Tuned Data Mining: A Benchmark Study on Different Tuners

Wolfgang Konen
Cologne University of Applied Sciences,
51643 Gummersbach,
Germany
wolfgang.konen@fh-koeln.de

Patrick Koch
Cologne University of Applied Sciences,
51643 Gummersbach,
Germany
patrick.koch@fh-koeln.de

Oliver Flasch
Cologne University of Applied Sciences,
51643 Gummersbach,
Germany
oliver.flasch@fh-koeln.de

Thomas Bartz-Beielstein
Cologne University of Applied Sciences,
51643 Gummersbach,
Germany
thomas.bartz-beielstein@fh-koeln.de

Martina Friese
Cologne University of Applied Sciences,
51643 Gummersbach,
Germany
martina.friese@fh-koeln.de

Boris Naujoks
Cologne University of Applied Sciences,
51643 Gummersbach,
Germany
boris.naujoks@fh-koeln.de

ABSTRACT

The complex, often redundant and noisy data in real-world data mining (DM) applications frequently lead to inferior results when out-of-the-box DM models are applied. A tuning of parameters is essential to achieve high-quality results. In this work we aim at tuning parameters of the preprocessing and the modeling phase conjointly. The framework TDM (Tuned Data Mining) was developed to facilitate the search for good parameters and the comparison of different tuners. It is shown that tuning is of great importance for high-quality results. Surrogate-model based tuning utilizing the Sequential Parameter Optimization Toolbox (SPOT) is compared with other tuners (CMA-ES, BFGS, LHD) and evidence is found that SPOT is well suited for this task. In benchmark tasks like the Data Mining Cup (DMC) tuned models achieve remarkably better ranks than their untuned counterparts.

Categories and Subject Descriptors

I.2.6 [**Learning**]: Parameter learning

General Terms

Algorithms

Keywords

Parameter tuning, Data Mining, Sequential Parameter Optimization

GECCO'11, July 12–16, 2011, Dublin, Ireland.

1. INTRODUCTION

The practitioner in data mining is confronted with a wealth of machine learning methods containing an even larger set of method parameters to be adjusted to the task at hand. In addition, careful feature selection and feature generation (constructive induction) is often necessary to achieve good quality. This increases the number of possible models to consider even more. How can we find good data mining models with a small amount of manual intervention? Which are general rules that work well for many data mining tasks? It is the aim of the *Tuned Data Mining (TDM)* framework to provide a general framework for adaptive construction of data mining models in a semi-automated or automated fashion. In this paper we describe the first steps undertaken along this alley.

The paper is structured as follows: In the following paragraph and in Sec. 2.1 the TDM framework is described in general. In Sec. 2.2 and Sec. 2.3 methods for numerical feature preprocessing and generic feature selection are presented. In Sec. 2.4 it is described that additional parameters may arise when the problem is cost-sensitive while Sec. 2.5 gives a short overview about the model-based parameter optimization framework SPOT. Sec. 3 presents the results on three benchmark tasks. Sec. 4 discusses these results with emphasis on comparison of tuning algorithms and feature processing. We conclude our findings in Sec. 5.

Features of TDM.

The goal of TDM [14] for classification and regression tasks can be formulated as follows: Find a recipe / template for a generic data mining process which works well on many data mining tasks. More specifically:

- Besides from reading the data and task-specific data cleansing, the template is the same for each task. This makes it easily reusable for new tasks.
- Well-known machine learning methods, e.g., Random Forest (RF) [6, 18] or Support Vector Machines (SVM) [23, 27] available in `R` are reused within the `R`-based

template implementation, and the template is open to the integration of new user-specific learning methods.
- Feature selection and/or feature generation methods are included in a systematic way within the optimization / tuning loop.
- Parameters are either set by general, non-task-specific rules or they are tuned by an automatic tuning procedure. We propose here to use SPOT (**S**equential **P**arameter **O**ptimization **T**oolbox) [2, 3] in its recent `R`-implementation [1]. A comparison with other tuning algorithms is possible.

The interesting point from a learning perspective is: Given a certain data mining model, is it possible to specify *one* set of tunable parameters together with their ROI (region of interest) such that for several challenging data mining tasks a high-quality result is reached after tuning? If the answer to this question is 'Yes', we can combine machine learning and its parameter tuning in a black-box fashion which will facilitate its wide-spread use in industry and commerce.

Related work.

There are several other data mining frameworks with a similar scope in the literature, e.g., ClearVu Analytics [12], Gait-CAD [21], MLR [4], and RapidMiner [20]. We plan to compare our findings with results from these frameworks at a later point in time. Bischl et al. [5] have recently given an interesting overview of well-known resampling strategies. Their findings that careful model validation is essential to avoid overfitting and oversearching in tuning is compatible with similar findings in [15].

