use a variant of the CMA-ES [23] for their infill optimization. Since the CMA-ES is not able to handle discrete parameters it has do be replaced by a more flexible optimizer.

In our R-toolbox mlrMBO we use a different approach. In order to reduce call stack overhead, it is much more efficient to be able to query model prediction for many different points in parallel in each optimizer iteration. Therefore we use an optimization strategy we call focus search, that calculates predictions for large

Algorithm 2 Infill optimization: focus search.

Require: cheap black box function $g : \mathcal{X} \to \mathbb{R}$, number of restarts $n.restarts$, number of focus iterations $n.iters$, number of random search points $n.points$

for $u \in \{1, ..., n.restarts\}$ **do**
  Set $\tilde{\mathcal{X}} := \mathcal{X}$
  for $v \in \{1, ..., n.iters\}$ **do**
    generate random design $\mathcal{D} \subset \tilde{\mathcal{X}}$ of size $n.points$
    compute $\mathbf{x}^*_{u,v} = (x^*_1, ..., x^*_d) := \arg\min_{\mathbf{x} \in \mathcal{D}} g(\mathbf{x})$
    for $\tilde{\mathcal{X}}_i$ in $\tilde{\mathcal{X}}$ **do**
      if $\tilde{\mathcal{X}}_i$ numeric: $\tilde{\mathcal{X}}_i = [\tilde{x}_i^{(l)}, \tilde{x}_i^{(u)}]$ **then**
        $\tilde{x}_i^{(r)} := \frac{1}{4}(\tilde{x}_i^{(u)} - \tilde{x}_i^{(l)})$
        $\tilde{x}_i^{(l)} := \max\{\tilde{x}_i^{(l)}, x_i^* - \tilde{x}_i^{(l)}\}$
        $\tilde{x}_i^{(u)} := \min\{\tilde{x}_i^{(u)}, x_i^* + \tilde{x}_i^{(u)}\}$
      if $\tilde{\mathcal{X}}_i$ categorical: $\tilde{\mathcal{X}}_i = \{\tilde{v}_i^{(1)}, ..., \tilde{v}_i^{(s)}\}$, $s > 2$ **then**
        $\bar{x}_i :=$ sample uniformly from $\tilde{\mathcal{X}}_i \backslash x_i^*$
        $\tilde{\mathcal{X}}_i := \tilde{\mathcal{X}}_i \backslash \bar{x}_i$
Return $\mathbf{x}^* := \underset{u \in \{1,...,n.restart\}, v \in \{1,...,n.iters\}}{\arg\min} g(\mathbf{x}^*_{u,v})$

chunks of points. Due to its simple and flexible structure, focus search is also able to handle mixed, hierarchical parameter spaces. The corresponding pseudo code is given in Algorithm 2. The main idea is an iterated and sequentially refined random search using $n.points$ random points. After each of $n.iters$ random search iteration, the constraints of the feasibility region are focused around the current best point. Due to the stochastic of this procedure and to handle multi-modality of the infill criterion, the procedure is restarted $n.restarts$ times.

3 Experiments

As an example of multi-objective parameter configuration, we consider the bi-objective minimization of the false-negative and the false-positive rate (FNR and FPR) in binary classification. We assume that all considered classifiers produce binary labels in prediction, but can be controlled either via weighting classes or individual observations, to reweigh preferences for one class or the other. We are interested in computing the complete Pareto front in order to obtain the best parameter settings for all potential trades-offs for the two objectives.

As a general strategy for balancing FNR and FPR, class weighting can be used. Without loss of generality it is sufficient to adapt the weight ω for the positive class. This is done by assigning each observation of the positive class the weight ω during model fit, i.e., during the minimization of the loss function, while we assign a weight of 1 to each element of the negative class. The larger the value of ω, the more importance is assigned to observations of the positive class, which in the limit results in an eventual FPR of 1. On the other hand, reducing ω to 0 results eventually in FPR = 0.

