Multi-objective selection of algorithm portfolios

Daniel Horn, Bernd Bischl, Aydın Demircioğlu, Tobias Glasmachers,
Tobias Wagner, and Claus Weihs

Abstract We propose a method for selecting a portfolio of algorithms optimizing multiple criteria. We select a portfolio of limited size and at the same time good quality from a possibly large pool of algorithms. Our method also helps to decide which algorithm to use for each trade-off between conflicting objectives. Many algorithms depend on a number of parameters and, therefore, require problem-specific tuning for a suitable performance. In multi-objective tuning, different parameter settings of one algorithm will lead to different trade-

Daniel Horn · Claus Weihs
Fakultät Statistik, Technische Universität Dortmund, 44221 Dortmund, Germany

✉ daniel.horn@tu-dortmund.de
✉ claus.weihs@tu-dortmund.de

Bernd Bischl
LMU München, 80539 München, Germany

✉ bernd.bischl@stat.uni-muenchen.de

Aydın Demircioğlu · Tobias Glasmachers
Ruhr-Universität Bochum, 44780 Bochum, Germany

✉ aydin.demircioglu@ini.rub.de
✉ tobias.glasmachers@ini.rub.de

Tobias Wagner
Technische Universität Dortmund, 44221 Dortmund, Germany

✉ wagner@isf.maschinenbau.uni-dortmund.de

ARCHIVES OF DATA SCIENCE, SERIES A
(ONLINE FIRST)
KIT SCIENTIFIC PUBLISHING
Vol. 2, No. 1, 2017

DOI 10.5445/KSP/1000058749/24
ISSN 2363-9881

offs between the conflicting objectives. Hence, discrete approximations of the corresponding Pareto front resulting from different parameter settings of each algorithm must be compared. Our technique is applied post-hoc to these approximations. It discards algorithms that contribute only insignificantly to the overall Pareto front and delivers simple and interpretable decision rules which algorithm to choose based on the desired trade-off. The new method is applied to the selection of approximative support vector machine solvers, where the objectives are high accuracy and short training time. The analysis hints at dropping several solvers completely and yields insights into the specific strengths of the remaining solvers.

1 Introduction

In a world of ever-growing science, new algorithms are frequently published. But how to decide which algorithm to use for solving a given problem? While it is relatively straightforward to choose an algorithm with respect to a single performance measure, this is much harder if multiple contradicting measures must be considered. Moreover, most algorithms have parameters that must be tuned to reach top performance. To understand which algorithm can reach which trade-offs, an offline analysis on representative test problems can be performed. But given such performance data, or possibly multiple replications thereof, how to decide which algorithm is the best one for a specific trade-off? In most cases there will be no single algorithm dominating the complete Pareto front. Instead a portfolio of optimal algorithms must be selected along with a concise decision rule. Our main question can be formulated as follows:

Choose a portfolio of algorithms as small as possible, with a median Pareto front as close to the median Pareto front of all algorithms as possible. Which algorithms should be selected for the portfolio, and which of them should be used for which trade-off?

To our best knowledge there is no method that gives a systematic answer to this question. In this paper we try to fill this gap. We consider a restricted setting, namely the analysis of multiple algorithms applied to a *single* problem instance. In a case study, we apply multiple support vector machine (SVM) solvers to the problem of quickly learning an accurate predictive model for a specific supervised learning problem.

Our question is closely related to the area of algorithm selection (Hutter et al, 2011), where the goal is to make an online recommendation of an algorithm from a portfolio based on *features* of the problem instance. In contrast, in our offline analysis we aim to understand and to analyse the typical trade-offs the different algorithms can reach on typical data sets. Although the possible trade-offs are very likely to be blurred by data-dependent effects, there might be some patterns that can be observed along multiple data sets. For example, algorithm A may be preferable on most data sets, if just one of the objectives is considered. Hence, it could be suggested to use algorithm A on new data sets if minimization of this objective is important. We do not aim at giving concrete instructions, but rather want to give thumb rules which algorithms could be preferable in which situation.

Our method is motivated by the need to select SVM solvers that realize different trade-offs between accuracy and training time. We are interested in identifying certain characteristics of the different SVM solvers, e.g., being particularly suitable for slow, high-accuracy solutions or rather for reaching reasonable accuracy quickly.