To our knowledge, SPOT was used for systematic parameter tuning in data mining by Konen et al. [15, 17] for the first time. Here, we extend those results in two directions:

a) inclusion of a new benchmark task (appAcid) with a high number of features putting emphasis on feature selection, feature preprocessing and its tuning, and

b) the comparison of SPOT with other tuning algorithms.

A tuning algorithm (or short: tuner) is a method to find optimal values for a parameter set, often within a prescribed region of interest (ROI). We consider the following tuners alternatively to SPOT: As a baseline tuner a strategy based on random search with Latin hypercube design (LHD) is used, where the total budget is spent by placing trial points in the ROI with equal density. Local-search methods like the algorithm of Broyden, Fletcher, Goldfarb and Shanno (BFGS) [7, 8] are other possible choices from classical optimization. As state-of-the-art evolution strategy we consider the Covariance Matrix Adaptation ES (CMA-ES) by Hansen et al. [11]. This choice is motivated by the good reputation of this ES as a numerical optimizer and the tuning parameters considered in our DM tasks are mostly numeric. The REVAC tuning method of Nannen and Eiben [22] was compared with CMA-ES by Smit et al. [24]. REVAC is a good tuning alternative to SPOT and we plan to include it in the future.

2. METHODS

2.1 Tuned Data Mining (TDM) Template

We consider classification tasks, but the approach can be — and is in the TDM framework — easily generalized to regression tasks as well. If we have a preprocessed data set, the following steps of the data mining process can be formulated in a generic way [14]:

Data Mining Template:
- Sampling, i.e., the division of the data in training and test set (random, k-fold cross validation (CV), ...)
- Generic feature generation (Sec. 2.2) and generic feature selection (Sec. 2.3, currently RF-based variable ranking and GA)
- Modeling: currently SVM, RF, MC.RF (see Sec. 2.4), but other models, especially all those available in `R` can easily be integrated
- Model application: predict class and (optional, depending on model) class probabilities
- User-defined postprocessing (optional)
- Evaluation of model: confusion matrix, gain matrix, score, generic visualization, ...

All these steps are controlled by general or model-specific parameters. Some of these parameters may be fixed by default settings or by generic rules. Other parameters usually need task-specific optimization, a process which is generally referred to as "tuning". With a general-purpose tuner like SPOT (cf. Sec. 2.5) or other tuners it is possible to embed the above data mining template in a tuning optimization loop:

Tuned Data Mining Template:
while (budget not exhausted) **do**
  Choose parameter values ('design points')
  to be optimized by tuner.
  Run data mining template with these values,
  Report and return results to tuner
end while

One point concerning the tuning part deserves further attention: The above pseudo-code makes it necessary that — given a model — both (a) the parameter set to be tuned and (b) the parameter range (ROI = region of interest) have to be prescribed beforehand. The question whether *one* such triple {model, parameter set, ROI} fits for a large variety of tasks, can only be answered by experiments. (If no *one* such triple for all tasks can be found, a somewhat weaker requirement can be formulated: Is there a collection of multiple such triples, which covers all tasks? Can we make a decision, based solely on training data [15], which of those triples gives high-quality results also for unseen test data?).

2.2 Generic feature generation

As generic choices for numeric feature preprocessing we consider the following options:
- Principal Component Analysis (PCA) as a standard method to decorrelate highly correlated inputs, combined with the option to select only the first few principal components (PCs) with large eigenvalues.
- Nonlinear feature generation: add all monomials of degree 2 for the first N_{PC} principal components. More specifically, if $\vec{p}^{(i)}$ is the vector of PC i and $p_k^{(i)}$ is its kth element, then we form new vectors $\vec{m}^{(ij)}$ with

$$m_k^{(ij)} = p_k^{(i)} p_k^{(j)}, \quad i,j = 1,\ldots,N_{PC},\ i<j \qquad (1)$$

  and let the feature selection algorithm choose the most appropriate features from the union {principal components, monomials}.

These choices are of course only a first step, and we plan to include further feature-generating operators in a more

complete framework. Parameters of the feature generation like N_{PC} can be included in the tuning loop.

2.3 Generic feature selection

Selecting the right features is often of great importance for high-quality results in data mining.[1] Standard approaches like sequential forward selection or sequential backward elimination [19] allow quite accurate selection of the right features for a certain model. But they have the disadvantage of high computational costs of $O(N^2)$, where N is the number of input variables.

Another option is variable ranking, where a certain pre-model (e.g. a RF with reduced number of trees) allows to rank the input variables according to their importance. Given this importance (where it is a tacit assumption that the importance of the pre-model is also representative for the full model), it is possible to transform the combinatorial feature selection problem into a simpler numeric optimization problem which has moderate computational costs for arbitrary numbers of input variables:

> *Importance selection rule:* Sort the input variables by decreasing importance $I_n, n = 1, \ldots, N$ and select the first K variables such that
>
> $$\sum_{n=1}^{K} I_n \geq X_{perc} \sum_{n=1}^{N} I_n \qquad (2)$$

This means that we select those K variables which capture at least the fraction $X_{perc} \in [0, 1]$ of the overall importance.

