Technische Universität Dortmund
Fakultät Statistik

Model and Algorithm Selection in Statistical Learning and Optimization.

Dissertation

by

M.Sc. Data Sciences
BERND BISCHL

in partial fulfillment of
the requirements for the degree of
Doktor der Naturwissenschaften

Vorgelegt: Dortmund, Juni 2013
1. Gutachter: Prof. Dr. Claus Weihs
2. Gutachter: Prof. Dr. Jörg Rahnenführer
Kommissionsvorsitz: JProf. Dr. Uwe Ligges
Tag der mündlichen Prüfung: 30. August 2013

I sincerely thank Claus Weihs and Jörg Rahnenführer for supervising my thesis, providing guidance when necessary and the freedom to pursue my own ideas in equal measure.

Many of the things I learned during the last years are due to discussions with many competent colleagues and friends in Dortmund. This thesis would have probably looked very different without them.

Abstract

Modern data-driven statistical techniques, e.g., non-linear classification and regression machine learning methods, play an increasingly important role in applied data analysis and quantitative research. For real-world we do not know a priori which methods will work best. Furthermore, most of the available models depend on so called hyper- or control parameters, which can drastically influence their performance. This leads to a vast space of potential models, which cannot be explored exhaustively. Modern optimization techniques, often either evolutionary or model-based, are employed to speed up this process.

A very similar problem occurs in continuous and discrete optimization and, in general, in many other areas where problem instances are solved by algorithmic approaches: Many competing techniques exist, some of them heavily parametrized. Again, not much knowledge exists, how, given a certain application, one makes the correct choice here. These general problems are called algorithm selection and algorithm configuration. Instead of relying on tedious, manual trial-and-error, one should rather employ available computational power in a methodical fashion to obtain an appropriate algorithmic choice, while supporting this process with machine-learning techniques to discover and exploit as much of the search space structure as possible.

In this cumulative dissertation I summarize nine papers that deal with the problem of model and algorithm selection in the areas of machine learning and optimization. Issues in benchmarking, resampling, efficient model tuning, feature selection and automatic algorithm selection are addressed and solved using modern techniques. I apply these methods to tasks from engineering, music data analysis and black-box optimization.

The dissertation concludes by summarizing my published R packages for such tasks and specifically discusses two packages for parallelization on high performance computing clusters and parallel statistical experiments.

Contents

1 Introduction

"It is better to have an approximate answer to the right question than an exact answer to the wrong one."

– John Tukey

Modern data-driven statistical techniques play an increasingly important part in applied data analysis and quantitative research. In this work I will focus mainly on classification and regression techniques which try to answer the following question: Given a set of observations $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, the training data, and a class of potential models $\mathcal{F}$, can we inductively find a model $\hat{f} \in \mathcal{F}$, which predicts the true output y by $\hat{y} = \hat{f}(\boldsymbol{x})$ for unseen $(\boldsymbol{x}, y)$ with a low error on average? Here, $\boldsymbol{x}$ will be from $\mathcal{X}$, the feature space, which may or may not coincide with $\mathbb{R}^p$ and $y \in \mathcal{Y}$, which is a finite set for classification and $\mathbb{R}$ for regression. We will usually assume that the elements of D are drawn i.i.d. from a joint, unknown distribution $\mathcal{P}$ on $\mathcal{X} \times \mathcal{Y}$, and in order to obtain $\hat{f}$ we must fit f to the training data in a statistical sense. A plethora of different models have been proposed during the last decades to handle these types of problems and many of them can be written in the form of regularized risk minimization:

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) + \lambda \Omega(f) \ . \tag{1}$$

Here, $\mathcal{F}$ is the space we are allowed to select our model from, $L(\cdot, \cdot)$ is a loss function that measures the distance between a true label (or value) y and its prediction $f(\boldsymbol{x})$, and $\Omega(\cdot)$ is an operator on $\mathcal{F}$ that measures the complexity of f, called a regularizer or complexity penalty. The trade-off between loss and regularization is controlled by λ. If f is specified by a vector of "natural" model parameters $\boldsymbol{\alpha}$, (1) becomes:

$$\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i, \boldsymbol{\alpha}), y_i) + \lambda \Omega(\boldsymbol{\alpha}) \ . \tag{2}$$

"Natural" means that we can directly estimate these parameters by minimizing (2), which distinguishes them from the so-called hyper-parameters discussed further below.
The reader will find excellent introductions to the whole topic in the books by Hastie et al. (2009) and Bishop (2006).
Much theoretical knowledge is available concerning the learning problem (2): Loss functions usually correspond to assumptions about the error distribution (e.g., quadratic loss usually implies the assumption of normally distributed errors), while regularizers corre-

spond to assumptions about the prior distribution on the parameters in a Bayesian sense. Non-differentiability often induces sparseness in the solution (e.g., see the hinge loss for support vector machines or the L_1-regularizer for feature selection), but we pay the price of having to solve a computationally and numerically more demanding problem. The kernelization of (2) introduces a further theoretical tool to arrive at non-linear modeling with essentially linear computational techniques and allows to handle spaces $\mathcal{X}$ which are structured in non-standard ways.

Considering all of the above, one might argue that to arrive at the optimal model and an associated solution strategy for a given task in statistical learning, one simply needs to consider the task's abstract properties and requirements. Up to a degree this is certainly true and can be considered a great success and an indication of the maturation of a still very young field. Unfortunately, the model building process is not as straightforward as it might sound: Many nuts and bolts still remain, and the practitioner usually has a larger set of alternative methods at his disposal, which are applied and tweaked in a lengthy trial-and-error fashion. The formal terminology for choosing among them is called model selection. Comparing such models – when only a finite amount of training data is available – is done by resampling, i.e., generating a sequence of training sets $D_1, \dots D_k$ and test sets $\tilde{D}_1, \dots \tilde{D}_k$ with

$$\frac{1}{k}\sum_{j=1}^{k}\frac{1}{|\tilde{D}_j|}\sum_{(\boldsymbol{x}_i, y_i)\in\tilde{D}_j}\rho\left(\hat{f}_{D_j}(\boldsymbol{x}_i), y_i\right) \approx E_{\mathcal{P}}\left(\rho\left(f(\boldsymbol{x}), y\right)\right) \tag{3}$$

Here, ρ is the true measure of performance as implied by the application (which not necessarily coincides with L). To simplify the notation it was assumed that we are interested in the estimated mean performance according to ρ (which must not always be the case). Now, what are the reasons that make model construction and selection less straightforward than the arguments above might have implied?

Sometimes the loss function of choice is not computationally tractable. The usual approach is to select a reasonable approximation available for L. Convex upper surrogate losses are a fine choice to approximate the zero-one loss, e.g., the hinge loss or the (less often used) Huber loss, while the second one can be numerically preferable because of its differentiability. If we want to estimate probabilities instead of predicting discrete class labels in classification, our loss function usually corresponds to the negative log-likelihood of the stochastic model we are fitting. At this point we have already begun to approximate the mathematical problem we were originally interested in, without perfect guarantees which approach will be the optimal choice – and this will not be the last time we are forced to proceed in such a manner. Consider, e.g., the following settings for a random forest: the number of trees, the size of the drawn subsets, whether to draw with replacement or not and the various settings of the tree itself (which splitting criterion, etc.). And the more we deviate from "standard statistical learning" scenarios (for which the defaults of these settings have often been optimized), the more advantageous it might be to correctly configure the algorithm. All these different modeling choices cause expressed and sometimes more subtle consequences regarding statistical, computational and practical properties of the model we are going to obtain. Usually it is quite hard to understand and quantify the trade-offs between these properties before actually running an excessive amount of experiments and trying everything out.

Moreover, often it is not $\boldsymbol{x}$ that enters our model f, but in reality we employ (sometimes extensive) pre-processing, so a more honest model equation might be $f(\tau(\boldsymbol{x}, \boldsymbol{\gamma}), \boldsymbol{\alpha}, \boldsymbol{\beta})$, where $\boldsymbol{\beta}$ denotes the hyper-parameters of the learning algorithm and τ denotes the pre-processing operation (which mostly comes with its own parameters $\boldsymbol{\gamma}$). Furthermore, feature selection must often be performed to obtain good results or to better interpret the model.

In my presented publications I follow a holistic, data-dependent philosophy, where every modeling choice is dictated by simply measuring the relevant performance metric through an appropriate resampling scheme for the available data. The modeling choices are then efficiently optimized by using an appropriate technique, depending on the structure of the decision space and the available budget, see Fig. 1.