Table 1. Description of the used data sets. #obs is the number of observations and #feats is the number of features in the respective data sets. The class ratio gives the proportional size of the smaller class in comparison to #obs, i.e., a class ratio of 0.5 indicates that both classes are of equal size. The did is the unique data set identifier on the OpenML platform.

name	#obs	#feats	class ratio	did
ada_agnostic	4 562	48	0.25	1043
eeg-eye-state	14 980	14	0.45	1471
kdd_JapaneseVowels	9 961	14	0.16	976
mozilla4	15 545	5	0.33	1046
pendigits	10 992	16	0.10	1019
phoneme	5 404	5	0.29	1489
spambase	4 601	57	0.39	44
wind	6 574	14	0.47	847
waveform	5 000	40	0.34	979

In the first experiment we tune the objectives FNR and FPR of a support vector machine (SVM) with an RBF kernel. We compare the performance of four multi-objective tuning algorithms on this problem: From the TunePareto R package we chose the latin and the evolution strategy. The simple latin strategy performs LHS sampling. The more advanced evolution strategy is based on the well-known NSGA-II approach, but adapts both the recombination and the mutation strategy (see [11] for more details). From our toolbox mlrMBO we chose SMS-EGO and ParEGO fused with the CB infill criterion. SMS-EGO is run in its sequential variant while ParEGO is forced to propose 4 points in each iteration. The advantage here is that all 4 points can be evaluated in parallel and the algorithm can gain a theoretical speed up of factor 4.

In the second experiment we optimize a joint parameter space of three base learners: an SVM, a random forest and a regularized logistic regression. Unfortunately TunePareto is not able to handle such complex parameter spaces, therefore we are only able to use ParEGO and SMS-EGO. As a baseline we also run a simple latin hypercube sampling.

We run the experiments on several data sets (see table 1), all of them are available on the open machine learning platform OpenML [24]. To obtain a bearable overall execution time for our experiments, we decided to use only mediocre sized data-sets. However, we think that the results can be transferred to larger data-sets. FNR and FPR are measured by 10-fold cross validations. For an unbiased estimation of the Pareto front we use a nested resampling strategy: The tuning is performed on only 50% of the data points, the remaining 50% are used for a-posteriori test evaluation. In this paper we only report measures based on the test evaluations.

We perform 10 replications per algorithm and data set. Each tuning algorithm is presented the same cross-validation splits for each statistical replication. To analyze our results we normalize the resulting decision vectors $\tilde{Y} = [\mathbf{y}_1^T, ..., \mathbf{y}_n^T]^T$, where $\tilde{Y}$ is the union over the results of all algorithms for each data set and

replication. We apply the transformation $\frac{\mathbf{y}_i - \min \mathbf{y}_i}{\max \mathbf{y}_i - \min \mathbf{y}_i}$, so that the corresponding Pareto front of each objective and replication has the range $[0, 1]$. We then compute the dominated hypervolume (with reference point $(1.1, 1.1)$) and the 50% empirical attainment function (eaf) [25] for all algorithms and data sets. To compare the performances of the various algorithms we folow the procedure proposed by Demsar [26]. We perform Friedman tests to detect significant differences in the performances followed by post-hoc Nemenyi tests. For further insight into the data we do also perform pairwise Wilcoxon tests with paired samples. All tests are performed with significance level $\alpha = 0.05$.

The experiments are conducted using our own software packages written in R [27], including (but not only) mlrMBO [28], mlr [29] and BatchExperiments [30] and were executed on the LiDOng cluster of TU Dortmund university.

3.1 Tuning a support vector machine

We tune an SVM with RBF kernel, i.e., we optimize the cost parameter C, the inverse kernel width γ and the class weighting parameter ω ($d = 3$). The region-of-interest for C and γ is defined as $2^{[-15,15]}$, for ω we use the interval $2^{[-7,7]}$. All parameters are optimized on a logarithmic scale.

We consider a total budget of 160 ($\approx 50d$) function evaluations. For both ParEGO and SMS-EGO we use the same random LHS as initial design with size 30 ($= 10d$). As surrogate we use a Kriging model with matern-$\frac{3}{2}$ kernel and a small nugget effect (10^{-4}) for numerical stability. We force ParEGO to propose 4 points in each iteration, SMS-EGO is run in its sequential variant. For our focus search we set $n.iters = n.restarts = 3$ and $n.points = 1000$. For TunePareto's evolution-strategy we set $\mu = \lambda = 20$.