We use the importance delivered by `R`'s `randomForest` package [18] in our current implementation, but other importance measures could be used equally well.

Another option for feature selection are Genetic Algorithms (GA) [10], which are population based optimization algorithms using binary strings to represent candidate solutions. Each bit k of the binary string defines, whether the k-th feature of the overall feature set is selected as model input or not. The initial population can be drawn randomly or initialized by prior knowledge, e.g., a certain bias can be given to the probability that features are selected, when some basic importance about the features is known in advance. Starting from the initial population individuals are varied by means of recombination and mutation. The best solutions of parents and offspring are taken into the next generation (survivor selection). The prediction error on an independent validation set is usually used as an objective function for the GA. GA-based feature selection is more time consuming than RF-based variable ranking, but worthwhile if the latter does not yield good classification results.

Feature selection by means of GA (FS-GA) was run five times given a total budget of 250 generations as a stopping criterion which led to convergence of the fitness improvement in all runs. The population size was 5 generating 20 offspring. As recombination operator uniform crossover was performed with probability 0.8. As mutation operator we used a simple bit-flip mutation with one bit flip on average per string. No adaptation of probabilities for the variation operators was considered in our study. In future work we plan to investigate more sophisticated GA approaches with adaptive strategy parameters or other termination criteria.

[1]In this paper the term *feature* may refer to either an input variable or a derived feature, e.g., along the lines described in Sec. 2.2.

Table 1: Gain matrices for the tasks DMC-2007 and DMC-2010. Example: Predicting a true "1" as a "0" in DMC2010 has a negative gain -5. A problem with varying off-diagonal matrix elements or with varying diagonal matrix elements is called *gain-sensitive*.

DMC 2007		predict (p)		
		A	B	N
true (t)	A	3	-1	0
	B	-1	6	0
	N	-1	-1	0

DMC 2010		predict	
		0	1
true (t)	0	1	0
	1	-5	0

2.4 Cost-sensitive modeling

Many classification problems require cost-sensitive or equivalently gain-sensitive modeling. This is the case if the cost (or negative gain) for different misclassifications differs or if the gain for correct classifications differs, see for example Tab. 1. Advanced classification algorithms can be made gain-sensitive by adjusting different parameters. For example in RF the following options are available (N_c is the number of classes):

CLASSWT: a class weight vector with length N_c indicating the importance of class i.

CUTOFF: a vector c_i with length N_c and sum 1 specifying that the predicted class i is the one which maximizes v_i/c_i where v_i is the fraction of trees voting for class i. The default is $c_i = 1/N_c \; \forall i$.

Manual adjustment is difficult, because many parameter settings have to be considered and suitable values depend on the gain matrix and the a-priori probability of each class in a non-trivial manner. Therefore, careful tuning of those parameters is often of great importance to reach high-quality results.

An alternative to task-specific parameter tuning are wrapper models which can turn any (cost-insensitive) base model into a cost-sensitive meta model. An example is the well-known MetaCost algorithm [9]. The implementation in [15] is an RF-based version of MetaCost which we abbreviate with MC.RF in the following. Due to space constraints we refer the reader to [15] for details on MC.RF.

2.5 Generic tuning with SPOT

SPOT provides tools for tuning many parameters simultaneously [3]. It is well-suited for optimization problems with noisy output functions (as they occur frequently in data mining) and it can reach good results with only a few model-building experiments since it builds a surrogate model during its sequence of runs. This is constantly refined as the tuning progresses. SPOT has recently been made available as an `R`-package [1].

After some initial experiments, the set of parameters and ROIs as specified in Tab. 2 were used for all the results reported below. We have $3, 5, 7, \ldots$ parameters for a ($N_c = 2, 3, 4, \ldots$)-class problem, since one of the parameters in each vector CUTOFF and CLASSWT is fixed by a constraint:

$$CUTOFF[N_c] = 1 - \sum_{i=1}^{N_c - 1} CUTOFF[i]$$

and $CLASSWT[N_c] = 10$. Infeasible solutions, e.g. those where the sum of $CUTOFF$ exceeds 1, are transformed by

Table 2: Tunable parameters and their ROI for the classification models RF and MC.RF. Index $i \in 1, \ldots, N_c - 1$, where N_c is the number of classes. As an example the best tuning results from Sec. 3 for DMC-2010 are shown in column "best".

	RF		MC.RF	
	ROI	best	ROI	best
CUTOFF[i]	[0.1,0.8]	0.734	[0.1,0.8]	0.448
CLASSWT[i]	[2.0,18]	5.422	[2.0,18]	4.6365
XPERC	[0.05,1.0]	0.999	[0.05,1.0]	0.9505

Table 3: Task overview

Task	records (training / test)	inputs	classes	cost-sensitive?
DMC-2007	50000 / 50000	20	3	yes
DMC-2010	32428 / 32427	37	2	yes
appAcid	3326 / 1109	212	5	(yes)[2]

[2] indirectly through MCA, see Sec. 3.3

appropriate scaling to feasible solutions (e.g., dividing by the sum). It would be more convenient to use a real linear constraint term here, but the considered implementations of the tuning algorithms don't provide such an option yet.