Looking at the target function in equation (2), many research questions naturally arise:

1. Given a number of candidate models, how can we compare these in a valid statistical fashion? This seems to be an obvious question and we require a rigorous answer. Being able to quantitatively compare proposed solutions for a given task is at the core of scientific research.
2. How are we going to pre-process our data to arrive at optimal results?
3. How should we decide upon the model class and set parameters which cannot be determined by the usual model fit, e.g., maximum likelihood estimation or regularized risk minimization? These problems are called model selection and hyper-parameter tuning. In computer science these are better known under the terms algorithm selection and algorithm configuration, often applied to expensive optimization or decision procedures, although not limited to such methods.
4. Which features should be included in our model? This is the well-known feature selection problem in statistical learning.
5. How can we deal with the computational costs when performing such experiments?

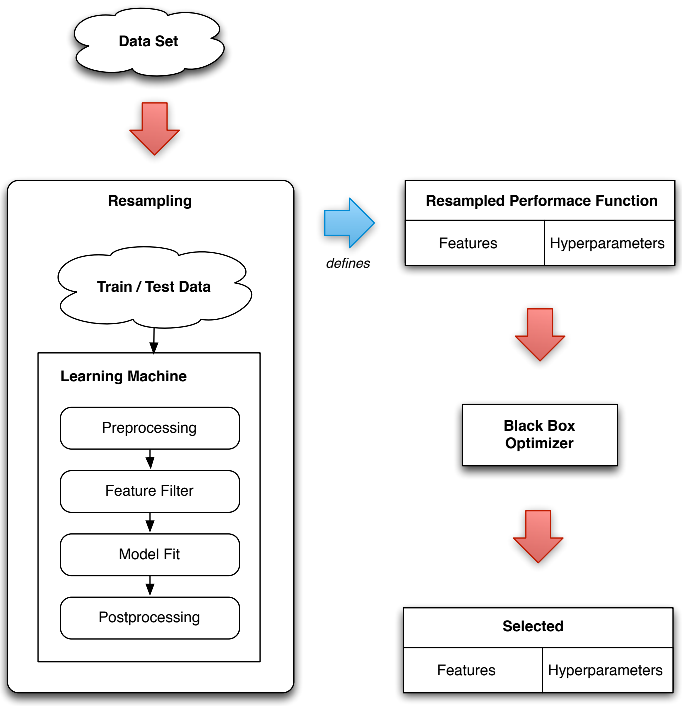

Figure 1: Schematic chain of modeling operations, jointly optimized.

6. How can we succinctly define and rapidly perform complex experiments to investigate the questions above?

As the last two questions already indicate, investigating these issues comes at a substantial computational cost. We would like to compare different models on different data sets, we will have to search over hyper-parameter and feature set spaces and we need to resample the data in order to assess the performance. This often results in thousands of model fits, which themselves might not be computationally cheap – depending on the size of the data and the scaling of the chosen methods. Of course, one obvious advantage is that the described scenario and many similar tasks in computational statistics are what is called "embarrassingly parallel". This means they consist of independent subtasks which can be solved in parallel on multiple cores and machines to drastically speed up the computation.

Statistical learning and especially classification has been successfully employed in the areas of bioinformatics, finance, audio / image recognition and many more. In this work I will focus on two areas of active current research. I will demonstrate that statistical learning techniques and model selection can be applied in a meaningful and efficient way to: a) music data analysis, which is a natural domain for feature selection as often hundreds of features are available through signal processing toolboxes and users are often interested in interpreting which variable sets work especially well for their considered tasks; b) the algorithm selection problem in optimization. I consider specifically the continuous domain for black-box function optimization and the traveling salesman problem as an example with a discrete decision space.

The subsequent sections are structured in the following way: In Sec. 2 three papers are presented that deal with the benchmarking and tuning of machine learning methods. The first article is a methodological overview of resampling techniques with specific considerations for regression models used in optimization. The second paper addresses the statistical evaluation of the recent class of local classification models by employing a bias-variance type of analysis on the benchmarking results so that significant insights regarding the advantage of these models can be gained. The third paper demonstrates how parameter-heavy models – in this case resulting from the chaining of pre-preprocessing steps with kernel models – can be efficiently optimized with a model-based approach.

Sec. 3 discusses feature selection in the domain of music classification. The first contribution compares feature group selection to a group lasso approach in order to investigate which groups of features are most helpful for an instrument recognition task. The second paper introduces a hybrid (1+1) evolutionary algorithm to select small feature sets for music genre recognition.

Using predictive modeling to improve research in optimization scenarios is the topic of Sec. 4. The first paper introduces a set of quantitative low-level features for the characterization of objective function landscapes in black-box optimization. The features are analyzed and selected by an evolutionary, cost-sensitive, multi-objective algorithm. The next paper deals with discrete optimization instead of continuous optimization, namely the traveling salesman problem. An evolutionary method is presented to artificially construct simple and difficult problem instances for the 2-opt algorithm. The third and last paper the extends the first one by constructing a cost-sensitive algorithm selection model to predict an optimal black-box optimizer given the proposed low-level function features.

Sec. 5 presents the packages **BatchJobs** and **BatchExperiments** for massive parallelization of statistical experiments on a cluster. The packages can be used to schedule R jobs to different high performance computing back-ends. While the former offers a flexible interface for generic jobs based on Map, Reduce and Filter operations, the latter provides a succinct definition language for embarrassingly parallel statistical experiments.

Finally, Sec. 6 summarizes all R packages that have been created and published alongside this dissertation.

2 Benchmarking and Tuning Machine Learning Methods

"Essentially, all models are wrong, but some are useful."

– George E. P. Box

2.1 Contributed Material

Bischl, B., Mersmann, O., Trautmann, H., and Weihs, C. (2012c). Resampling methods for meta-model validation with recommendations for evolutionary computation. *Evolutionary Computation*, 20(2):249–275

Schiffner, J., Bischl, B., and Weihs, C. (2012). Bias-variance analysis of local classification methods. In Gaul, W., Geyer-Schulz, A., Schmidt-Thieme, L., and Kunze, J., editors, *Challenges at the Interface of Data Analysis, Computer Science, and Optimization*, volume 43 of *Studies in Classification, Data Analysis, and Knowledge Organization*, pages 49–57. Springer

Koch, P., Bischl, B., Flasch, O., Bartz-Beielstein, T., Weihs, C., and Konen, W. (2012). Tuning and evolution of support vector kernels. *Evolutionary Intelligence*, pages 1–18

2.2 Resampling Methods for Meta-Model Validation with Recommendations for Evolutionary Computation

Bischl et al. (2012c) give an encompassing overview of the theory behind resampling strategies for model evaluation and many practical guidelines to select the correct strategy for a task at hand. Resampling data is an important topic in applied model evaluation when only a finite amount of data is available, which has to be efficiently used for training, validation and testing purposes simultaneously. In general, it refers to the generation of training and test sets from the original population by means of sampling and, subsequently, the estimation of model performance, see Fig. 2.

While most of the discussed methods can also be employed for normal regression and classification, the focus is on stressing the importance of evaluating model accuracy of regression techniques used as surrogate fitness models found in evolutionary computation and model-based optimization, see, e.g., Jones et al. (1998) or Paenke et al. (2006). In this case, data is often scarce when target function evaluations are expensive and the subject of proper evaluation during optimization has been somewhat neglected in most available publications, but see Tenne and Armfield (2008) for one of the notable exceptions.

Our paper describes in-depth all state-of-the-art resampling strategies, including bootstrapping, subsampling, cross-validation and various extensions like stratification and

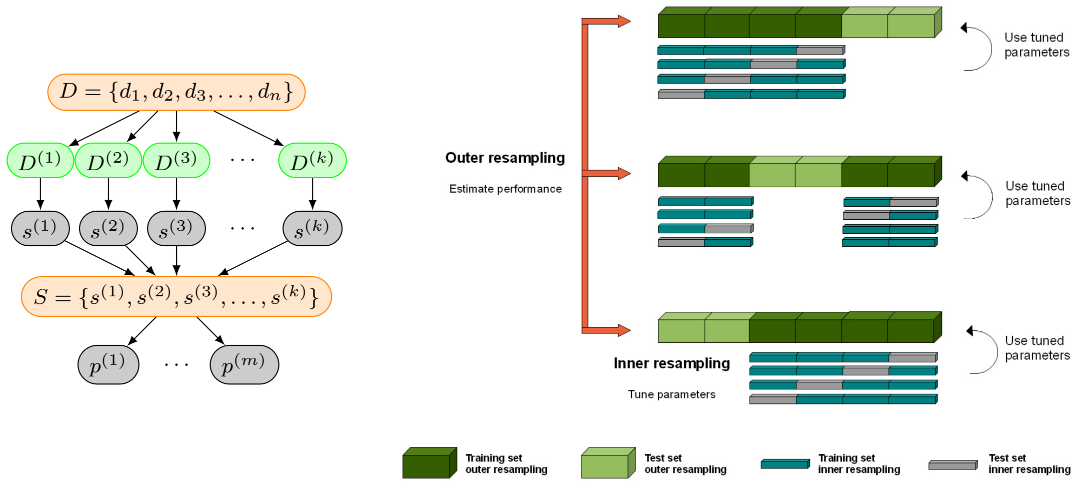
$D = \{d_1, d_2, d_3, \dots, d_n\}$
$D^{(1)}$ $D^{(2)}$ $D^{(3)}$ $\cdots$ $D^{(k)}$
$s^{(1)}$ $s^{(2)}$ $s^{(3)}$ $\cdots$ $s^{(k)}$
$S = \{s^{(1)}, s^{(2)}, s^{(3)}, \dots, s^{(k)}\}$
$p^{(1)}$ $\cdots$ $p^{(m)}$
Outer resampling
Estimate performance
Use tuned parameters
Use tuned parameters
Use tuned parameters
Inner resampling
Tune parameters
Training set outer resampling
Test set outer resampling
Training set inner resampling
Test set inner resampling

