Mach Learn
DOI 10.1007/s10994-012-5286-7

Meta-learning for evolutionary parameter optimization of classifiers

Matthias Reif · Faisal Shafait · Andreas Dengel

Received: 16 September 2010 / Accepted: 16 March 2012

Abstract The performance of most of the classification algorithms on a particular dataset is highly dependent on the learning parameters used for training them. Different approaches like grid search or genetic algorithms are frequently employed to find suitable parameter values for a given dataset. Grid search has the advantage of finding more accurate solutions in general at the cost of higher computation time. Genetic algorithms, on the other hand, are able to find good solutions in less time, but the accuracy of these solutions is usually lower than those of grid search.

This paper uses ideas from meta-learning and case-based reasoning to provide good starting points to the genetic algorithm. The presented approach reaches the accuracy of grid search at a significantly lower computational cost. We performed extensive experiments for optimizing learning parameters of the Support Vector Machine (SVM) and the Random Forest classifiers on over 100 datasets from UCI and StatLib repositories. For the SVM classifier, grid search achieved an average accuracy of 81 % and took six hours for training, whereas the standard genetic algorithm obtained 74 % accuracy in close to one hour of training. Our method was able to achieve an average accuracy of 81 % in only about 45 minutes. Similar results were achieved for the Random Forest classifier. Besides a standard genetic algorithm, we also compared the presented method with three state-of-the-art optimization algorithms: Generating Set Search, Dividing Rectangles, and the Covariance Matrix Adaptation Evolution Strategy. Experimental results show that our method achieved the highest average accuracy for both classifiers. Our approach can be particularly useful when training classifiers on large datasets where grid search is not feasible.

Editor: Hendrik Blockeel.

M. Reif (✉) · F. Shafait · A. Dengel
German Research Center for Artificial Intelligence, Trippstadter Straße 122, 67663 Kaiserslautern, Germany
e-mail: matthias.reif@dfki.de

F. Shafait
e-mail: faisal.shafait@dfki.de

A. Dengel
e-mail: andreas.dengel@dfki.de

Published online: 13 April 2012

Keywords Meta-learning · Parameter optimization · Genetic algorithm · Feature selection

1 Introduction

In pattern recognition, classifiers are trained on labeled data to produce a function that maps data points from the feature space to the label space. This function is used to predict the label of a new data point. Typically, classifiers contain several parameters, which influence their performance. A typical performance measure is the classification accuracy that is the rate of correctly predicted labels. For one single dataset, the optimal parameter setting for the classifier maximizes the performance on that dataset. The task of finding these optimal parameter values is often called parameter tuning or parameter optimization and is a crucial step for obtaining good results from most of the classifiers. Parameter tuning has several aspects, which make it hard in general:

- Parameter settings that result in good performance values for one dataset may not lead to good results for another dataset.
- The parameter values usually depend on each other. It is not sufficient to optimize multiple parameters independently.
- The evaluation of one parameter setting can be already very time-consuming. Most of the classifiers must be trained from scratch again.
- The search space can be huge due to several facts: (1) A classifier can have many parameters. (2) The range of the parameters may be large. (3) Real-valued parameters can be sensitive to very small changes.

Many deterministic and probabilistic approaches with different complexities and worst case guarantees have been used for parameter optimization of classifiers. A widely used deterministic method for parameter optimization is grid search because it is simple and typically delivers good results. However, if the run-time of the optimization is important, it may not be feasible anymore. A large number of classifier evaluations have to be performed using this exhaustive method. Some authors have used more sophisticated deterministic optimization algorithms such as the Dividing Rectangles (DIRECT) algorithm (Jones et al. 1993) or Quasi-Newton methods (Ayat et al. 2002). However, grid search remains the most widely used method as long as its computational demands can be met.