The paper is organized as follows. In Sect. 2 we give a short introduction to multi-objective optimization. We present our new method in Sect. 3, which is applied in Sect. 4 to the selection of support vector machine solvers. We conclude in Sect. 5 with a short summary.

2 Multi-Objective Optimization

In this paper we are dealing with multi-objective optimization (MOO)[1], i.e., the optimization of multiple objectives $f_1, \ldots, f_m : \mathscr{X} \to \mathbb{R}$ (to be minimized, w.l.o.g.) or equivalently a vector-valued objective function $\mathbf{f} = (f_1, \ldots, f_m) : \mathscr{X} \to \mathbb{R}^m$, where $\mathscr{X}$ denotes the set of feasible decision vectors. In general the objectives are contradicting, and we are interested in the best achievable trade-offs. For a minimization problem, a solution $\mathbf{x}$ is said to dominate a solution $\mathbf{x}'$ ($\mathbf{x} \preceq \mathbf{x}'$) if

$$\forall i : f_i(\mathbf{x}) \leq f_i(\mathbf{x}') \quad \wedge \quad \exists i : f_i(\mathbf{x}) < f_i(\mathbf{x}') \ .$$

This relation defines a partial order on $\mathscr{X}$ and is therefore adequate to define an optimality criterion: A solution $\mathbf{x}$ is Pareto optimal if it is a minimal element,

[1] see e.g. Ehrgott (2013) for an introduction into multi-objective optimization

i.e., if $\nexists \mathbf{x}' \in \mathscr{X} : \mathbf{x}' \preceq \mathbf{x}$. The set $\mathscr{X}^* = \{\mathbf{x} \in \mathscr{X} \,|\, \mathbf{x} \text{ is Pareto optimal}\}$ is the Pareto set, $\mathscr{Y}^* = \mathbf{f}(\mathscr{X}^*)$ is the Pareto front. The task of MOO algorithms is the approximation of this sets. We denote such approximations with $\hat{\mathscr{X}}^*$ and $\hat{\mathscr{Y}}^*$. Moreover we say that $\mathbf{x}$ is dominated by a set $\mathscr{S}$ if $\exists \mathbf{x}' \in \mathscr{S} : \mathbf{x}' \preceq \mathbf{x}$.

One important task in multi-objective optimization is to rate the goodness of an optimization result $\hat{\mathscr{Y}}^*$. While this is easy in single objective optimization it is unclear how to compare different Pareto fronts, especially when they are incomparable w.r.t. dominance. Since many optimizers are stochastic it is not sufficient to compare single Pareto fronts, but multiple fronts from multiple replications must be considered. In MOO two main approaches do exist:

Performance Indicators: A unary performance indicator assigns a performance score to a Pareto front. This makes arbitrary fronts directly comparable. Multiple independent Pareto fronts simply result in a vector of independent performance measures that can be compared with well known statistical methods. A binary indicator assigns a score to a pair of sets (Pareto fronts). The sign of this value indicates which front it considers superior. If one of the fronts is the "true" (optimal) front then the indicator value is called *optimality gap*. An overview over existing performance indicators is found in Zitzler et al (2003). One disadvantage of performance indicators for our application is that the reduction to a single number necessarily loses a lot of information.

Empirical Attainment Function: A different approach was described by Grunert da Fonseca et al (2001). Instead of mapping $\hat{\mathscr{Y}}^*$ to a single number the idea is to look at the multivariate distribution of $\hat{\mathscr{Y}}^*$ itself, which is captured by the attainment function $a_{\hat{\mathscr{Y}}^*} : \mathbb{R}^m \to [0,1]$, $\mathbf{z} \mapsto P(\hat{\mathscr{Y}}^* \preceq \mathbf{z})$, where P is the probability function. For $|\hat{\mathscr{Y}}^*| = 1$ it is equivalent to the multivariate cumulative distribution function. The attainment function can be estimated via the empirical attainment function (eaf)

$$a_n(\mathbf{z}) = \frac{1}{n} \sum_{i=1}^{n} I\big(\{\hat{\mathscr{Y}}_i^* \preceq \mathbf{z}\}\big) \ ,$$

where $\hat{\mathscr{Y}}_1^*, \ldots, \hat{\mathscr{Y}}_n^*$ denote the Pareto fronts of n independent optimization runs and I is the indicator function. For $q \in [0,1]$, the q-eaf (lower q-quantile) of a_n is the Pareto optimal subset of the topological closure of the set

$$a_n^{-1}\Big(\big[\operatorname*{arg\,min}_{q' \in \{a_n(\mathbf{z}) | \mathbf{z} \in \mathbb{R}^m\}} (|q' - q|), 1\big]\Big) \ .$$