Figure 2: Left: Generic resampling scheme with training sets $D^{(i)}$, aggregated loss function values $s^{(i)}$ on test sets and aggregated performance measures $p^{(j)}$. Right: Nested resampling scheme, used for parameter tuning or feature selection. Both figures taken from Bischl et al. (2012c).

nested resampling necessary for tuning or feature selection (see Fig. 2). For all of these, we discuss their statistical properties, advantages and shortcomings. Common pitfalls and their ramifications are emphasized and the reader is provided with practical guidelines on how to proceed in his own experiments.
After the statistical preliminaries are covered, a focused discussion on model evaluation in evolutionary optimization follows, especially regarding the various trade-offs between model accuracy, usefulness for optimization and model interpretability. The article closes with two examples where proper model evaluation is illustrated in the context of model-based optimization.

2.3 Bias-Variance Analysis of Local Classification Methods

Local classifiers (Bottou and Vapnik, 1992; Tutz and Binder, 2005) are motivated by the assumption that for many data situations it is difficult to construct a global model for the whole population of observations and one should rather focus on modeling locally concentrated subsets of points individually. Many local approaches have already been published, and one of the best known methods is the k-nearest neighbor classifier. But localized versions of nearly every other model exist as well, e.g., for discriminant analysis variants, logistic regression, neural networks, support vector machines or boosting. One

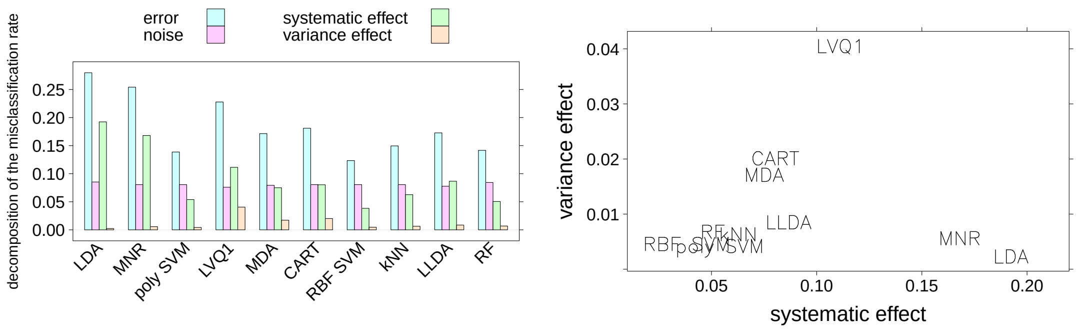

Figure 3: Left: Error rates, systematic and variance effects and data set noise rate averaged over the considered data sets. Right: Variance effect versus systematic effect. Both taken from Schiffner et al. (2012), left figure slightly adapted. Global classifiers are: linear discriminant analysis (LDA), multinomial regression (MNR), support vector machine with polynomial kernel (poly SVM). Local classifiers are: learning vector quantization (LVQ1), mixture discriminant analysis (MDA), classification tree (CART), support vector machine with radial basis function kernel (RBF SVM), k-nearest neighbors (kNN), localized linear discriminant analysis (LLDA), random forest (RF).

of the main contributions of our article (Schiffner et al., 2012) is that, after a brief theoretical categorization of the different local classification approaches, we analyze why local and hence more flexible models might exhibit a lower error in applications. This is demonstrated through the theoretical tool of a bias-variance decomposition of the prediction error: It is a well-known fact that under quadratic loss the expected error of a predictor can be decomposed into its bias – the systematic deviation from the Bayes predictor – and its variance – the random fluctuation around its mean prediction. We follow the work of James (2003) and perform a bias-variance decomposition under the usual zero-one loss for classification. The effects of bias and variance on local classification models are then studied through a benchmark study consisting of 26 publicly available data sets and 10 classifiers.

Concerning the bias of local models our intuition is strongly confirmed. The more flexible local methods exhibit a significantly lower bias than their global counterparts. Interestingly, we pay for this reduced systematic effect in error only with a slightly increased variance effect for most local methods, resulting in a lower error in total. Fig. 3 shows the main aggregated results for all considered classifiers, including total error, systematic and variance effects and their correlation.

2.4 Tuning and Evolution of Support Vector Kernels

While kernel-based methods like Support Vector Machines (SVMs) are very popular prediction models because of their high accuracy and flexibility introduced through the kernel trick, they also possess at least one major weakness: Their performance is highly sensitive to the correct choice of the kernel function as well as its parameters (in addition to the general complexity parameter, usually denoted by C).

In our paper (Koch et al., 2012) a model-based approach is proposed to not only tune the hyper-parameters of an SVM, but also the associated pre-processing operations. Such operations are especially important if complex modeling tasks from data mining or engineering have to be solved. We consider two of these: a) a time-series forecasting problem from water resource management, i.e., the prediction of stormwater fill levels based on past rainfall to prevent the flooding of sewage systems; b) the imbalanced classification problem of predicting acid concentrations from fluid spectroscopy measurements. While in the former case specific parameter dependent pre-processing of the time-series data must be performed to obtain meaningful features, in the latter task the high number of strongly correlated features is decorrelated by a principal component analysis. The principal components are used as new features and reduced by a feature selection filter and classification thresholds of the final model are adapted to account for the high class imbalance. This leads to tuning problems of the dimensions 9 (two embedding dimensions for two time series, four parameters for time-series pre-processing and C, γ, ϵ for support vector regression with the Gaussian kernel) and 13 (percentage of filters retained after filtering, five reweighting parameters for the five classes, five probability-thresholds for the five classes and C, γ for an SVM with the Gaussian kernel), respectively. For both of these tasks a sequential model-based optimization (SMBO) approach is used which solves both problems in only a few hundred fitness evaluations. This technique, outlined in Alg. 1, relies on the main idea of iteratively substituting the true fitness function (here: resampled model prediction performance) with a surrogate model which can be optimized much faster. In many cases a Gaussian process model (also called a kriging model) is used in such scenarios (our work being no exception), as this has proven to be most effective in the scenario of expensive global black-box optimization: a) It is a nonlinear regression (or alternatively: interpolation) technique that can model difficult functional landscapes with a low amount of data. b) It is a fully stochastic model which allows the assessment of local model uncertainty and therefore the definition of the so-called expected improvement (EI) criterion (Jones et al., 1998) to determine promising new points for further evaluation. c) Its infill criterion to decide the next point for evaluation can be efficiently optimized as analytical gradients are available for the EI, at least for

Algorithm 1: Sequential Model-Based Optimization

```
1 Let $g$ be the black-box function that should be optimized;
2 Generate initial design $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$;
3 Evaluate $g$ (possibly replicated) on design $y_i = g(\boldsymbol{x}_i)$;
4 Let D=$\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)\}$;
5 while budget not exhausted do
6     Build surrogate model $\hat{g}$ based on $D$;
7     Select next promising point $\boldsymbol{x}^*$ by optimizing in-fill criterion for $\hat{g}(\boldsymbol{x})$, e.g., $EI(\boldsymbol{x})$;
8     Evaluate new point (possibly replicated) $y^* = g(\boldsymbol{x}^*)$;
9     Extend design $D \leftarrow D \cup \{(\boldsymbol{x}^*, y^*)\}$;
```

the standard covariance kernels.
The paper also contains a second part where the automatic construction of kernels is attempted by genetic programming (Koza, 1992). This line of research is motivated by published articles that claim to have shown that this approach works in very general scenarios. We were eager to improve upon these results by enhancing the genetic algorithm as well as the SVM parameter optimization during its execution. But it turns out that on the considered benchmark problems even our improved genetic programing approach leads to disappointing results. Although good-performing common kernel functions can be recovered by the genetic search, which is surely interesting in its own right, in no case a significantly better result can be obtained by this computationally much more demanding technique. We therefore discuss the various reasons why the papers we compared our results to obtained a qualitatively different conclusion. These are two-fold, one is the often suboptimal tuning of the default SVM when compared to their respective GP approach, the other is the statistical over-interpretation of narrow confidence intervals obtained from performance evaluations during cross-validation. A further general problem is the often hard reproducibility due to missing details of the experimental setup and unpublished code.