All SPOT-tuning experiments for the DMC (Data Mining Cup)-tasks are performed with the following settings (see [1] for further details): 50 sequence steps, 3 new design points in each step, up to 5 repeats per design point (to dampen statistical fluctuations), and 10 initial design points. This leads to 747 data mining models to be built for each experiment. In the appAcid task we restricted the number of model evaluations to up to 200, with 2 repeats per design point. For the SPOT metamodel RF was used as a fast surrogate model building tool, but other techniques such as Kriging could have been used as well.

3. RESULTS ON BENCHMARK TASKS

The benchmark tasks studied in this paper are briefly summarized in Tab. 3. The two different DMC competitions [13] with their realistic size (65,000 and 100,000 records, 20 and 37 input variables, respectively) provide interesting benchmarks as they go beyond the level of toy problems. Many comparative results from other teams participating in the Data Mining Cup allow to gauge the quality of our results achieved with the general template. The appAcid benchmark task is a DM application from engineering featuring a quite large number (212) of highly correlated input variables.

Note that in all results described below no task-specific model adjustment or task-specific postprocessing has taken place. Only the general TDM framework with its general models and one ROI for the tuning of each model (see Tab. 2) has been used.

Each tuning experiment was performed in the following way: For each task the training data set was further divided into training records and validation records (cross-validation (CV) in the case of SVM and out-of-bag (OOB) error estimate [18] in the case of RF). For each parameter setting defined by a design point of the tuner a model was trained on the training records and evaluated on the validation records. The validation accuracy was used as optimization signal for the tuner. The final best parameter setting found by the tuner was used to train a model on the full training data set. Its accuracy was evaluated on the independent and unseen test data set (column TST in Tab. 5; symbols annotated with TST in 3).

3.1 DMC-2007

DMC-2007 is a three-class, cost-sensitive classification task with the gain matrix shown in Tab. 1, left. The data consists of 50000 training records 50000 test records and with 20 attributes. Class N, with 76% , has a much higher frequency than the other classes A and B, but only A and B classified correctly will contribute positively to the gain. The DMC-2007 contest had 230 participants whose resulting score distribution is shown in Fig. 1 as boxplot (we removed 13 entries with score < 0 in order to concentrate on the important participants). Our results from different models are overlayed as horizontal lines and arrows to this diagram. We can learn from this:

- Using the default parameters in RF or MC.RF gives only bad results, well below the mean of the DMC participants' distribution. This is no surprise for the base RF[3], because it minimizes the misclassification error and is thus not well-suited for a cost-sensitive problem. But it is a surprise for MC.RF which is supposed to behave optimally in the presence of cost-sensitive effects [15].
- The tuned results delivered by SPOT are much better: Model RF.tuned reaches the highest first quartile and the results of model MC.RF.tuned are close to this quartile. It is thus crucial to tune CLASSWT and CUTOFF for cost-sensitive problems.
- The CV estimate of the total gain (red dashed line) is in good agreement with the final gain (blue arrows).

Note that hand-tuning of CLASSWT and CUTOFF usually leads to gains in the range of 6000–7000, and it is in general a very time-consuming task since no good rule-of-thumb exist for these parameters.

3.2 DMC-2010

DMC-2010 is a two-class, cost-sensitive classification task with the gain matrix shown in Tab. 1, right. The data consists of 32428 training records and 32427 test records with 37 attributes. Class 0 is with 81.34% of all trainings records much more frequent than class 1. Given this a-priori probability and the above gain matrix, there is a very naïve model "always predict class 0" which gives a gain of $32428 \cdot (1.5 * 81.34\% - 5 * 18.66\%) = 9310$ on the training data. Any realistic model should do better than this.

The data of DMC-2010 require some preprocessing, because they contain a small fraction of missing values, some obviously wrong inputs and some factor variables with too many levels which need to be grouped. This task-specific data preparation was done beforehand.

Altogether 67 teams participated in the DMC-2010 contest whose resulting score distribution is shown in Fig. 2 as boxplot (we removed 26 entries with score < 5000 or NA in