For example, with $q = 50\%$ it describes the median performance of an optimizer. The set has a simple interpretation: for $n \to \infty$ an objective vector z dominates

the 50%-eaf if and only if it was dominated in strictly less than 50% of all optimization runs.

3 Multi-objective selection of algorithm portfolios

We consider the following situation: The performance of a set of k algorithms $\mathscr{A} = \{A_1, \ldots, A_k\}$ shall be compared on a single test problem with respect to m possibly contradicting performance measures, and the best algorithm for each trade-off shall be identified. Here we focus on the common case $m = 2$. Each algorithm has its own set of parameters, the tuning of which is specific to the problem at hand as well as to the desired trade-off between the objectives.

We assume that multi-objective parameter tuning has been performed in advance for each algorithm separately on the test problem, resulting in a Pareto front of optimal trade-offs. Usually there is stochasticity in the tuning process (stochastic optimization) and/or in the performance assessment (e.g., variations in runtime measurements). We assume further that each tuning has been repeated $n > 1$ times, resulting in n independent Pareto fronts for each algorithm. From these we can compute the empirical attainment function, and in particular the median front (50%-eaf). In the following we aim to optimize the median front. However, the median can easily be replaced with another quantile.

An algorithm selection method aims to pick a suitable algorithm $A_j \in \mathscr{A}$. The ideal selector picks an algorithm exactly where its performance is not dominated by any other algorithm – hence its (median) front is composed of non-dominated parts of the (median) fronts of the individual algorithms. We refer to the resulting front as the *common front* of the selector. The goal of multi-objective algorithm selection is to approximate the ideal common front, which consists of the non-dominated subset of the union of the (median) fronts.

In the sequel we propose such a selection method. It takes the $k \cdot n$ Pareto fronts (n repetitions for k algorithms) as input, and it outputs a rule that makes transparent selection decisions among a small set of top-performing algorithms.

Our method is divided into three steps: In the first step we reduce the computational burden of the following steps by removing all dominated algorithms. In the second step we apply well known MOO methods to select the best subset out of the remaining algorithms. In the last step we decide which algorithm to use in which part of the common Pareto front.

Step 1: *Remove dominated algorithms.* The complete removal of dominated algorithms has many benefits: It reduces the computational complexity of subsequent processing steps, it simplifies the resulting selection rule, and an algorithm can be removed from the software.

If the front of an algorithm A_j is completely dominated by the common front then, intuitively, it does not contribute at all to the common Pareto front and can thus be dropped. With multiple repetitions the situation is more complicated since the median front can be dominated while single fronts contain Pareto optimal sections. We cope with this situation by dropping algorithms whose fronts are dominated in at least η out of n repetitions.

Step 2: *Select the best algorithm portfolio.* In order to obtain interpretable results we pursue the goal of finding a small subset of algorithms of the best possible quality. This is in itself a bi-objective problem, where we aim to minimize the size of the subset and to maximize its quality. To assess the quality of a *subset of algorithms* we resort to standard MOO techniques based on quality indicators: We minimize the gap between the Pareto front generated by the subset and the full set of algorithms, measured by a binary quality indicator. Any binary indicator (or the binary version of an unary indicator) can be used, a popular choice is the dominated hypervolume (also called S-metric). In accordance with our goal to optimize median performance this calculation is based on the 50%-eaf. Because both the optimization results and the 50%-eaf are just sets of points, we can use the latter ones as input to the performance indicator.