2.5 Outlook

There are at least three different directions to extend the research discussed in this section: a) Usually no single learning algorithm dominates all other candidates in a realistic benchmarking experiment and it is also not possible to identify meaningful subregions of the data set space where this is the case by naive exploratory analysis. Meta learning (in its basic form) tries to rectify this by relating statistical properties of the data sets (so-called meta-features) to model performance. This relation is obtained by solving either one classification problem or as many regression problems as there are candidates.

A large amount of publications is available on this topic (see Vanschoren, 2010, for a recent overview), although no exceptional breakthrough seems to have been reached in this area. Personally, I think that we will never succeed in directly tackling the meta learning problem through simple statistics and prediction as too many (sometimes subtle algorithmic) factors are at work and optimization will always play a part. But using models learned from experience might provide for a very valuable prior distribution to start the optimization, see the related discussion on instance parameters in the outlook of Sec. 4. b) The tuning / configuration problem for machine learning is difficult to solve in a general and efficient way because its decision space is mixed continuous / categorical, the objective function is black-box, will in most cases be stochastic and can be arbitrarily expensive to evaluate. Also, especially if we want to configure over the space of multiple models at once, dependent (hierarchical) parameters will occur (e.g., consider the choice of a kernel function and its kernel parameters). Quite a lot of these issues are already addressed by the works of Hutter et al. (2011) and Thornton et al. (2012) which replace the Gaussian process by a random forest to perform the surrogate modeling. But one possible way of enhancing the search process is not addressed: In many cases we can identify promising parameter settings and model classes by considering only a smaller portion of data during training / resampling. Of course it is hard to apriorily decide the size of this portion and it is likely that dynamically increasing it in the later stages of the tuning is beneficial. As the time-complexity of many learning algorithms scales at least quadratically with the amount of training examples this algorithmic "knob" offers a major potential for gaining efficiency, but one pays for this with a possibly distorted and noisier objective function.

3 Feature Selection in Music Data Analysis

"An unsophisticated forecaster uses statistics as a drunken man uses lamp-posts - for support rather than for illumination."

– Andrew Lang

3.1 Contributed Material

Bischl, B., Eichhoff, M., and Weihs, C. (2010a). Selecting groups of audio features by statistical tests and the group lasso. In *9. ITG Fachtagung Sprachkommunikation*. VDE Verlag

Bischl, B., Vatolkin, I., and Preuss, M. (2010b). Selecting small audio feature sets in music classification by means of asymmetric mutation. In *PPSN XI: Proceedings of the 11th International Conference on Parallel Problem Solving from Nature*, volume 6238 of *Lecture Notes in Computer Science*. Springer

3.2 Feature Selection

Feature selection (Guyon and Elisseeff, 2003) is an important topic in applied statistical learning. It is imperative to note that there are different reasons for not using the full feature set: a) First, it is a well-known fact that the predictive power of classification and regression models can deteriorate if a high number of correlated and noisy features is present and naively included into the model without prior selection. We might therefore cast this problem directly into an optimization problem where we operate on the space of binary characteristic vectors from $\{0, 1\}^p$, but we have to be aware of the fact that this is provably NP-hard (Amaldi and Kann, 1998). Often we have to avoid enumerating all candidate solutions and therefore one has to employ computationally cheaper heuristics. It should also be noted that, even if given infinite computational power, searching the whole binary space might be disadvantageous from a modeling perspective. The more candidate sets we visit, the more we have to deal with the fact that we discover solutions which look promising on the data used for evaluation, but which are suboptimal w. r. t. expected loss. This problem rears its ugly head in the same fashion in tuning applications (or in any scenario where extremely many candidate models are compared on the same data and optimized via a lengthy computational process). It is not exactly the same as overfitting and could even be orthogonal to it. A more appropriate term might be "oversearching" (Quinlan and Cameron-Jones, 1995) and it is closely related to the

problem of multiple testing. How to deal with it in an optimal fashion is currently not known and an area of further research.
b) Reducing the feature set results in a simpler model and we are often willing to trade-off a certain proportion of predictive power for this advantage. One of the most common reasons for such a reduction is that we want to interpret the model, others are reduction of feature computation time or storage space in future applications. Also obtaining the features for future observations could come at a cost, be it in a computational, monetary or ethical sense.
For both settings a multitude of different selection methods is currently available. a) Filters: They operate independently from the subsequently applied learner and rank features by assigning them a numerical value according to their "relevance". These range from simple univariate measures of separability, correlation or mutual information to more complex multivariate criteria that are defined on all feature subsets (e.g., see minimum-redundancy-maximum-relevancy by Peng et al., 2005, as a popular choice). They are usually fast but do not consider the interaction between the selected feature set and the applied learning algorithm. b) Wrappers: They use the predictive power of the used learner – that they access as a black-box – to evaluate feature sets (Kohavi and John, 1997). Here, one either moves locally through the search space (forward and backward search) or genetic algorithm variants are used. They are computationally expensive but can be combined with any learning algorithm and performance measure of choice. c) Embedded methods: They directly integrate the feature selection into the model fitting process. Examples are the well-known recursive feature elimination scheme for SVMs by Guyon et al. (2002) and regularization approaches via the L_1 norm (coined the "lasso" by Tibshirani, 1994) or L_0 norm[1] (Weston et al., 2003). The former one is computationally simpler to handle, but still leads to nontrivial optimization problems. The lasso corresponds to an L_1 constraint on the model coefficients (in a linear model formulation) and can also be interpreted from a Bayesian perspective as a Laplacian prior distribution for these coefficients. It assumes a sparse model structure, where many model coefficients are supposedly zero, and effectively recovers this sparse structure through the specific geometry of the penalty constraint.

[1]The L_0 "norm" is not a vector norm in the strict sense.

3.3 Selecting Groups of Audio Features by Statistical Tests and the Group Lasso

In Bischl et al. (2010a) the classification task of discriminating between musical instruments is approached from the perspective of feature selection. A few hundred audio features are available for this endeavor, and these are grouped into sets, according to their methodological origin. We consider the absolute amplitude envelope, mel-frequency cepstral coefficients (MFCCs), the pitchless chromagram and linear predictor coefficients (LPCs). Researchers working in this area are naturally interested in the question which sets (or which combinations) work best for modeling the task under consideration.

We offer two solutions to this selection problem: a) We reduce it to a statistical testing problem between models (of different feature group sets) by applying the framework of Hothorn et al. (2005) and therefore continue the work begun in Szepannek et al. (2009). This approach can be combined with any learning algorithm. b) We offer a much faster (but less model-independent) approach based on logistic regression and the group lasso (Meier et al., 2008). For the latter, we minimize the regularized risk

$$\min_{\alpha,\alpha_0}\left(\sum_{i=1}^{n} L(y_i, \alpha^T x_i + \alpha_0) + \lambda \sum_{g=1}^{G} ||\alpha_{I_g}||_2\right) ,$$

where L is the binomial loss acting on the linear model and I_g is an indicator function referring to the index set of all features in group g. The feature groups I_g usually partition all available features, so $\alpha^T = (\alpha^T_{I_1}, \ldots, \alpha^T_{I_G})$ (if one allows a slight abuse of notation and some reordering of the features). As one clearly sees, disjoint groups of features are penalized internally by the L_2 norm, but an L_1 norm acts as a regularizer on the groups itself, provoking a sparse selection of these and achieving feature selection at the group level.

We conclude that the computationally cheaper method already produces relevant orderings of the feature subgroups, but the statistical testing allows for more detailed results and more flexibility in cases where the logistic regression model is a misspecification.

3.4 Selecting Small Audio Feature Sets in Music Classification by Means of Asymmetric Mutation

In Bischl et al. (2010b) the problem of automatically classifying the musical genres of songs is considered, which is helpful in indexing large music databases or learning personal preferences of listeners (Blume et al., 2011). Again, we are interested in reducing the

hundreds of potential audio signal features to a manageable number, but we are not selecting on a group level anymore. In Bischl et al. (2010b) a specific evolutionary strategy (ES) is proposed which combines a simple (1+1) mutation strategy (with an asymmetric mutation rate to favor small feature sets) and a correlation-based heuristic to ensure a promising generation of candidate feature sets during the search.
This algorithm is validated on a database of 120 commercial albums whose songs are labeled as Classic, Pop/Rock, Rap, Electronic, R&B, ClubDance and Heavy Metal by experts from the AllMusicGuide[2]. The largest available feature set consists of 572 elements. Our proposed algorithm (abbreviated ESGH in Fig. 4) is compared to classifying without feature selection, random search, a greedy forward search (GFS) using the same correlation-based heuristic as ESGH to rank features (also called a rank search) and different (1+1) ES variants. Fig. 4 shows exemplary results for three genres.