The results are shown in the left columns of both Fig. 1. On all test functions both SMS-EGO and ParEGO are at least as good as latin and evolution, most times both MBMO algorithms outperform both baselines. The boxplots show higher hypervolume values on nearly all data sets, the median eaf-plots do confirm the hypervolume values. It appears to be difficult to give a ranking between the two MBMO algorithms, only on the wind data set we observe that ParEGO performs better than SMS-EGO. The differences between latin and evolution are much clearer on some data sets, but since they take turns in dominating each other it is also impossible to state a winner here.

The Friedman test gives a p-value of 0.001, therefore we have a strong indicator that the differences in the figures are significant. In table 2 the results of the post-hoc Nemenyi tests and the paired Wilcoxon tests are shown. The tests mainly confirm the impression we got from the figures, but here the results are even more distinct. Apparently some variance caused by the different resampling instances was eliminated in the pairing process. Both ParEGO and SMS-EGO are not significantly worse than the baselines on all 9 test functions, and they are significantly better on 6 to 8 test functions. However, the Nemenyi test does not show a significant difference between ParEGO and evolution. Comparing SMS-EGO and ParEGO the Nemenyi test does not show a significant difference, only on 2 data sets differences were detected by the Wilcoxon test. Comparing latin

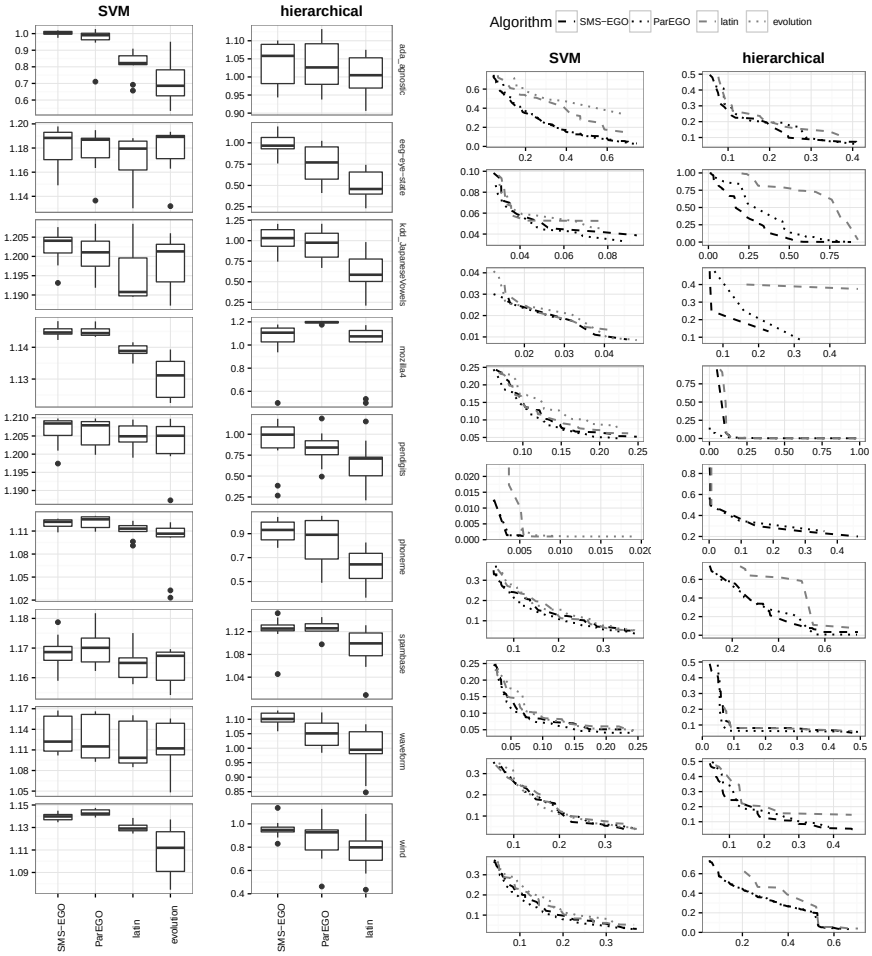


Fig. 1. Left figure: Hypervolume values for both experiments and all data sets. Right figure: 50% empirical attainment functions of the two objectives FNR and FPR for both experiments, the axis limits were focused on the interesting parts of the Pareto fronts. The order of the data sets is the same as in the left figure.