Our goal of finding a small subset $\mathscr{S}$ of algorithms from $\mathscr{A}$ with the best possible quality, translates into the two objectives low cardinality $f_c(\mathscr{S})$ and high quality $f_q(\mathscr{S})$. To pick a single point (corresponding to an algorithm subset $\mathscr{S} \subset \mathscr{A}$) from the resulting Pareto front we use a standard scalarization approach, namely minimization of the augmented Tschebyscheff norm

$$u_{\mathbf{w}} = \max\left\{w_c f_c(\mathscr{S}), w_q f_q(\mathscr{S})\right\} + \rho\left(w_c f_c(\mathscr{S}) + w_q f_q(\mathscr{S})\right) \quad , \qquad (1)$$

where $w = (w_c, w_q) \in [0,1]$, $w_c + w_q = 1$ is a predefined weight vector and ρ a small constant (0.05 in our experiments). The best subset is then defined by $\mathscr{S}^* = \arg\min\left\{u_{\mathbf{w}}(\mathscr{S}) \,\middle|\, \emptyset \neq \mathscr{S} \subset \mathscr{A}\right\}$.

Step 3: *Obtain a concise decision rule.* The goal of this step is to obtain a simple and interpretable rule for the selection of an algorithm.

At first glance the situation is trivial: The common front is defined by a finite set of points, each point belongs to one algorithm, and each algorithm covers single points or connected segments of the common front, with gaps

Algorithm 1 Offline Multi-objective Algorithm Selection

Require: Data D (the Pareto fronts), Hyperparameters η, i_b, $\mathbf{w}$, cp

0) Normalize data D to [0, 1]

1) Remove algorithms that are dominated in more than η replications

2) Select the best subset of algorithms with respect to the optimality gap calculated via the binary performance indicator i_b based on the 50%-eaf and the number of used algorithms. The best subset $\mathscr{S}^*$ is defined by $\mathscr{S}^* = \underset{\mathscr{A} \in (\emptyset \neq \mathscr{S} \subset \mathscr{A})}{\arg\min} \; u_{\mathbf{w}}(\mathscr{S})$.

3) Calculate the nondominated 50%-eaf front of all remaining algorithms. Learn a pruned decision tree with parameter cp to calculate split points between algorithms.

return The common reduced Pareto front

in between. The simplest proceeding is to take the common front as-is and to make the decision with a nearest-neighbor rule. However, in areas where there is no clearly dominant algorithm the nearest-neighbor rule may be fragmented and hard to interpret. We aim to capture the essence of the nearest neighbor rule with a simple (hence interpretable) decision rule. For this purpose we apply a standard classifier, here a pruned decision tree (Breiman et al, 1984):

As we assume a bi-objective problem, we know that for non-dominated points the value of the second objective will decrease while the value of the first one increases. Hence, the solutions of the Pareto front can be indexed with regard to one of the objectives and it is sufficient to train the decision tree based on only one of them. Afterwards the tree is pruned using the complexity parameter cp. Doing so controls the number of switches between algorithms on the front. In order to produce comparable results over multiple data sets the objectives are normalized to the interval $[0, 1]$ before calculating the tree.

Our method has four hyperparameters: the number of non-dominated fronts η in step 1, the choice of the binary multi-objective performance indicator i_b, the weight vector w of the scalarization function in step 2, and the complexity control parameter cp for tree pruning in step 3. All of them are easy to understand and allow to adjust the method to specific desires. A summary of the complete method is given in Algorithm 1. An implementation in R including some example data is available online (Horn, 2015).

4 Application: Approximative SVM solver

Support Vector Machines (SVM) are a state-of-the-art method for classification, combining excellent performance on many problems with a sound mathemat-

SVM solver	description	parameters
LLSVM (Zhang et al, 2012)	Low Rank Approximation	matrix rank
LASVM (Bordes et al, 2005)	Online SMO	ε (accuracy), #epochs
BVM (Tsang et al, 2007)	Enclosing Ball	ε (accuracy)
CVM (Tsang et al, 2005)	Enclosing Ball	ε (accuracy)
LIBSVM (Chang and Lin, 2011)	SMO, the *exact* solver	ε (accuracy)
SVMperf (Joachims and Yu, 2009)	Cutting Plane	ε (accuracy), #cutting planes

Table 1 SVM solvers with parameters that were subject to tuning.

ical foundation (Cortes and Vapnik, 1995). While an SVM by itself is only able to perform linear classification, the use of the so called kernel trick enables SVMs to model non-linear decision boundaries. We focus on the popular Gaussian Radial Basis Function (RBF) Kernel. To reach sensible results the parameters of the SVM, namely the regularization parameter C and the bandwidth of the RBF Kernel γ, must be tuned to the problem (data set) at hand.