For high-dimensional optimization problems or training on large datasets, grid search becomes computationally infeasible. In such scenarios, probabilistic optimization methods such as genetic algorithms (Goldberg 1989) are usually preferred. For instance, architecture, weights, and learning parameters of artificial neural networks were optimized using such evolutionary approaches by Yao (1993). The Covariance Matrix Adaptation Evolution Strategy (Hansen and Ostermeier 2001) is a modification of standard genetic algorithm that updates the covariance matrix used for selecting the next generation. Friedrichs and Igel (2005) used this method to optimize multiple parameters of Support Vector Machines (SVM). Merelo et al. (2001) presented G-LVQ, which uses a genetic algorithm to optimize Learning Vector Quantization (Kohonen 1998). Other probabilistic methods have also been used for parameter optimization of classifiers. For example, to optimize parameters of an SVM, Pattern Search (Kolda et al. 2003) was used by Momma and Bennett (2002) as well as by Eitrich and Lang (2006), and a gradient descent-based approach was used by Chapelle et al. (2002).

The results achieved using genetic optimization depend on the choice of the start population (see Sect. 2 for details). It is known that the performance of a genetic algorithm can be increased if good starting points are used. It was also shown that manually selecting good start points for the Nelder-Mead simplex optimization algorithm increases its performance

(Shafait and Breuel 2011). In this paper, we present an approach that uses experience to find good start points for the evolutionary parameter optimization of classifiers. The approach uses concepts from meta-learning and adapts them to parameter optimization.

Meta-learning typically tries to predict the applicability of different classification algorithms on a dataset, either by predicting only the best algorithm (Bensusan and Giraud-Carrier 2000a; Ali and Smith 2006), predicting a ranking of multiple algorithms (Merz 1995; Berrer et al. 2000; Brazdil et al. 2003), or predicting the performance of algorithms individually (Gama and Brazdil 1995; Sohn 1999; Köpf et al. 2000; Bensusan and Kalousis 2001). More recently, discovering suitable data mining workflows have been investigated by using data mining ontologies (Hilario et al. 2011) and workflow templates (Kietz et al. 2010). Meta-learning has been rarely used for parameter prediction. The only work in this direction uses meta-learning to predict kernel parameters for Support Vector Machines (Soares et al. 2004; Soares and Brazdil 2006). A good survey on meta-learning is the paper by Smith-Miles (2008).

The approach presented in this paper uses meta-learning to predict good parameter values for a new dataset. However, instead of directly applying the predicted parameters, they serve as a base for a following optimization step. The random initialization of the genetic algorithm is replaced by using these predicted values as start points. Combining the predictions of meta-learning with a genetic optimization reduces the risks of poor solutions compared to applying both methods separately. If the meta-learning predicts poor parameter values, the following optimization step is able to correct them. On the other hand, if the genetic algorithm starts with an already promising solution, its risk of ending up far away from the optimum will be reduced as well. This is an important aspect because genetic algorithms do not provide any worst case guarantees as grid search does.

For the evaluation of the presented approach, we used two state-of-the-art classifiers:

Support Vector Machine (SVM): SVM (Cortes and Vapnik 1995) is a non-linear maximum margin classifier using a kernel function. A typical kernel function is the radial basis function that contains the parameter γ. The penalty parameter C of SVMs controls the trade-off between minimizing the number of wrongly labeled training samples and maximizing the margin. We considered optimization of these two parameters in this work. A more detailed description of the SVM classifier including its parameters can be found in Burges (1998). We choose the SVM classifier because it has been very successfully used for a wide range of datasets from different domains. Additionally, the training time is rather high compared to other classifiers. This makes it harder to optimize, because the evaluation of just one parameter setting can be already very time-consuming.

Random Forest (RF): RF (Breiman 2001) is an ensemble classifier combining random selection of features with bagging. Multiple decision trees are created using a random subset of the attributes. For classification, all trees are applied and make a prediction. The final class is determined by a voting scheme among them. RF is also a widely used classifier and contains several parameters. In this work, we optimized five parameters: number of trees, number of randomly chosen attributes, minimum number of instances for a split, minimum split ratio, and the number of categories for random category split search. The minimum split ratio defines how balanced a split must be in order to be accepted. The last parameter determines the maximal number of possible splits of an attribute for performing an exhaustive split search instead of a random split search.