[3] with CLASSWT=CUTOFF=NULL

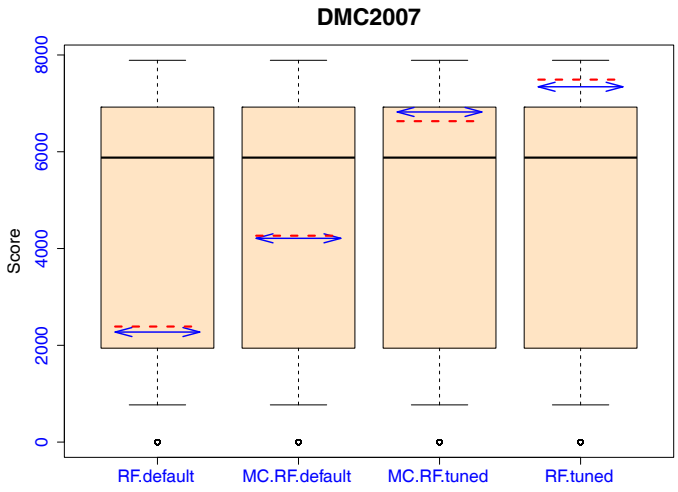

Figure 1: Results for the DMC-2007 benchmark: The boxplot shows the spread of score (gain) among the competition participants, the red dashed lines show the score of our models on the training data (10-fold CV), the blue arrows show the score of these trained models on the real test data.

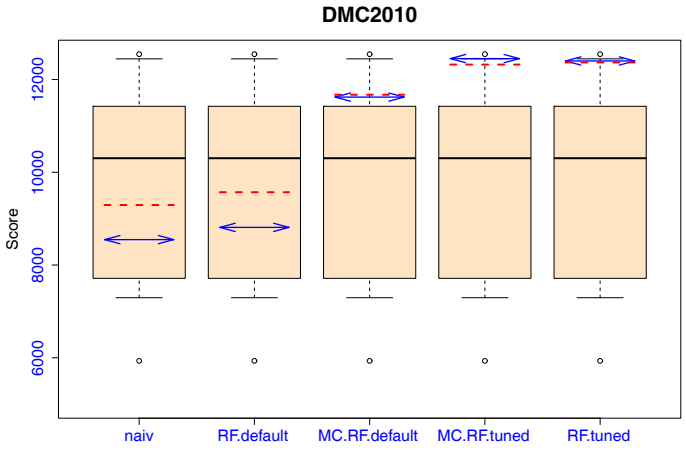

Figure 2: Results for the DMC-2010 benchmark.

order to concentrate on the important teams). Our results from different models are overlayed as horizontal lines and arrows in this diagram. We can learn from this:

- The model RF.default is not significantly better than the naïve model. Indeed it behaves nearly identical to the naïve model in an attempt to minimize the misclassification error.
- Except for the naïve model, the CV estimates of the total gain (red dashed lines) are again in good agreement with the final gain (blue arrows).
- MC.RF.default shows a competitive performance in this setup (at the lower rim of the highest quartile), but both tuned models achieve again considerably better results: They are at the upper rim of the highest quartile; within the rank table of the real DMC-2010 contest this corresponds to rank 2 and rank 4 for MC.RF.tuned and RF.tuned, resp.

3.3 appAcid

Acid concentrations in the fluid of a plant are to be classified in this benchmark, based solely on spectroscopy data [28]. In the appAcid task there are five defined classes, each denoting a certain range of acid concentration. Table 4 shows that the record numbers R_c for each class are highly unbalanced. The user-defined goal is to maximize the mean class accuracy

$$MCA = \frac{1}{5}\sum_{c=1}^{5}\frac{1}{R_c}\sum_{i=1}^{R_c}L(\vec{x}_i) \tag{3}$$

where $L(\vec{x}_i)$ is 1 for each correctly predicted record $\vec{x}_i$ and 0 otherwise. This means that each of the 70 records of class 5 (they define a critical plant state) has a much higher importance than one of the 1880 records of class 3. Thus the benchmark is also indirectly cost-sensitive although the gain matrix is the unit matrix in this case.

Table 4: Number of records belonging to each class in the appAcid dataset.

Class c	Number of records R_c
1	228
2	1528
3	1880
4	731
5	70

The research question here is whether classification methods based on TDM can achieve a similar or even better performance than the GerDA results reported in [28]. GerDA, as described in [25, 29], learns unsupervisedly interesting feature combinations with an approach based on Boltzmann machines (it has to be noted that in [28] the classifier superimposed on the GerDA features was optimized for the overall misclassification rate instead of MCA). All results are substantially better than the baseline Linear Discriminant Analysis (LDA).

We show in Fig. 3 a comparison of different tuners for the RF learning algorithm. The results of SVM are comparable to this, for more details we refer the interested reader to [16]. Each point in Fig. 3 denotes the mean value from 5 repeated tuning experiments with different random seeds. The error bars denote the corresponding standard deviations. In

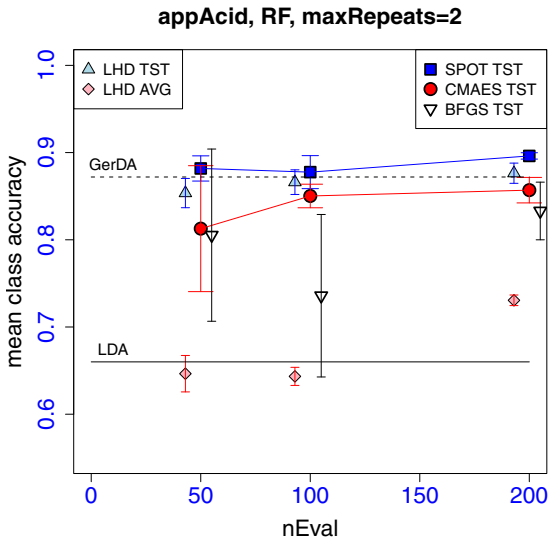

Figure 3: Results of TDM-based RF tuning for the appAcid problem. See text for legend explanation.