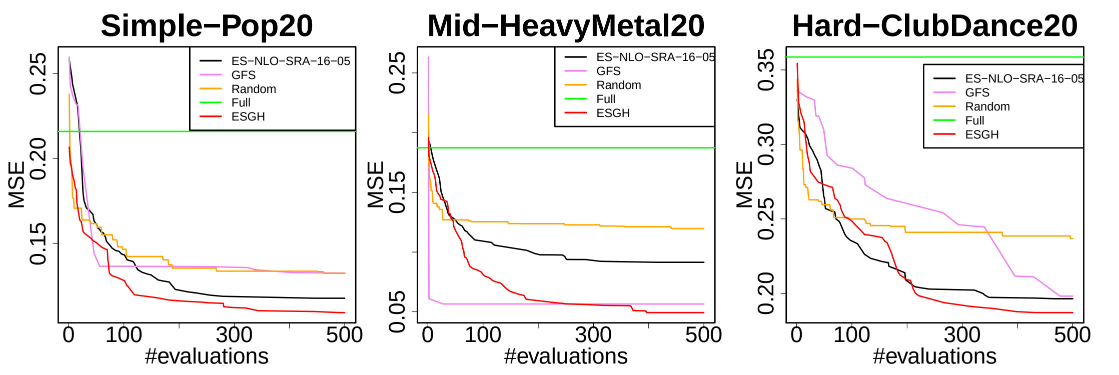

Figure 4: Comparison of best ES variant (ES-NLO-SRA-16-05, for details see Bischl et al., 2010b) with baseline methods and hybrid ES (ESGH) on three genre prediction tasks, taken from Bischl et al. (2010b). The baseline methods are greedy forward search (GFS), random search (Random) and the full model without feature selection (Full). Displayed is the number of optimizer iterations vs. the MSE between the predicted and true proportion of genre segments per song (a song is segmented into a number of shorter parts which are in turn classified). All results are averaged over five statistical runs.

We can conclude that our asymmetric, hybrid strategy reliably selects sparse solutions on the considered genre tasks with a relatively low number of iterations. It combines the best properties of the (already very well-performing, but sometimes suboptimally terminating) rank search and the general (1+1) ES (which converges much slower as no search space guidance is provided in the algorithm for generating candidate points).

[2]`www.allmusic.com`

3.5 Outlook

Much work still needs to be done regarding this topic. First, one can rightfully treat the feature selection as a multi-criteria problem (Vatolkin, 2013). In most cases we cannot explicitly write down trade-off costs between the loss in prediction and the size of the feature set, and even if we can, such information is rarely included in the modeling process. Another very promising aspect seems to be to select features across multiple data sets. The reason is simple and also obvious: Quite often we are not interested in the selection of an optimal feature set for one data set, but for a well-defined task domain. Then it is not sufficient to perform the selection separately for each data set, as multiple correlated and redundant solutions can exist, and one is interested in a single feature set which consistently performs well across the whole domain. Note that in Mersmann et al. (2011) (see Sec. 4) we already introduce a feature selection method which addresses both aspects. Finally, although general search heuristics like ESs often perform well when given enough fitness evaluations they "ignore" quite a lot of easily accessible information which could and should be exploited during the search: Cheap filter rankings are always computable and could at least be used to create a skewed initial, prior distribution over the feature space. And most learning models allow the computation of a variable importance measure after they have been fitted, which could also be fed back to the ES.

4 Statistical Modeling for Optimization Problems and Algorithm Selection Techniques

"It is unworthy of excellent men to lose hours like slaves in the labour of calculation which could safely be relegated to anyone else if machines were used."
– Gottfried W. Leibniz

4.1 Contributed Material

Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., and Rudolph, G. (2011). Exploratory landscape analysis. In Krasnogor, N., editor, *Proceedings of the 13th annual conference on genetic and evolutionary computation (GECCO '11)*, pages 829–836. ACM
Mersmann, O., Bischl, B., Trautmann, H., Wagner, M., Bossek, J., and Neumann, F. (2013). A novel feature-based approach to characterize algorithm performance for the traveling salesperson problem. *Annals of Mathematics and Artificial Intelligence*, pages 1–32
Bischl, B., Mersmann, O., Trautmann, H., and Preuss, M. (2012b). Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In *Genetic and Evolutionary Computation Conference (GECCO)*

4.2 Exploratory Landscape Analysis

In order to better relate the outcome of optimization benchmarks to properties of the considered test functions, we suggest a new technique coined Exploratory Landscape Analysis (ELA) in Mersmann et al. (2011). Although one might first think of high-level function properties designed by experts for this purpose, we instead construct relatively cheap low-level computer generated features to express the high-level ones, see Fig. 5. These can be computed automatically for any new, unknown objective function.
In our approach we extract these features from systematically sampled points in the decision space, to empirically describe aspects like convexity, multi-modality and curvature. Their implementation ranges from calculating simple statistics from objective value distributions over fitting surrogate models to the landscape to running short sequences of local optimization.
We demonstrate that these features can successfully be employed to predict the BBOB'09/10 (Hansen et al., 2009) function grouping with low classification error. We create a new multi-objective genetic algorithm to select the cheapest and most relevant features for this task. Our genetic algorithm is able to switch on complete feature groups

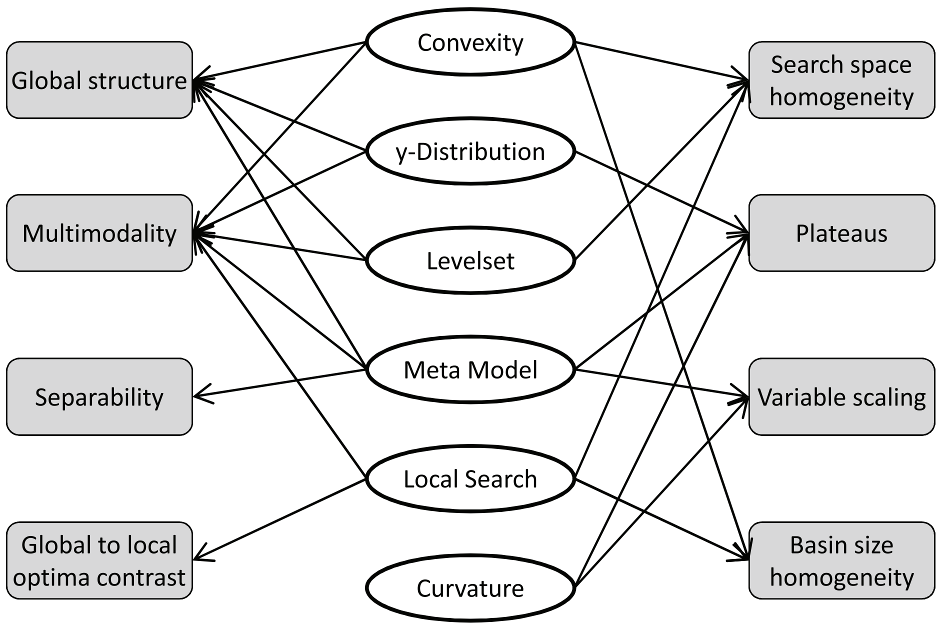

Figure 5: Relationships between high-level features (grey) and low-level feature classes (white), taken from Mersmann et al. (2011).

(the groups have been created according to feature type and computational costs) in its mutation and cross-over operators. Technically, we measure the misclassification rate, the computational cost per selected feature set and the model complexity (number of features). We then approximate the Pareto front for these three criteria with a binary variant of the S-metric selection evolutionary multi-objective algorithm (SMS-EMOA) (Emmerich et al., 2005) which optimizes the dominated hypervolume of its population. In a second experiment, we predict seven high-level function properties (manually assigned by experts) by constructing seven classification problems. We use our multi-criteria algorithm to simultaneously optimize the same features, but this time consider the maximum misclassification rate over all seven classification problems. We therefore construct a Pareto front of relevant features for multiple data sets.

4.3 A Novel Feature-Based Approach to Characterize Algorithm Performance for the Traveling Salesperson Problem

The traveling salesman problem (TSP) is one of the most well-known and well-studied NP-hard discrete optimization problems. Although a polynomial time approximation scheme exists (Arora, 1998), heuristical approaches are often used due to their often good performance and much simpler implementation. In Mersmann et al. (2012) we investigate the common 2-opt local search algorithm which modifies only one pair of edges per iteration (Johnson and McGeoch, 1997). Despite the success of this algorithm, it is still hard to understand the performance of this algorithm from a theoretical point of view, as experimentally obtained results are often much better than worst-case guarantees obtained from theoretical considerations.

We measure the difficulty of a problem instance for a given heuristic by calculating the approximation ratio, i.e., the multiplicative loss when running the heuristic in relation to the optimal solution, which is feasible to compute through a branch-and-cut approach[3] on small to medium sized TSP instances. The approximation rate is a common measure of performance for TSP approximation schemes. As before in Mersmann et al. (2011), we are interested in understanding which properties of problem instances mainly influence problem difficulty and algorithmic performance.