Table 2. Results of the paired pairwise t-Tests ($\alpha = 0.05$) for the SVM experiment. + indicates the first, - the second algorithm was significant better, otherwise 0 is displayed. The Friedman test for equal median hypervolume values rejected its null hypotheses with an p-value of 0.001, the last line of the table shows the p-values of the post-hoc Nemenyi tests.

	SMS-EGO vs ParEGO	SMS-EGO vs latin	SMS-EGO vs evolution	ParEGO vs latin	ParEGO vs evolution	latin vs evolution
ads_agnostic	+	+	+	+	+	+
eeg-eye-state	0	+	+	0	0	-
kdd_JapaneseV...	0	+	+	0	0	-
mozilla4	0	+	+	+	+	+
pendigits	0	0	0	+	+	0
phoneme	0	+	+	+	+	0
spambase	0	+	+	+	+	0
waveform	0	+	+	+	0	0
wind	-	+	+	+	+	+
p-Value	0.885	0.003	0.031	0.031	0.185	0.885

and evolution we see 3 wins for latin, 2 wins for evolution and 4 draws. In this case, the Nemenyi test does also not find a significant difference.

Overall we can state that both MBMO algorithms outperform the baselines from the TunePareto package. Although ParEGO was forced to propose 4 points in each iteration, which should be a disadvantage in the optimization (new information is not included every single, but every 4th iteration), it was able to reach the same performance as SMS-EGO.

3.2 Tuning over multiple models with hierarchical structure

Now we consider a joint model space containing three base learners: an SVM, a random forest and an L2-regularized logistic regression. The hierarchical structure of the corresponding parameter set (containing numeric, integer and discrete parameters) is displayed in Fig. 2. Note that most parameters are dependent on the discrete parameter learner.

We try to use almost the same settings as in the first experiment. However, to handle the mixed parameter spaces, some changes have to be done: to accurately sample the larger parameter space we increased the size of the initial design to 60 and the overall budget to 300 points. This corresponds to $10\tilde{d}$ and $50\tilde{d}$, where $\tilde{d} = 6$ is the number of numeric and integer parameters in $\mathcal{X}$. We use our thinning approach as initial design with an oversampled design of size $n_{exh} = 10n_{init}$. As surrogate model we use a random forest containing 500 trees, its uncertainty is estimated by the standard deviation of the single trees mean responses.

The results are displayed in the right columns of Fig. 1. As in the SVM experiment we can see on all data sets that both MBMO algorithms are at least

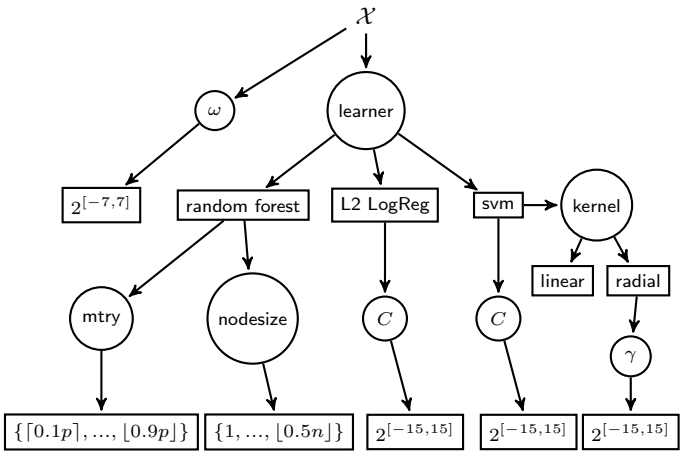


Fig. 2. Joint parameter space of the second experiment. Circles denote variables, rectangles denote values of numerical and discrete parameters, arrows denote the hierarchic structure. Here n denotes the number of observations, p the number of features in the respective data set.