each tuning experiment the relevant tuner has a budget of nEval model trainings to find good parameters for the tunable parameter set. nEval is deliberately set to quite low values, since the model training is the time-consuming part of the tuning process. Legend: The symbols annotated with { SPOT | CMAES | BFGS | LHD } TST show the MCA on the independent test set after tuning with these different tuners. LHD AVG: average MCA of all design points visited during tuning with the LHD tuner (shown also in Tab. 6 together with the corresponding numbers for the other tuners).

Besides the tuning of the CUTOFF and CLASSWT parameters for RF we also incorporated the number of trees *ntree* and number of proposed splits *mtry* in the tuning process. For the SVM model parameters as RBF kernel parameter γ and regularization parameter C were optimized instead of these RF specific parameters.

The results from all three tasks are summarized in Tab. 5.

4. DISCUSSION

4.1 Comparison of SPOT and LHD

It is a striking feature of our experiments that the LHD tuner, which simply performs a Latin hypercube design random search with the same budget as all other tuners, reaches results similar to the best tuner SPOT and better than the other ones (CMA-ES, local search). This is however true only for the best tuning result. In average the performance is of course much lower for LHD than for SPOT (see Tab. 6). SPOT places more design points in the 'interesting' region with the help of its surrogate model. This may be valuable for other tasks where the interesting region contains small local minima that are not easy to detect. However, such small local minima seem not to be present in our task appAcid.

4.2 Comparison of SPOT, CMA-ES, and BFGS

Surprisingly, although CMA-ES has a good reputation as a general-purpose numerical optimizer, it does not perform as well as SPOT or LHD on the data mining tasks considered. The reasons for this behavior may be twofold. Firstly, the budget, i.e., the number of function evaluations is rather low and the response function is noisy which is not in favor of the matrix adaptation needed for good CMA-results. Secondly, most of the tuning parameters have tight constraints (see Tab. 2). The CMA-ES has known problems if the border of the ROI is crossed: A constraint-enforcing extra term can lead to a local minimum at the border. Indeed, we often observed that a 'best' solution found by CMA has parameter values exactly at the ROI border.

Table 6: Average MCA of all design points visited during tuning for task appAcid with the tuners SPOT, CMA-ES and LHD (budget nEval=100).

model	SPOT	CMA-ES	LHD
RF	**81.4%**	58.9%	64.3%
SVM	**75.5%**	48.1%	44.5%

BFGS as a purely local-search optimizer performs slightly worse than CMA-ES, sometimes with outliers considerably worse. However, BFGS was the best among several other local-search algorithms tested in preliminary experiments.

4.3 Optimality Conditions of SPOT

One might ask whether SPOT as a global optimizer does a good job in finding the local optimum in the vicinity of the best solution selected by SPOT. An experiment was conducted to find out if a local search starting from this best solution can produce better results compared to the already optimized SPOT parameters. In general, any method for numerical local optimization can be used, but only methods allowing box constraints and guaranteed convergence to local optima are well-suited for this task.

We used a setup called *hybridization* between global and local optimizer strategies. In the literature many publications describe hybridizations of metaheuristics and hill-climbing strategies. For the sake of simplicity we follow the taxonomy of Talbi [26], so that we can term our algorithm a high-level relay hybrid.

For local optimization we used an extended version of the well-known BFGS algorithm by Byrd et al. [8], which allows box-constraints. The best parameter setting found by SPOT with nEval=200 for the appAcid problem with classifier RF was used as the starting point. BFGS was initialized with this parameter setting and was run five times.

The result was negative in the sense that it showed oversearching effects: Although BFGS might find slight improvements in the validation set accuracy used as the target for the tuner, the resulting parameter set had an MCA on the independent test set *worse* by 0.5%-1.0% compared to the MCA of the best SPOT solution. We conclude that extensive local search does not pay off for the noisy optimization environment usually encountered in data mining tasks.

4.4 Feature processing revisited

The TDM approach presented in this study, which uses generic feature-processing and -selection methods, performs competitive to GerDA [25, 28]. GerDA implements a very sophisticated feature generation approach. It might be interesting to ask, *which* TDM feature processing elements contribute most to this success. Table 7 shows that turning

Table 5: Results compared for the SPOT-tuned models (budget nEval=200) and the benchmark tasks considered. The result (gain for the DMC-tasks, MCA for appAcid) has to be maximized. CV: cross-validated result on the training set, TST: result on the independent test set. Each cell contains mean ± standard deviation from 5 repeated runs with different random seeds. The "Rank DMC" column a/b is the rank a of TST-result within the real DMC result table with b entries.