A further obstacle that has to be resolved for our undertaking is the generation of a representative instance set as a basis for the analysis. A generic evolutionary approach is presented which allows the construction of TSP instances in the Euclidean plane which are simple or hard to solve for the algorithm under consideration. Fig. 6 shows two evolved examples.

To analyze our obtained performance results, we again use a statistical learning approach by defining a set of computationally cheap instance features to characterize TSP instances. From the problem instance features classification rules are derived which predict the hardness level of an instance. It is possible to predict the correct instance classes with only marginal errors and our results are supported by an exploratory analysis of the evolved instances and the respective optimal tours. Instances of moderate difficulty can now be constructed by a newly introduced "morphing" of easy TSP instances into hard ones (i.e., a convex-combination of node coordinates). Systematic changes of the feature levels along the path are identified. A multivariate adaptive regression spline (MARS) model is successfully applied to predict the approximation quality of 2-opt, based on the features of an adequate subset of the generated instances with appropriately high accuracy.

[3]Concorde solver: `http://www.tsp.gatech.edu/concorde.html`

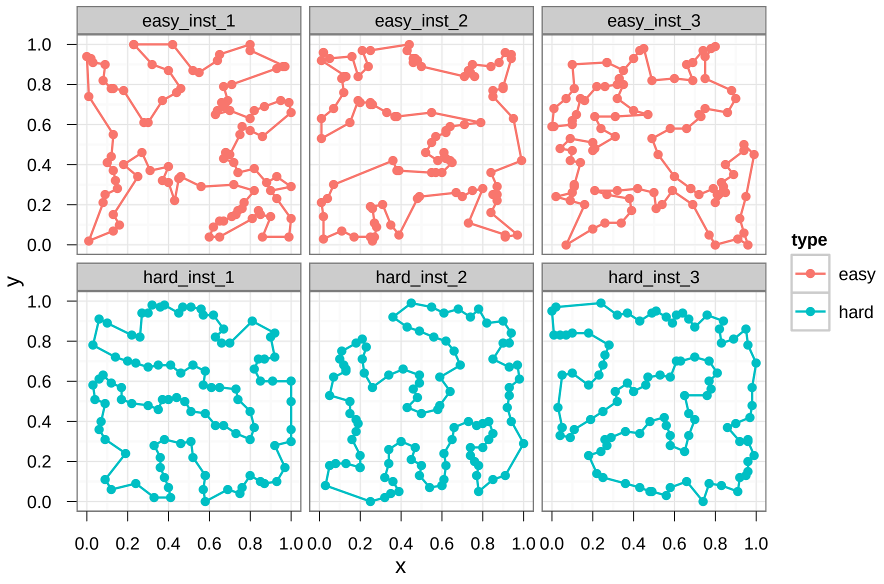

Figure 6: Examples of evolved simple and hard TSP instances, including their optimal tours, taken from Mersmann et al. (2012).

4.4 Algorithm Selection Based on Exploratory Landscape Analysis and Cost-Sensitive Learning

The algorithm selection problem (ASP) stems from the now very old desire to choose the right tool for a task without enumerating all available options. A cleverly constructed portfolio combined with an efficient selection mechanism should surely score better than any single algorithm. To make the algorithm selection approach work, we need the following components: a) a representative set of benchmark instances that we can run our algorithms and later train our prognostic model on; b) a carefully constructed, automatically computable set of instance features to characterize them; c) a proper statistical model which predicts a (hopefully well-performing) algorithm given the feature vector of a problem instance.

Our approach studies the algorithm selection problem in the context of black-box function optimization. It builds upon the already mentioned Exploratory Landscape Analysis features from Mersmann et al. (2011). From the benchmarking performance results, a diverse set of four optimizers is constructed which would lead to very good results if we had an oracle to always select the appropriate algorithm out of the restricted subset subset for a given problem.

	med(rel. ERT)	max(rel. ERT)	med(rel. ERT)	max(rel. ERT)
optimum	1.45	4.17	1.45	4.17
all features	1.54	86.76	1.90	58731.08
cheap features	1.61	58731.08	2.70	58731.09

Table 1: Cross-validated classification results with cost-sensitive support vector regression based on omitting function instances (left) and whole functions (right) in case all features or only the "cheap" features are used. Taken from Bischl et al. (2012b). The table shows expected run time (ERT) ratio of the best available algorithm and the selected one for a specific test function; displayed statistics are median and maximum values across test functions. Note that "best available algorithm" refers to the best one w. r. t. the whole BBOB data set, not only the best one from our reduced set of four algorithms. The optimal performance for this set (if we had an oracle to always select the correct algorithm from it) is displayed in the first line. The large numbers for the maximum value indicate that some methods pick a non-converging algorithm in some cases (for these non-converging algorithms an ERT imputation was done prior to the analysis).

Traditionally, the central modeling task is either cast into a normal classification problem ("given the problem features, predict the best candidate algorithm") or multiple regression problems ("given the problem features, predict each candidate's performance value, then select the algorithm with the best estimated performance value"). We argue that in a realistic setting – where prediction errors can never be avoided – neither approach is optimal. The reason for this is that the model takes the realistic loss (here difference in runtime to reach the optimum of the test function up to a desired accuracy) into account that occurs under a wrong selection. The loss obviously depends on the difference in performance of the optimal (available) selection and the selection obtained by estimation, which in turn implies some variation of cost-sensitive learning. The fact that the selection costs differ for each observation in the training set suggests the not very well-studied scenario of cost-sensitive learning with example specific costs. We apply the recently proposed method of one-sided support vector regression by Tu and Lin (2010) for this purpose.

Concerning our results regarding the ASP (see Tab. 1), we show that it works in two different scenarios: a) We can generalize to new instances of the same function for all functions in the BBOB test set and can predict the optimal or close to optimal portfolio candidate. b) To a surprisingly high degree we can also generalize to completely new test functions. But it also becomes clear that the set of BBOB functions might lie too "sparse" in feature space (in a sense they have been designed that way) to correctly evaluate our approach in a cross-validation scheme as we have too few similar examples to learn and generalize from for a specific test function.

4.5 Outlook

All of the covered papers present a fruitful combination of optimization methods on the one side and statistics / machine learning on the other. A few paths of further research seem especially promising and come directly to mind when considering the work of this section: a) We want a proper instance feature- and model-based combination of algorithm selection and configuration in one coherent method. To achieve this, a new expected improvement criterion must probably be constructed which takes the special relation of instance features and configuration settings into account. For one recent, quite flexible non-model-based approach see Kadioglu et al. (2010). For a model-based approach which does not take instance features of new test instances into account see Hutter et al. (2011). b) For black-box optimization one could gain more efficiency by measuring ELA features and algorithm performance during (multi-start) optimizer runs and by dynamically selecting the optimizer class and its configuration based on the observed data (online setting). c) Multi-objective black-box optimization can be considered as a further domain of application for ELA-based analysis, algorithm selection and configuration.

5 A Framework and Two R Packages for Statistical Experiments on High Performance Computing Clusters

"When in doubt use brute force."

– Ken Thompson

5.1 Contributed Material

Bischl, B., Lang, M., Mersmann, O., Rahnenführer, J., and Weihs, C. (2012a). Computing on high performance clusters with R: Packages BatchJobs and BatchExperiments. Technical report

The contributed paper has been submitted to the *Journal of Statistical Software* under the title *BatchJobs and BatchExperiments: Abstraction mechanisms for using R in batch environments* and is currently under review. The associated R packages **BatchJobs** and **BatchExperiments** and their documentations are publicly available on the Comprehensive R Archive Network (CRAN) and at their project page at `http://batchjobs.googlecode.com`.