as good as the baseline latin hypercube sampling, and they outperform it on most data sets. Since the joint parameter space in this experiment is a super set of the parameter space in the SVM experiment, the optimal hypervolume value in this experiment is at least as high as in the SVM experiment. Unfortunately on most data sets much lower hyper-volume values are reached. This can be explained by the much larger search space and the severely restricted budget, but might indicate room for further for improvements. In contrast to the SVM experiment, here, SMS-EGO seems to be a bit better than ParEGO.

The observations from the figures are confirmed by the results of the tests in table 3. The Friedman test finds significant differences with a p-value of 0.0006, the Nemenyi tests show that both ParEGO and SMS-EGO are significantly different from latin. SMS-EGO and ParEGO are significantly better than latin on 8 to 9 data sets.Comparing SMS-EGO and ParEGO we see that SMS-EGO is significantly better than ParEGO on 5 data sets, ParEGO does only win on the *mozilla4* data set, the remaining 3 data sets are draws. Although this difference is not significant according to the Nemenyi test, SMS-EGO seems preferable in this experiment.

4 Implementation in mlrMBO

In order to ensure open and convenient access to our proposed methods, as well as reproducibility of results for studies as in this paper, we have implemented them into the mlrMBO R package [28] for model-based optimization. In this section we will demonstrate its usage for multi-objective parameter configuration of machine learning algorithms where we combine mlrMBO with our mlr toolbox

Table 3. Results of the paired pairwise Wilcoxon tests ($\alpha = 0.05$) for the joined space experiment. + indicates the first, - the second algorithm was significant better, otherwise 0 is displayed. The Friedman test for equal median hypervolume values rejected its null hypotheses with an p-value of 0.0006, the last line of the table shows the p-values of the post-hoc Nemenyi tests.

	SMS-EGO vs ParEGO	SMS-EGO vs latin	ParEGO vs latin
ada_agnostic	0	+	0
eeg-eye-state	+	+	+
kdd_JapaneseVowels	+	+	+
mozilla4	-	+	+
pendigits	0	+	+
phoneme	0	+	+
spambase	0	+	+
waveform	+	+	+
wind	+	+	+
p-Value	0.466	0	0.026

for machine learning in R [29]. Therefore, we pick the same situation as in experiment 1, the simultaneous optimization of the FNR and FPR of an SVM with an RBF kernel, but keep some more advanced settings at their defaults for simplicity.

In the first lines of the example code the machine learning problem is defined. Lines 3 and 4 define the machine learning task, we use the predefined classification problem on the sonar data set. In lines 5 to 7 the SVM is created, we allow reweighing of the classes and optimization of the weighting parameter. The next lines describe how to measure performance for our learning problem and the bi-criteria objective function. Lines 8 to 9 define the data splits, here a 10-fold cross validation. Lines 10–16 specify the objective: the cross-validated FNR and FPR scores of the SVM, for given hyper-parameters x. In lines 17-24 the parameter space is defined. The lines 25-28 bundle all information regarding the optimization problem in a single R object. The last part of the example code defines the optimizer. Here we define a default multi-objective optimization with ParEGO using the confidence bound (CB) infill criterion. In lines 34–36 the surrogate learner (a Kriging model) is constructed. Lines 37–39 sample the initial design using a random LHS, lines 40–42 perform the actual optimization.

```
1 library(mlrMBO)
2 configureMlr(show.learner.output = FALSE)
3 # We use the predefined mlr task sonar
4 task = sonar.task
5 # The learner whose performance we optimize:
6 cl.lrn = makeLearner("classif.svm")
```

```
cl.lrn = makeWeightedClassesWrapper(cl.lrn)
# The validation procedure
rin = makeResampleInstance(cv10, task)
# Function to resample the performance
fn = function(x) {
  lrn = setHyperPars(cl.lrn, par.vals = x)
  resample(learner = lrn, show.info = FALSE,
    task = task, resampling = rin,
    measures = list(fnr, fpr))$aggr
}
# Description of the parameter space
par.set = makeParamSet(
  makeNumericParam("cost", lower = -15,
    upper = 15, trafo = function(x) 2^x),
  makeNumericParam("gamma", lower = -15,
    upper = 15, trafo = function(x) 2^x),
  makeNumericParam("wcw.weight", lower = -7,
    upper = 7, trafo = function(x) 2^x))
# Description of target function
obj.fun = makeMultiObjectiveFunction(
  fn = fn, has.simple.signature = FALSE,
  par.set = par.set, n.objectives = 2L)
# Build the MBO control for multicrit optim.
ctrl = makeMBOControl(n.objectives = 2L)
ctrl = setMBOControlInfill(ctrl, crit = "cb")
ctrl = setMBOControlMultiCrit(ctrl,
  method = 'parego')
# Construct the surrogate learner
mbo.lrn = makeLearner("regr.km",
  predict.type = "se")
# Sample the initial design
design = generateDesign(n = 10L,
  par.set = par.set, fun = randomLHS)
# Start MBO
mbo(fun = obj.fun, design = design,
  learner = mbo.lrn, control = ctrl)
```