DMC		Result		Rank
Year	Model	CV	TST	DMC
2007	RF.tuned	7491 ± 24	$\mathbf{7343} \pm 38$	37/230
	MC.RF.tuned	6632 ± 33	6822 ± 131	61/230
2010	RF.tuned	12368 ± 83	12400 ± 23	4/67
	MC.RF.tuned	12322 ± 94	$\mathbf{12451} \pm 103$	2/67

appAcid	Result	
Model	CV	TST
RF.tuned	$(88.2 \pm 0.4)\%$	$(\mathbf{89.9} \pm 0.3)\%$
MC.RF.tuned	$(87.8 \pm 0.3)\%$	$(88.8 \pm 0.9)\%$
SVM.tuned	$(86.4 \pm 0.6)\%$	$(86.1 \pm 1.5)\%$
GerDA [28]		87.2

off the extra feature generation (monomials) gives a 3.2% decrease in accuracy, turning off PCA gives a larger decrease (6.5%), but the largest decrease in accuracy (7.3%) occurs if we turn off the feature selection (FS), see line 6 in Tab. 7. FS-SRF reduces the feature set to about 40 out of 257 features, while the GA feature selection (FS-GA) selects even less features, between 8 and 19 in the best individuals of the five GA runs.

The GA feature selection results are comparable with the FS-SRF approach. This is however only true if a biased initialization procedure was used to generate the starting population: The first 15 PCA features with the largest eigenvalues were selected with higher probability than the other features with lower eigenvalues. If, contrarily, the starting population had all features selected with the same probability, then the GA would usually stop in a local minimum with roughly half of the features selected and with a MCA of only 85%. The advantage of the biased procedure can clearly be seen in the best individuals of the five GA runs: The principal components with the highest and 2nd highest eigenvalue are selected in every best individual, and monomials between principal components with highest eigenvalues are also selected more frequently compared to other features.

We conclude from the GA experiments that, if such prior knowledge is used for the creation of the initial population, then the GA is capable to find a good working feature subset for the model.

The experiments demonstrate the importance of good feature selection (FS). But using *only* FS is also suboptimal, as line 5 in Tab. 7 shows. The overall best result is achieved only if all three elements PCA, monomials and FS are present.

Table 7: Class accuracy MCA of best tuning solution (budget nEval=200) on task appAcid when different feature processing elements are activated. FS-SRF: feature selection based on the sorted RF-importance, FS-GA: GA-based feature selection.

	PCA	monomials	FS- SRF	FS- GA	class accuracy
1	X	X	X	-	$(89.95 \pm 0.41)\%$
2	X	X	-	X	$(89.47 \pm 0.52)\%$
3	X	-	X	-	$(86.72 \pm 0.77)\%$
4	-	X	X	-	$(83.38 \pm 0.78)\%$
5	-	-	X	-	$(82.90 \pm 1.35)\%$
6	X	X	-	-	$(82.60 \pm 0.92)\%$
7	-	-	-	-	$(82.59 \pm 0.42)\%$

5. CONCLUSION

This paper has shown first steps towards a general, self-adaptive data mining framework which combines feature selection, model building and parameter tuning within one integrated optimization environment. We have studied with TDM three challenging classification tasks with cost-sensitivity where standard models using default parameters do not achieve high-quality results. This puts the necessity of parameter tuning for data mining into focus: We have shown:

1. Parameter tuning with SPOT gives large improvements. In the case of DMC-2010, the untuned RF model had rank 21 out of 67 in the DMC ranking table. With tuning the RF model could be boosted to rank 4, the MC.RF model to rank 2 out of 67 (Fig. 2).
2. At least for our three benchmark classification tasks with their quite different characteristics we were able to show that *one* generic template with *one* parameter set and ROI is sufficient to achieve high-quality results.
3. For DM tasks containing noise and constraints, it seems that SPOT and LHD as non-local tuners perform better than CMA-ES or the local-search method BFGS.
4. Furthermore Sec. 4.3 has collected evidence that the final solution delivered by SPOT cannot be improved by a relayed local search.
5. Feature selection is essential, especially for tasks with a large number of inputs. Sophisticated feature selection schemes like GA do not show much benefit over less computing-intensive variable ranking schemes in our case. However, GAs can be a good option if all other feature selection methods do not work very well for some reason.

In future work we want to compare the TDM framework with other DM frameworks with a similar scope [4, 12, 20, 21]. One benefit of the generic TDM approach is already visible now: If *one* framework can be used for very different tasks, then it is easy to transfer the processing elements which are found useful for one task to the other tasks. This speeds up the search for good solutions considerably.