The major limitation of kernelized SVMs is the training time. Standard solvers like LIBSVM (Chang and Lin, 2011) are known to scale at least quadratically with the data size (Bottou and Lin, 2007). Many approximation methods have been developed to mitigate this problem. The main idea is to sacrifice a bit of accuracy for a significantly shorter training time. Different approximate solvers realize different trade offs of the two contradicting objectives *prediction accuracy* and *training time*. We applied a logarithmic transformation to the training time, since only its order of magnitude is relevant.

We chose six different implementations that cover a diverse set of different approximation techniques, see Table 1. In addition to the hyperparameters C and γ every solver has its own set of approximation parameters – changing these parameters should lead to different trade-offs between training time and accuracy. In this situation a practitioner needs to know which solver is able to realize which trade-off, respectively, which solvers are the *best* ones.

For each of the six solvers we obtained Pareto fronts by optimizing the parameters C and γ as well as the algorithm-specific approximation parameters. For every evaluation of a parameter configuration an SVM must be trained on a large benchmark problem which is a time-consuming process. For an efficient optimization we applied the model-based multi-objective optimizer ParEGO (Knowles, 2006). To utilize parallel computation on our super computer we used the multi-point extension of ParEGO proposed in Horn et al (2015).

Due to space constraints we cannot give all details of the study here. An exhaustive explanation considering a single replication of each optimization

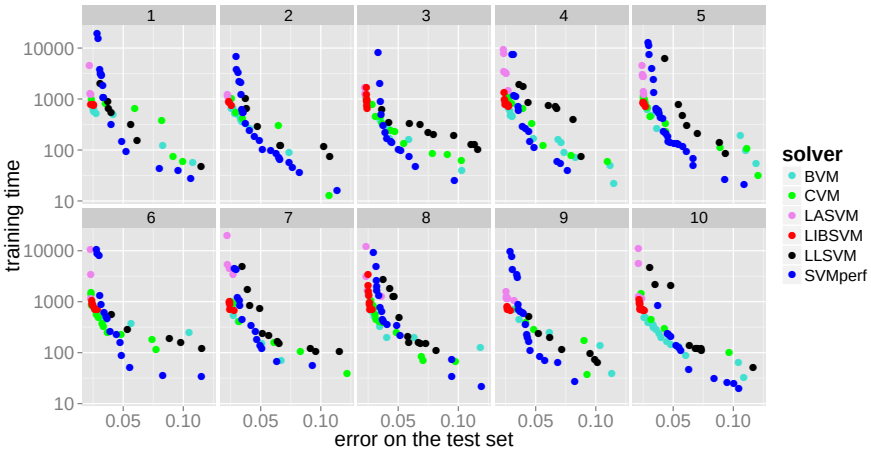

Fig. 1 Pareto fronts of 10 independent optimizations on a binarized version of the MNIST dataset for 6 SVM solvers. Optimization was performed with ParEGO using a budget of 220 SVM trainings.

can be found in Horn et al (2016a). Complete results of ten replications on four selected data sets are available in our R-Package. For these results, 6 solvers $\times$ 10 replications $\times$ 4 data sets $\times$ 220 iterations $=$ 52 800 SVMs had to be trained, resulting in a sequential computation time of nearly four years.

Figure 1 displays the resulting Pareto fronts for the MNIST data set (Chang and Lin, 2011). Some solvers perform clearly better than others. For example, LLSVM is nearly completely dominated. However, based on these plots it is not feasible to pick an optimal portfolio of solvers only through visual inspection.

The 50%-eaf displayed in Fig. 2 provides much clearer insights. It becomes apparent that LLSVM does not affect the common Pareto front, and that SVMperf holds the lion's share of its lower part. But it is still hard to distinguish between the four remaining solvers: LASVM reaches the best error values, while LIBSVM is only slightly behind. Both BVM and CVM fill the gap towards SVMperf, but they are highly redundant and certainly not both of them are needed for a good approximation of the common front. A more sophisticated method is needed to decide which subset of solvers is essential for obtaining (near) optimal trade-offs. Therefore we apply the method proposed in Sect. 3. We discuss the results for the MNIST data set in detail and briefly summarize results across further data sets towards the end of this section.