5.2 Computing on High Performance Clusters with R: Packages BatchJobs and BatchExperiments

Extensive computer experiments are a common way to compare statistical methods and to gain insights into their behavior under different application scenarios as well as to study their respective advantages and disadvantages. Many universities and research facilities offer access to dedicated computing clusters and the fast development of cloud computing services like Amazon's EC2 platform will further improve the situation. Another relevant factor is the "embarrassingly parallel" nature often encountered in statistical computations. Therefore distributing the computationally intensive parts of such experiments to high performance clusters seems obvious. But it is not a trivial effort to harness these types of systems. They are typically managed by job schedulers, i.e., one cannot directly invoke processes on nodes, but has to define special job definition files including our required computational resources and then submit these via operating system commands. On top of this, these definition files and their associated commands are not standardized across batch systems, further complicating reuse of code, reproduction of results and collaborative work. If we want to perform statistical experiments in R, we need a

framework that contains a description language for all our computational steps. In Bischl et al. (2012a) we introduce two R packages for exactly this purpose. They enable the user to concentrate on the design of his statistical experiments and to efficiently use all available resources. Further design goals have been portability and reproducibility: Both packages are applicable in any batch computing environment and the back-end can be switched by a single line of code. Torque/Maui, Load Sharing Facility (LSF) and Sun Grid Engine implementations are already available, Slurm and Condor will follow shortly. Furthermore, for every computational part a seed for the random number generator is automatically stored to ensure the reproducibility of stochastic experiments.
The first package, **BatchJobs**, builds upon the expressive power of the functional programming concepts of *Map*, *Reduce* and *Filter*. *Mapping* refers to the concept of applying a unary function $f(\cdot)$ to a vector or list of elements

$$(x_1, \ldots, x_n) \xrightarrow{f} (f(x_1), \ldots, f(x_n)) \ .$$

If we envision f to be a computational operation that we would like to apply to different parameters or data sets, we can directly see how the parallelization of such a task becomes useful. *Reducing* (also often called folding or accumulating) means the successive application of a binary function $g(\cdot, \cdot)$ to a vector or list until a single result remains.

$$(x_1, \ldots, x_n) \to g(\ldots(g(g(x_1, x_2), x_3), \ldots), x_n)$$

This can be used to recombine the results from a mapping operation or to use a divide and conquer approach to parallelize certain problems. Finally, *filtering* is formally defined to be the selection of elements from a list w. r. t. a logical predicate. In our context it will allow the selection of subsets of experiments or their results to work on them individually.
One important aspect which distinguishes this toolbox from other approaches (Schmidberger et al., 2009) is that it provides a persistent state of computation for all jobs in a database. This allows the user to query the status of all jobs at any point in time, submit only subsets (which is important when the complete number of experiments is too large to schedule at once) and provides facilities for extensions. Maybe even more importantly, it enables the user to work in a completely asynchronous fashion. This has the significant benefits that a) parallel jobs can be defined in exactly the right computational size and the scheduler can start them individually when resources become available; b) extremely large job sets can be handled that would otherwise potentially overwhelm the batch system.
In order to further increase the package's impact, it also contains: debugging facilities,

as this is one of the most difficult and time-consuming aspects of parallel programming; functions to "chunk" quickly terminating conceptual jobs into one batch system job to reduce overhead; and the option to work on the results of previous operations in parallel.

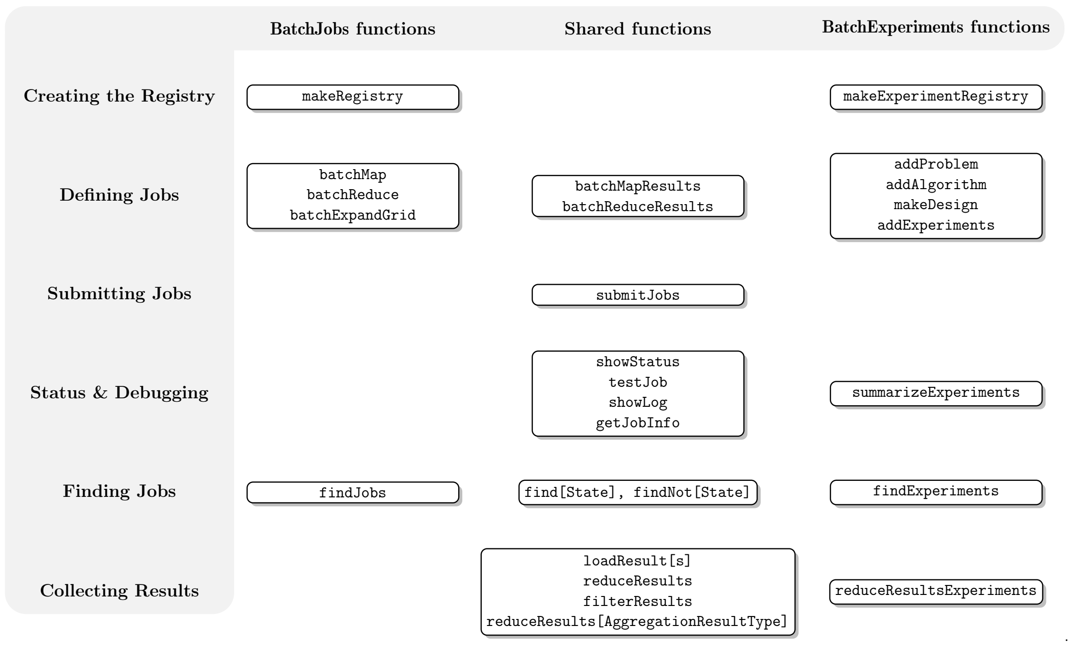

Figure 7: Workflow and most important functions, taken from Bischl et al. (2012a).

The second package, **BatchExperiments**, extends **BatchJobs** by making it directly applicable to statistical comparison experiments. The main insight is that an extremely large number of relevant practical studies can be cast into the form of "apply algorithm A on problem instance P and store some results". As both the problem generating function (e.g., imagine simulated data) and the algorithm can depend on parameter settings, it is possible to associate arbitrary statistical designs with both parts to study the influence of parameter variations on the results. Experiments are now defined as the cross-product of problems, algorithms and their respective parameter setting and they can be replicated any number of times. Such a replicated experiment constitutes one single job in the terminology of **BatchJobs**. The workflow for both packages and their most important common and individual functions are depicted in Fig. 7, while Fig. 8 shows the connection between problems, algorithms and parameters for **BatchExperiments**.

BatchExperiments contains further convenience mechanisms to simplify experimental work: a) Problems can have "static" as well as "dynamic" parts. For the former, imagine anything that does not change during an experiment, e.g., a data set, for the latter, think of anything that depends on parameters or is stochastically generated, e.g., a data simulation function. b) Stochastic problems have their own associated random number generator seed and can be "synchronized" across algorithm applications. This is useful as we generally want to reduce variance this way in stochastic comparison experiments. c) Problems, algorithms and experiments can be freely added to and removed from the experimental setup without repercussions. This is especially important as experiments often change during their execution when one learns due to the inspected first results, e.g., one wants to include further problems, algorithms or parameter settings in the study.

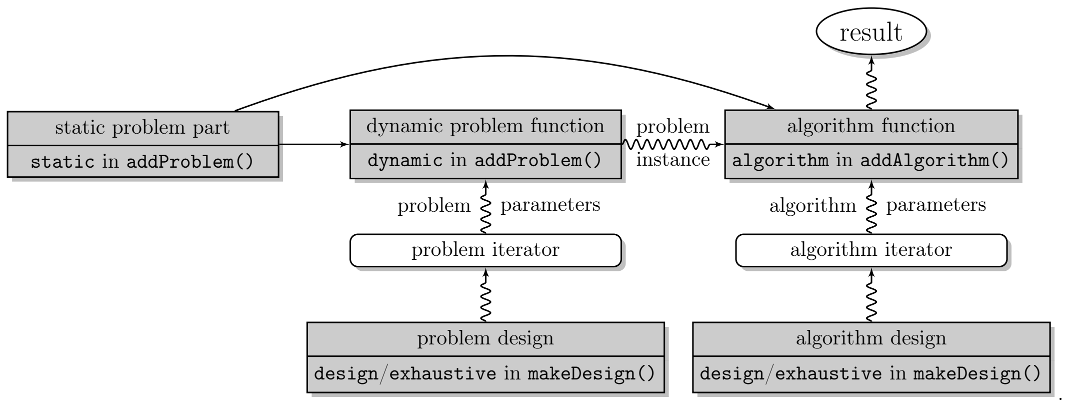

Figure 8: Relationship between **BatchExperiment** concepts of problems, algorithms and parameters, taken from Bischl et al. (2012a).

5.3 Outlook

While extensions for other batch systems are nearly trivial, we also plan the support of environments where no shared filesystem is available and therefore file-staging mechanisms for input and output data need to be provided. Furthermore, some experiments call for the definition of dependent jobs, where certain computational parts can only be executed when their prerequisites have already terminated. Such dependency structures can be defined in a graph, and a topological sorting of the nodes would then define a valid job order for scheduling.

BatchExperiments naturally allows extensive benchmark studies in the areas of machine learning, optimization and survival analysis. Furthermore, another interesting application might be the combination of **BatchJobs** with a model-based optimizer to solve extremely time-consuming problems in black-box optimization, algorithm configuration and general computer experiments.

6 Accompanying Software

"An algorithm must be seen to be believed, and the best way to learn what an algorithm is all about is to try it."