The rest of this paper is structured as follows: first, we give a brief introduction to genetic algorithms in Sect. 2 and highlight their problems using SVM parameter optimization as an example. Section 3 describes the methodology of the new approach and Sect. 4 contains an extensive evaluation of it. The conclusion and future work make up the last section.

2 Genetic algorithms for parameter optimization

Genetic algorithms are a common search strategy and optimization method that are inspired by biological evolution. A population of individuals evolves over multiple generations. The individuals with higher fitness values survive and new individuals are generated by crossing-over and/or mutating existing individuals in the population. This leads to an increase in the average fitness of the population over multiple generations.

In optimization, each individual of the population is a candidate solution of the problem. The fitness of the individual is the quality of this solution. The goal is to find a solution with the highest fitness value. The algorithm works in principle as follows:

- Creation of a random start population of r individuals.
- Evaluation of the fitness of all individuals in the population.
- Random selection of individuals for next generation according to their fitness. Some individuals may be doubled whereas individuals with a poor solution will be discarded.
- Probabilistic mutation of single individuals.
- Probabilistic cross-over by recombining two individuals to create two new individuals.
- Stop if the stop criterion is fulfilled. Otherwise, continue with step 2.

As stopping criterion, different rules can be used. For instance, the search can be stopped if a maximum number of generations were reached or if no better solution was found during a defined number of consecutive generations. The output of the algorithm is usually the best solution found during the search.

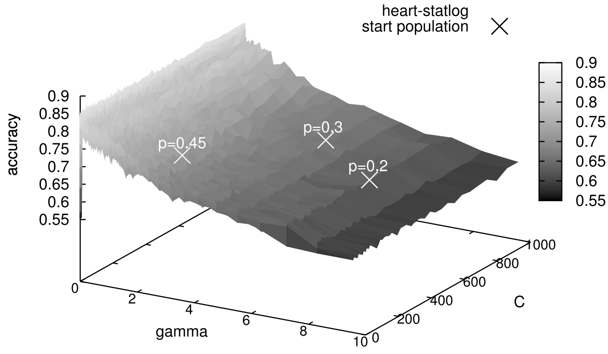
For parameter optimization of a classifier, a solution is one specific setting of all parameters of the classifier that should be optimized. The fitness of the individual is the performance achieved by the classifier using these parameter values. This means, whenever a solution has to be evaluated, the concerned classifier has to be trained again. For most classifiers, this is the most time-consuming part in contrast to the actual classification. If cross-validation is used, even multiple trainings of the classifier are needed for determining the quality of one parameter setting.

Because genetic algorithms select solutions with higher quality, the search is guided to regions of the search space with more promising solutions. Although the search is not guided directly to a local optimum and may not find any better solution over generations, it converges to a local optimum in sufficient time with a high probability.

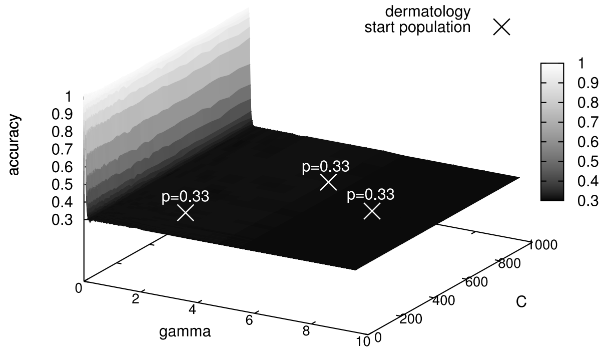
A problem occurs when big parts of the search space consist only of solutions with the same fitness value. In this case, it is likely that all individuals of a random start population have equal fitness values. This is problematic for selecting individuals for the next generation probabilistically according to their fitness values. If all solutions have the same fitness value, they also have the same selection probability. The search cannot be guided to regions with better solutions and randomly moves within the region with equal quality solutions instead. This increases the number of generations needed to find the best solution significantly.