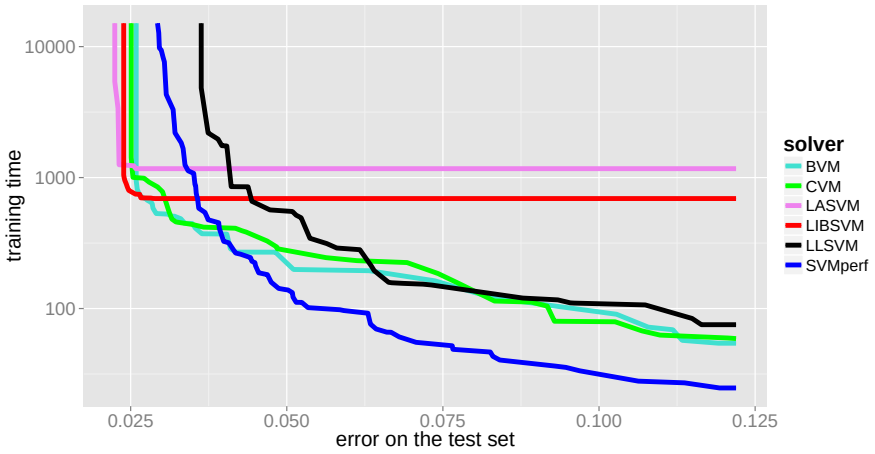

Fig. 2 50%-eaf of the data from Fig. 1.

We configure the algorithm as follows. We set $\eta = \frac{n}{2} = 5$ which is consistent with considering median fronts. With the weight vector $\mathbf{w} = (0.05, 0.95)$ we give far more weight to the objective of minimizing the optimality gap (measured by the hypervolume indicator with reference point $(1.1, 1.1)$, with objective values normalized to the range $[0, 1]$) than to selecting a small set of algorithms. This rather conservative setting tends to keep an algorithm in the portfolio in case of doubt. The pruning parameter of the decision tree is set to $cp = 0.1$, so as to detect only relevant partitionings of the common Pareto front.

Step 1: *Remove dominated algorithms.* Only LLSVM was removed in this step since it produces non-dominated points only in a single repetition. This is in line with the 50%-eaf data.

Step 2: *Select the best algorithm portfolio.* Figure 3 illustrates the selection of the algorithm portfolio. It visualizes the quality of all $31 = 2^5 - 1$ non-empty subsets of algorithms w.r.t. minimization of the optimality gap measured in terms of dominated hypervolume and the number of algorithms. Due to the discrete nature of the second criterion we have one best (non-dominated) subset for each number of algorithms. For example, SVMperf is the best choice for a "portfolio" of size one, and allowing a second algorithm we would add LASVM. The final portfolio is selected from these candidates. The contour lines in Fig. 3

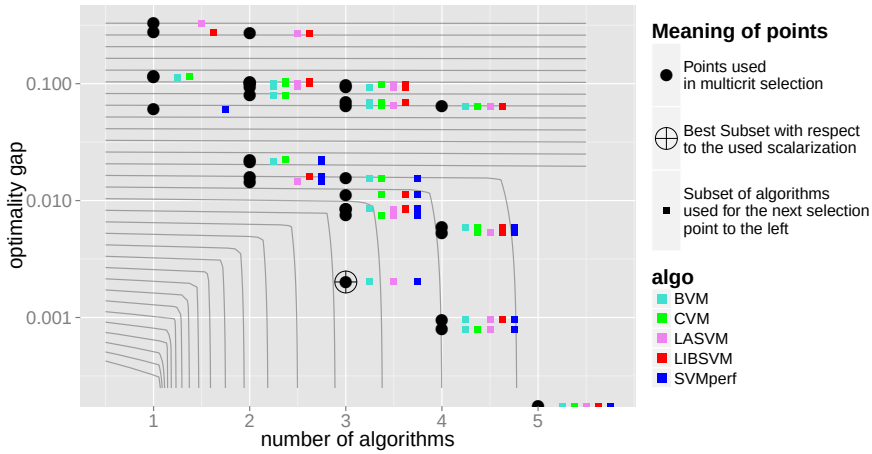

Fig. 3 Selection of the algorithm portfolio, as described in Sect. 3, step 2. Each black dot represents a subset of algorithms, consisting of the algorithms indicated by the color-coded squares to its right. The scalarization function is indicated with contour lines. The black dot marked with a cross-hair minimizes the scalarization function and hence represents the chosen portfolio.

indicate scalarized performance, from which it is easy to see that the portfolio consisting of LASVM, BVM, and SVMperf is the best one.