6. REFERENCES

[1] T. Bartz-Beielstein. SPOT: An R package for automatic and interactive tuning of optimization algorithms by sequential parameter optimization. Technical Report arXiv:1006.4645. CIOP Technical

Report 05-10, Cologne University of Applied Sciences, Jun 2010.

[2] T. Bartz-Beielstein, C. Lasarczyk, and M. Preuß. Sequential parameter optimization. In B. McKay et al., editors, *Proceedings 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland*, volume 1, pages 773–780, Piscataway NJ, 2005. IEEE Press.

[3] T. Bartz-Beielstein, C. Lasarczyk, and M. Preuss. The sequential parameter optimization toolbox. In Bartz-Beielstein et al., editors, *Experimental Methods for the Analysis of Optimization Algorithms*, pages 337–360. Springer, Berlin, Heidelberg, New York, 2010.

[4] B. Bischl. The `mlr` package: Machine learning in R. `http://mlr.r-forge.r-project.org`, accessed 14.04.2011.

[5] B. Bischl, O. Mersmann, and H. Trautmann. Resampling methods in model validation. In T. Bartz-Beielstein et al., editors, *Workshop WEMACS joint to PPSN2010*, number TR10-2-007 in Technical Reports, TU Dortmund, 2010.

[6] L. Breiman. Random forests. *Machine Learning*, 45(1):5 –32, 2001.

[7] C. Broyden, J. Dennis, and J. Moré. On the local and superlinear convergence of quasi-Newton methods. *IMA Journal of Applied Mathematics*, 12(3):223–245, 1973.

[8] R. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[9] P. Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining (KDD-99)*, pages 195–215, 1999.

[10] D. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-wesley, 1989.

[11] N. Hansen. The CMA evolution strategy: a comparing review. In J. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, editors, *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer, 2006.

[12] F. Jurecka. Automated metamodeling for efficient multi-disciplinary optimization of complex automotive structures. In *7th European LS-DYNA Conference, Salzburg, Austria*, 2009.

[13] S. Kögel. Data Mining Cup DMC. `http://www.data-mining-cup.de`, accessed 14.04.2011.

[14] W. Konen. The TDM framework: Tuned data mining in R. CIOP Technical Report 01-11, Cologne University of Applied Sciences, Jan 2011.

[15] W. Konen, P. Koch, O. Flasch, and T. Bartz-Beielstein. Parameter-tuned data mining: A general framework. In F. Hoffmann and E. Hüllermeier, editors, *Proceedings 20. Workshop Computational Intelligence*. Universitätsverlag Karlsruhe, 2010.

[16] W. Konen, P. Koch, O. Flasch, T. Bartz-Beielstein, M. Friese, and B. Naujoks. Tuned data mining: A benchmark study on different tuners. CIOP Technical Report 02-11, Cologne University of Applied Sciences, Feb 2011.

[17] W. Konen, T. Zimmer, and T. Bartz-Beielstein. Optimized modeling of fill levels in stormwater tanks using CI-based parameter selection schemes (in german). *at-Automatisierungstechnik*, 57(3):155–166, 2009.

[18] A. Liaw and M. Wiener. Classification and regression by randomForest. *R News*, 2:18–22, 2002. `http://CRAN.R-project.org/doc/Rnews/`.

[19] H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17:491–502, 2005.

[20] I. Mierswa. Rapid Miner. `http://rapid-i.com`, accessed 14.04.2011.

[21] R. Mikut, O. Burmeister, M. Reischl, and T. Loose. Die MATLAB-Toolbox Gait-CAD. In R. Mikut and M. Reischl, editors, *Proceedings 16. Workshop Computational Intelligence*, pages 114–124, Karlsruhe, 2006. Universitätsverlag, Karlsruhe.

[22] V. Nannen and A. E. Eiben. Efficient relevance estimation and value calibration of evolutionary algorithm parameters. In *IEEE Congress on Evolutionary Computation*, pages 103–110, 2007.

[23] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 12 2002.

[24] S. K. Smit and A. E. Eiben. Comparing Parameter Tuning Methods for Evolutionary Algorithms. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 399–406, May 2009.

[25] A. Stuhlsatz, J. Lippel, and T. Zielke. Feature extraction for simple classification. In *Proc. Int. Conf. on Pattern Recognition (ICPR), Istanbul, Turkey*, page 23, 2010.

[26] E. Talbi. A taxonomy of hybrid metaheuristics. *Journal of heuristics*, 8(5):541–564, 2002.

[27] V. N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, September 1998.

[28] C. Wolf, D. Gaida, A. Stuhlsatz, T. Ludwig, S. McLoone, and M. Bongards. Predicting organic acid concentration from UV/vis spectro measurements - a comparison of machine learning techniques. *Trans. Inst. of Measurement and Control*, 2011.

[29] C. Wolf, D. Gaida, A. Stuhlsatz, S. McLoone, and M. Bongards. Organic acid prediction in biogas plants using UV/vis spectroscopic online-measurements. *Life System Modeling and Intelligent Computing*, 97:200–206, 2010.