In the worst case, it can miss out the optimum completely and end up far away from it. One reason for that is a stopping criterion that is often used: the search is stopped if several consecutive generations do not lead to any improvement. This scenario is illustrated in Fig. 1 using the datasets *heart-statlog* and *dermatology* from the UCI machine learning repository (Asuncion and Newman 2007). The plot shows the SVM accuracy values for different combinations of the parameters γ and C. In Fig. 1(a), the three individuals have different selection probabilities due to their different fitness values. The better individuals will be selected more probably and the search will be continued in these more promising regions.

Fig. 1 On target functions with big flat areas, the probabilistic selection step of genetic algorithms will not guide the search to areas of high quality solutions



(a) For the *heart-statlog* dataset, the selection step will select the better solutions with higher probability.



(b) On the *dermatology* dataset, all individuals have the same quality and therefore the same selection probability as well.

In contrast to this, Fig. 1(b) shows the common case where most of the candidate solutions deliver the same performance. This value is often the default accuracy (like by guessing or picking the most frequent class) that many parameter settings achieve. In addition, there are only a fixed number of possible accuracy values, especially for a small amount of test data. Different decision boundaries have the same classification accuracy. This leads to individuals with equal selection probabilities and the genetic algorithm is not able to select individuals that are closer to the optimum with higher probability. These issues are worsened due to the time constraint: the number of required parameter evaluations should be as small as possible. This can be obtained by using a rather small population and a rigorous stopping criterion, but both actions intensify the previously described issues. Small populations increase the probability of all individuals having the same selection probability. A more rigorous stopping criterion increases the chance of the search being stopped in flat regions of the target function as well.

If the difference of the fitness values of individuals in the flat regions is very small but not zero, the choice of the selection scheme might be important. In contrast to the common Roulette Wheel selection where the selection probability of an individual directly correlates with its fitness, the selection probability using rank selection is only based on the rank of the individual. For solutions with small differences in their fitness, the better solution is selected more probably compared to Roulette Wheel selection. Therefore, genetic algorithms using rank-selection might be able to overcome the problem of flat regions. The choice of the selection scheme is investigated later in the evaluation section.

3 Methodology

For solving the mentioned limitations we propose to use specific start points for the genetic optimization of parameter values. The typically random initial population will be replaced by defined start points. These points should be already reasonably close to the optimal parameter values. This has two advantages: (1) The time needed for finding the best solution is reduced. (2) The risk of ending up far away from the optimal value is decreased. The presented approach utilizes experience about optimal parameter values of several datasets. Additionally, a similarity measure of datasets is used to apply this knowledge to new datasets. The start population of the genetic optimization with a population of size r is defined as the optimal parameter values of the r most similar datasets.

3.1 Meta-features

The similarity measure of datasets uses features that describe a dataset. Such features are often called meta-features. Different features and feature types were proposed in the literature. We selected a subset of them, which were computable on all datasets we used in our evaluation. The following 15 meta-features, that can be grouped according to their theoretical foundation, were used:

1. Simple meta-features (Engels and Theusinger 1998):
   - Number of samples
   - Number of attributes
   - Number of numerical attributes
   - Number of categorical attributes
2. Statistical meta-features (Gama and Brazdil 1995; Vilalta et al. 2004; Brazdil et al. 1994; King et al. 1995):
   - cancor1: canonical correlation for the best single combination of features
   - cancor2: canonical correlation for the best single combination of features orthogonal to cancor1
   - fract1: first normalized eigenvalues of canonical discriminant matrix
   - fract2: second normalized eigenvalues of canonical discriminant matrix
   - skewness: mean asymmetry of the probability distributions of the features (non-normality)
   - kurtosis: mean peakedness of the probability distributions of the features
3. Landmarking meta-features (Pfahringer et al. 2000; Bensusan and Giraud-Carrier 2000b):
   - Best stump node landmark
   - Average stump node landmark
   - Worst stump node landmark
   - Naive Bayes landmark

The *fract* features measure the collinearity of the class means. The landmarking approach utilizes simple, fast computable classifiers and uses the achieved classification accuracy as a feature of the dataset. The stump node landmarkers create single decision nodes for each attribute and the nodes with the best, the average, and the worst accuracy are used as landmarkers.