Step 3: *Obtain a concise decision rule.* From Fig. 2 this step seems trivial since at first glance the three algorithms LASVM, BVM and SVMperf seem to contribute to the common Pareto front from left to right in the given order. However, at a closer look the fronts of BVM and SVMperf have three intersection points. In between the intersection points the performance difference is minimal, hence switching forth and back between algorithms is undesirable. Figure 4 (MNIST & Hypervolume) visualizes the partitioning of the common Pareto front as proposed by the decision tree. Through pruning it reduces the three split points to only one.

The figure also shows the final result of our method. SVMperf covers the lower half the common Pareto front, which was expected already from Fig. 1. In the upper part our method decided to exploit the top-accuracy of LASVM and to use BVM to smoothen the transition between the other two solvers.

It turns out that this result depends on the quality measure. When replacing the hypervolume with the R2 indicator or the ε indicator then in one case

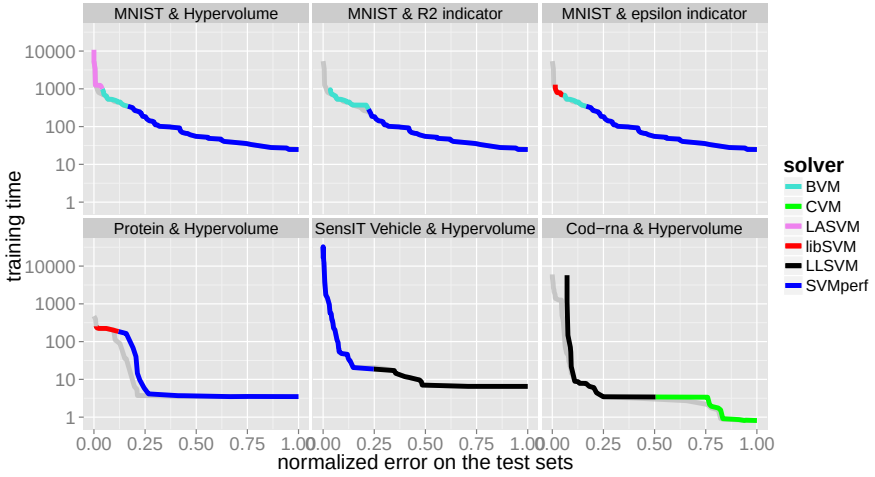

Fig. 4 The final common Pareto fronts for several examples. In the top row results on the MNIST dataset using different performance indicators are shown, in the bottom row results on three more datasets using the hypervolume indicator. The grey line denotes the 50%-eaf front of a portfolio using all six algorithms and can be used to measure the loss of the usage of the smaller subset.

LIBSVM is selected instead of LASVM, and in the other case BVM was used for high-accuracy SVM and neither LIBSVM nor LASVM were selected.

Results for a single data set are not sufficient for judging the overall value of SVM solvers in a portfolio. Figure 4 does also show the final results on three more datasets. Here we see a pretty consistent result - the fronts are made of two out of the three solvers LIBSVM, SVMperf and LLSVM. LIBSVM stands for high accuracy, LLSVM for very fast training and SVMperf for good trade-offs between those two specialized solvers. Complete results for those three datasets can be found in the example section of our R package on github.

5 Conclusion

Offline algorithm selection is relatively easy for single objectives, but gets more complicated for multiple ones. Instead of comparing single performance values, the performances of random sets (the Pareto fronts) must be compared. Also, in

general the best performing algorithm depends on the chosen trade-off, hence different algorithms must be selected in different parts of the Pareto front.

In this paper we have presented an approach for handling multiple objectives. Our method is based on the median empirical attainment function aggregating multiple observations of each Pareto front. By removing uninteresting and redundant algorithms we obtain a small subset of algorithms of the best possible quality that form a single common Pareto front. We obtain an interpretable selection rule in the form of a pruned decision tree.