– Donald Knuth

Several R packages have been developed by me and my respective co-authors during the completion of this thesis and are contributed alongside. All packages are publicly available on CRAN, except for **mlrMBO** (see below), which is still under heavy development, but publicly available as an experimental version from its GitHub repository. Much emphasis has been put onto the usage of proper software engineering, design and quality control techniques. All packages contain extensive R documentation, examples and unit test sets. For **mlr** a web tutorial is available at its project home under `https://github.com/berndbischl/mlr`, while **BatchJobs** and **BatchExperiments** have an associated Google Code page, which contains documentation regarding their installation and configuration, as well as an FAQ.
Nearly all modeling and machine learning experiments in my contributed papers have been written within the **mlr** framework. Parallelization was nearly always performed with the help of **BatchJobs**. It should be strongly emphasized that the mentioned software and abstraction mechanisms proved to be an invaluable and very necessary ingredient in conducting all previously covered experiments.

General purpose packages
BBmisc: A generic helper package, mainly for good package design. Includes many convenience functions.
ParamHelpers: Includes infrastructure to describe general parameter objects for machine learning models and evolutionary algorithms as well as a generic way to archive parameters and fitness values during optimization.

Framework mlr - machine learning in R
mlr: Provides a generic interface to now over 50 classification and regression techniques, all standard resampling techniques and performance measures in supervised learning. It furthermore:

- allows to use different deterministic and evolutionary optimizers to tune hyperparameters of machine learning models.
- offers many standard filter and wrapper feature selection techniques. Some embedded L_1 regularization methods are already available as base learners in **mlr**

itself.

- allows the extension of basic learning procedures with feature selection, tuning and pre-processing operators. This makes convenient nested resampling and generic, efficient tuning of data mining operator chains possible.

mlrMBO: Offers a sequential model-based optimization routine that can use any regression learner from **mlr** as a surrogate fitness model. Can be used in **mlr** as an efficient optimizer for tuning.

Optimization
tspmeta: A toolkit: to generate simple and hard TSP instances by an evolutionary algorithm; to morph them to create instances of intermediate difficulty; to calculate characterizing instance features; and to run a battery of TSP solvers.

I have also tried to improve the package **soobench** a little bit, which contains a battery of deterministic and noisy benchmark functions for single-objective optimization, generic noise generators, visualization and convenience functions for optimization benchmarks. But as this is mainly the work of O. Mersmann, it is not submitted here.

Parallelization and large scale experiments
BatchJobs and **BatchExperiments**: Both packages have already been covered in Sec. 5.

References

Amaldi, E. and Kann, V. (1998). On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209(1-2):237–260.

Arora, S. (1998). Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782.

Bischl, B., Eichhoff, M., and Weihs, C. (2010a). Selecting groups of audio features by statistical tests and the group lasso. In *9. ITG Fachtagung Sprachkommunikation*. VDE Verlag.

Bischl, B., Lang, M., Mersmann, O., Rahnenführer, J., and Weihs, C. (2012a). Computing on high performance clusters with R: Packages BatchJobs and BatchExperiments. Technical report.

Bischl, B., Mersmann, O., Trautmann, H., and Preuss, M. (2012b). Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In *Genetic and Evolutionary Computation Conference (GECCO)*.

Bischl, B., Mersmann, O., Trautmann, H., and Weihs, C. (2012c). Resampling methods for meta-model validation with recommendations for evolutionary computation. *Evolutionary Computation*, 20(2):249–275.

Bischl, B., Vatolkin, I., and Preuss, M. (2010b). Selecting small audio feature sets in music classification by means of asymmetric mutation. In *PPSN XI: Proceedings of the 11th International Conference on Parallel Problem Solving from Nature*, volume 6238 of *Lecture Notes in Computer Science*. Springer.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer.

Blume, H., Bischl, B., Botteck, M., Igel, C., Martin, R., Roetter, G., Rudolph, G., Theimer, W., Vatolkin, I., and Weihs, C. (2011). Huge music archives on mobile devices. *Signal Processing Magazine, IEEE*, 28(4):24–39.

Bottou, L. and Vapnik, V. N. (1992). Local learning algorithms. *Neural Computation*, 4(6):888–900.

Emmerich, M., Beume, N., and Naujoks, B. (2005). An EMO algorithm using the hypervolume measure as selection criterion. In *Proceedings of the Third international conference on Evolutionary Multi-Criterion Optimization*, EMO'05, pages 62–76.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182.

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422.

Hansen, N., Auger, A., Finck, S., and Ros, R. (2009). Real-parameter black-box optimization benchmarking 2009: Experimental setup. Technical Report RR-6828, INRIA.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer.

Hothorn, T., Leisch, F., Zeileis, A., and Hornik, K. (2005). The design and analysis of benchmark experiments. *Journal of Computational and Graphical Statistics*, 14:675–699.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *Proceedings of the 5th international conference on Learning and Intelligent Optimization*, LION'05, pages 507–523. Springer.

James, G. M. (2003). Variance and bias for general loss functions. *Machine Learning*, 51(2):115–135.

Johnson, D. S. and McGeoch, L. A. (1997). The traveling salesman problem: A case study in local optimization. In Aarts, E. H. L. and Lenstra, J. K., editors, *Local Search in Combinatorial Optimization*. Wiley.

Jones, D., Schonlau, M., and Welch, W. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492.

Kadioglu, S., Malitsky, Y., Sellmann, M., and Tierney, K. (2010). ISAC – instance-specific algorithm configuration. In *Proceedings of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 751–756. IOS Press.

Koch, P., Bischl, B., Flasch, O., Bartz-Beielstein, T., Weihs, C., and Konen, W. (2012). Tuning and evolution of support vector kernels. *Evolutionary Intelligence*, pages 1–18.

Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 97(1–2):273–324.

Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection*. MIT Press.

Meier, L., van de Geer, S., and Bühlmann, P. (2008). The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71.

Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., and Rudolph, G. (2011). Exploratory landscape analysis. In Krasnogor, N., editor, *Proceedings of the 13th annual conference on genetic and evolutionary computation (GECCO '11)*, pages 829–836. ACM.

Mersmann, O., Bischl, B., Trautmann, H., Wagner, M., Bossek, J., and Neumann, F. (2013). A novel feature-based approach to characterize algorithm performance for the traveling salesperson problem. *Annals of Mathematics and Artificial Intelligence*, pages 1–32.

Mersmann, O., Preuss, M., Trautmann, H., Bischl, B., and Weihs, C. (2012). Analyzing the BBOB results by means of benchmarking concepts. *Evolutionary Computation Journal, accepted*.

Paenke, I., Branke, J., and Jin, Y. (2006). Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation. *Evolutionary Computation, IEEE Transactions on*, 10(4):405–420.

Peng, H., Long, F., and Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1226–1238.

Quinlan, J. R. and Cameron-Jones, R. M. (1995). Oversearching and layered search in empirical learning. In *In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1019–1024. Morgan Kaufmann.

Schiffner, J., Bischl, B., and Weihs, C. (2012). Bias-variance analysis of local classification methods. In Gaul, W., Geyer-Schulz, A., Schmidt-Thieme, L., and Kunze, J., editors, *Challenges at the Interface of Data Analysis, Computer Science, and Optimization*, volume 43 of *Studies in Classification, Data Analysis, and Knowledge Organization*, pages 49–57. Springer.

Schmidberger, M., Morgan, M., Eddelbuettel, D., Yu, H., Tierney, L., and Mansmann, U. (2009). State of the art in parallel computing with R. *Journal of Statistical Software*, 31(1):1–27.

Szepannek, G., Harczos, T., Klefenz, F., and Weihs, C. (2009). Extending features for automatic speech recognition by means of auditory modelling. In *Proceedings of European Signal Processing Conference (EUSIPCO)*, pages 1235–1239.

Tenne, Y. and Armfield, S. (2008). Metamodel accuracy assessment in evolutionary optimization. In *Evolutionary Computation, IEEE Congress on (CEC)*, pages 1505–1512. IEEE Press, Piscataway, NJ.

Thornton, C., Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2012). Auto-WEKA: Automated selection and hyper-parameter optimization of classification algorithms. *CoRR*.

Tibshirani, R. (1994). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288.

Tu, H.-H. and Lin, H.-T. (2010). One-sided support vector regression for multiclass cost-sensitive classification. In *ICML*, pages 1095–1102.

Tutz, G. and Binder, H. (2005). Localized classification. *Statistics and Computing*, 15:155–166.

Vanschoren, J. (2010). *Understanding Machine Learning Performance with Experiment Databases*. PhD thesis, Katholieke Universiteit Leuven.

Vatolkin, J. (2013). *Improving Supervised Music Classification by Means of Multi-Objective Evolutionary Feature Selection*. PhD thesis, Department of Computer Science, TU Dortmund 2013.

Weston, J., Elisseeff, A., Schölkopf, B., and Tipping, M. (2003). Use of the zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, 3:1439–1461.

Eidesstattliche Erklärung
Hiermit erkläre ich, dass ich die vorliegende Dissertation selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Dissertation ist bisher keiner anderen Fakultät vorgelegt worden. Ich erkläre, dass ich bisher kein Promotionsverfahren erfolglos beendet habe und dass keine Aberkennung eines bereits erworbenen Doktorgrades vorliegt.

Bernd Bischl