5 Conclusion

Automatic parameter configuration is a challenging problem, this holds even more for multi-objective parameter configuration. The corresponding optimization problems are often very expensive, multi-modal and have mixed, hierarchical parameter spaces. Very few optimization algorithms are able to tackle all of these complications at once.

In this paper we proposed to use model-based multi-objective optimization algorithms for parameter configuration tasks. MBMO algorithms so far have been able to handle most of the above complications very well, except for the mixed,

hierarchical parameter spaces. Therefore we presented an extension based on random forests for MBMO algorithms toward those parameter spaces. We presented two parameter configuration experiments. In the first one we optimized the false negative rate and false positive rate over a simple 3-dimensional parameter space of an SVM. In the second one we optimized the same objectives over a more complex joint model space of an SVM, a random forest and an L2 penalized logistic regression. In both experiments the MBMO algorithms ParEGO and SMS-EGO were able to outperform the baselines.

Although the results of the SVM experiment are very promising, the results for the joint model space have room to get even better. In our opinion, especially the choice of the random forest as the surrogate model and its uncertainty estimator need further investigation. Moreover, we ignored that every parameter configuration problem is actually a noisy optimizing problem by fixing the crossvalidation splits. In our future work we plan on including the noise-aspect into the actual MBMO optimization.

References

1. Y. Jin, *Multi-Objective Machine Learning*, ser. Studies in Computational Intelligence. Springer, 2006.
2. R. M. Everson and J. E. Fieldsend, "Multi-class {ROC} analysis from a multi-objective optimisation perspective," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 918 – 927, 2006.
3. L. Graning, Y. Jin, and B. Sendhoff, "Generalization improvement in multi-objective learning," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*. IEEE, 2006, pp. 4839–4846.
4. M. Lopez-Ibanez, J. Dubois-Lacoste, T. Sützle, and M. Birattari, "The irace package, iterated race for automatic algorithm configuration," IRIDIA, Universit Libre de Bruxelles, Belgium, Tech. Rep. TR/IRIDIA/2011-004, 2011.
5. M. Lang, H. Kotthaus, P. Marwedel, C. Weihs, J. Rahnenfhrer, and B. Bischl, "Automatic model selection for high-dimensional survival analysis," *Journal of Statistical Computation and Simulation*, vol. 85, no. 1, pp. 62–76, 2015.
6. D. R. Jones, "A taxonomy of global optimization methods based on response surfaces," *Journal of Global Optimization*, vol. 21, no. 4, pp. 345–383, 2001.
7. C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-weka: Combined selection and hyperparameter optimization of classification algorithms," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13. New York, NY, USA: ACM, 2013, pp. 847–855.
8. P. Koch, B. Bischl, O. Flasch, T. Bartz-Beielstein, C. Weihs, and W. Konen, "Tuning and evolution of support vector kernels," *Evolutionary Intelligence*, vol. 5, no. 3, pp. 153–170, 2012.
9. Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61 – 70, 2011.
10. D. Horn, A. Demircioğlu, B. Bischl, T. Glasmachers, and C. Weihs, "A comparative study on large scale kernelized support vector machines," *Advances in Data Analysis and Classification*, pp. 1–17, 2016.