As an example we applied our method to the selection of SVM solvers. We analyzed in detail how our method excluded dispensable solvers and divided the common Pareto front into distinct parts for the remaining solvers.

Although the results are very promising, our method is tuned for this application. Hence, we proposed a test case generator in Horn et al (2016b) and tested our method in some artificial data situations. In future work it will be tested in more real-world application domains. Furthermore, we aim to generalize the method so that multiple data sets can be analyzed at the same time. We will also work on how to link between a single point on the common Pareto front and the parameter setting of the corresponding algorithm.

Acknowledgements

We acknowledge support by the Mercator Research Center Ruhr, under grant Pr-2013-0015 *Support-Vektor-Maschinen für extrem große Datenmengen* and partial support by the German Research Foundation (DFG) within the Collaborative Research Centers SFB 823 *Statistical modelling of nonlinear dynamic processes*, Project C2.

References

Bordes A, Ertekin S, Weston J, Bottou L (2005) Fast Kernel Classifiers with Online and Active Learning. Journal of Machine Learning Research 6:1579–1619

Bottou L, Lin CJ (2007) Support vector machine solvers. Large scale kernel machines pp 301–320

Breiman L, Friedman J, Olshen RA, Stone CJ (1984) Classification and Regression Trees. Wadsworth and Brooks, Monterey, CA

Chang CC, Lin CJ (2011) LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2:27:1–27:27, URL: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

Cortes C, Vapnik V (1995) Support-vector networks. Machine Learning 20(3):273–297, DOI 10.1023/A:1022627411411

Ehrgott M (2013) Multicriteria optimization, vol 491. Springer, DOI 10.1007/3-540-27659-9

Grunert da Fonseca V, Fonseca CM, Hall AO (2001) Inferential Performance Assessment of Stochastic Optimisers and the Attainment Function, Springer, Berlin, pp 213–225. DOI 10.1007/3-540-44719-9_15

Horn D (2015) URL: https://github.com/danielhorn/multicrit_result_test

Horn D, Wagner T, Biermann D, Weihs C, Bischl B (2015) Model-Based Multi-objective Optimization: Taxonomy, Multi-Point Proposal, Toolbox and Benchmark, Springer International Publishing, Cham, pp 64–78. DOI 10.1007/978-3-319-15934-8_5

Horn D, Demircioğlu A, Bischl B, Glasmachers T, Weihs C (2016a) A comparative study on large scale kernelized support vector machines. Advances in Data Analysis and Classification pp 1–17, DOI 10.1007/s11634-016-0265-7

Horn D, Schork K, Wagner T (2016b) Multi-objective Selection of Algorithm Portfolios: Experimental Validation, Springer International Publishing, Cham, pp 421–430. DOI 10.1007/978-3-319-45823-6_39

Hutter F, Hoos HH, Leyton-Brown K (2011) Sequential Model-Based Optimization for General Algorithm Configuration, Springer, Berlin, pp 507–523. DOI 10.1007/978-3-642-25566-3_40

Joachims T, Yu CNJ (2009) Sparse kernel svms via cutting-plane training. Machine Learning 76(2):179–193, DOI 10.1007/s10994-009-5126-6

Knowles J (2006) ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Transactions on Evolutionary Computation 10(1):50–66, DOI 10.1109/TEVC.2005.851274

Tsang IW, Kwok JT, Cheung PM, Cristianini N (2005) Core Vector Machines: Fast SVM Training on Very Large Data Sets. Journal of Machine Learning Research 6(4)

Tsang IW, Kocsor A, Kwok JT (2007) Simpler core vector machines with enclosing balls. In: Proceedings of the 24th International Conference on Machine Learning, ACM, New York, NY, USA, ICML '07, pp 911–918, DOI 10.1145/1273496.1273611

Zhang K, Lan L, Wang Z, Moerchen F (2012) Scaling up kernel svm on limited resources: a low-rank linearization approach. In: International Conference on Artificial Intelligence and Statistics, pp 1425–1434

Zitzler E, Thiele L, Laumanns M, Fonseca CM, da Fonseca VG (2003) Performance assessment of multiobjective optimizers: An analysis and review. Transactions on Evolutionary Computation 7(2):117–132, DOI 10.1109/TEVC.2003.810758