11. C. Müssel, L. Lausser, M. Maucher, and H. A. Kestler, "Multi-objective parameter selection for classifiers," *Journal of Statistical Software*, vol. 46, no. i05, 2012.
12. M. Zaefferer, T. Bartz-Beielstein, B. Naujoks, T. Wagner, and M. Emmerich, "A case study on multi-criteria optimization of an event detection software under limited budgets," in *Proc. 7th Int'l. Conf. Evolutionary Multi-Criterion Optimization (EMO)*, R. Purshouse *et al.*, Eds. Berlin: Springer, 2013, pp. 756–770.
13. P. Koch, T. Wagner, M. T. Emmerich, T. Bäck, and W. Konen, "Efficient multi-criteria optimization on noisy machine learning problems," *Applied Soft Computing*, vol. 29, pp. 357 – 370, 2015.
14. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.
15. D. Horn, T. Wagner, D. Biermann, C. Weihs, and B. Bischl, "Model-Based Multi-objective Optimization: Taxonomy, Multi-Point Proposal, Toolbox and Benchmark," in *Evolutionary Multi-Criterion Optimization*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2015, vol. 9018, pp. 64–78.
16. F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential Model-Based Optimization for General Algorithm Configuration," in *Proceedings of LION-5*, 2011, pp. 507–523.
17. D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.
18. J. Knowles, "ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 50–66, 2006.
19. W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze, "Multiobjective optimization on a limited amount of evaluations using model-assisted $\mathcal{S}$-metric selection," in *Proc. 10th Int'l Conf. Parallel Problem Solving from Nature (PPSN)*, 2008, pp. 784–794.
20. J. C. Gower, "A general coefficient of similarity and some of its properties," *Biometrics*, vol. 27, no. 4, pp. 857–871, 1971.
21. Q. Zhou, P. Z. G. Qian, and S. Zhou, "A simple approach to emulation for computer models with qualitative and quantitative factors," *Technometrics*, vol. 53, no. 3, pp. 266–273, Aug. 2011.
22. M. T. M. Emmerich, A. Deutz, and J. W. Klinkenberg, "Hypervolume-based expected improvement: Monotonicity properties and exact computation," in *Proc. IEEE Congress on Evolutionary Computation (CEC)*, 2011, pp. 2147–2154.
23. N. Hansen, *The CMA Evolution Strategy: A Comparing Review.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 75–102. [Online]. Available: http://dx.doi.org/10.1007/3-540-32494-1_4
24. J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "OpenML: Networked science in machine learning," *SIGKDD Explor. Newsl.*, vol. 15, no. 2, pp. 49–60, Jun. 2014.
25. V. G. da Fonseca and C. M. Fonseca, "The attainment-function approach to stochastic multiobjective optimizer assessment and comparison," in *Experimental Methods for the Analysis of Optimization Algorithms*, T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, Eds. Berlin Heidelberg: Springer, 2010, pp. 103–130.
26. J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

27. R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2016. [Online]. Available: https://www.R-project.org/
28. B. Bischl, J. Richter, J. Bossek, D. Horn, and M. Lang, "mlrmbo: A toolbox for model-based optimization of expensive black-box functions," 2016.
29. B. Bischl, M. Lang, L. Kotthoff, J. Schiffner, J. Richter, Z. Jones, and G. Casalicchio, *mlr: Machine Learning in R*, R package version 2.9. [Online]. Available: https://github.com/mlr-org/mlr
30. B. Bischl, M. Lang, O. Mersmann, J. Rahnenführer, and C. Weihs, "BatchJobs and BatchExperiments: Abstraction mechanisms for using R in batch environments," *Journal of Statistical Software*, vol. 64, no. 11, pp. 1–25, Mar. 2015.