Soares and Brazdil (2006) used meta-features specifically designed for SVMs to predict its parameters. Since our approach is not limited to a specific classifier, more general features are used in this paper. However, the underlying assumption that similar datasets require similar parameter values of the classifier is the same. Therefore, the features construct the base for a feature selection step, which also uses knowledge based on experience. The knowledge and its structure is described in the following section. The feature selection step will be explained in Sect. 3.3.

3.2 Knowledge base

The knowledge base K is the gathered experience of previous optimizations of the classifier on n unique datasets. It consists of their meta-features $m_1, \dots, m_n$ and the optimal parameter values of the classifier found for each dataset $\hat{\theta}_1, \dots, \hat{\theta}_n$:

$$K = \left\{ (m_1, \hat{\theta}_1), \dots, (m_n, \hat{\theta}_n) \right\}. \tag{1}$$

Additionally, since the presented algorithm calculates the meta-features m_i of a dataset and returns its optimized parameter values $\hat{\theta}_i$, this information can be added to the knowledge base $K_{\text{new}} = K \cup \{(m_i, \hat{\theta}_i)\}$. This increases the amount of experience and should improve further runs. A bootstrapping approach to create experience without any initial knowledge $K = \emptyset$ is also conceivable. The very first evaluations are done with random start populations. If some experience was gathered, this can already be used for defining a start population. The first evaluations may be repeated using the newly gathered knowledge to increase the achieved accuracy and to improve the start points for other datasets. For practical usage it is also possible to use a common knowledge base.

3.3 Meta-feature selection

To find the most promising start points for the genetic algorithm, we want to find r datasets within the knowledge base that are most similar to the new dataset. This step is motivated by case-based reasoning (CBR) research, which can be explained in one simple sentence (also known as the CBR assumption): Similar problems have similar solutions. Hence, we use a simple nearest neighbor approach to find r nearest datasets based on meta-features. However, the number of meta-features is relatively large considering that the number of samples (each dataset is one sample in this scenario) is very small. This problem is aggravated by the fact that the nearest neighbor algorithm is sensitive to irrelevant attributes.

One way to solve this problem is to manually select *a priori* a small subset of meta-features that provides information about properties that affect algorithm performance as done in Brazdil et al. (2003). However, it has been shown in meta-learning literature that the set of meta-features that is suitable for predicting performance of one learning algorithm may not be suitable for another learning algorithm (Kalousis and Hilario 2001). Therefore, an automatic meta-feature selection step is desired to avoid manual investigation of the meta-features for new target classifiers and knowledge bases.

For automatic feature selection, we use a brute-force search with a wrapper approach where each possible feature combination is tested. In a standard wrapper approach, the actual target function is evaluated for each considered feature set. In our case, where the target function is the achieved accuracy of the classifier, evaluating the target function would require to run the complete genetic optimization. This means training the classifier for each candidate parameter combination. Furthermore, since the genetic algorithm is a random-based method, multiple runs should be performed to get reliable results. This leads to a huge computational effort for the evaluation of a meta-feature set. Therefore, we modified the traditional wrapper approach to only estimate the actual performance of a meta-feature set using the knowledge base.

The knowledge base consists of both the meta-features and the optimal parameter values of the target classifier for each dataset. Hence, we can obtain two types of distances from the knowledge base: distances in the feature space (i.e. distances between meta-features of the datasets), and distances in the parameter space (i.e. distances between optimized parameter values on the datasets). For performing feature selection, we use the distance in the parameter space as the target function. We choose a subset of meta-features that minimizes distances in the parameter space. The reason for this is the assumption that the genetic algorithm will perform better if its start population is closer to the optimum.

More specifically, consider the set H contains all feature selection functions to select every possible subset of meta-features, $h(m)$ is the set of meta-features selected by a particular function h. For each dataset x of all datasets X in our knowledge base, let $\hat{\theta}_x$ be the optimal parameter values for x. We find the k-nearest-neighbors N_x in the feature space of the selected meta-features $h(m)$ and compute the corresponding distances in parameter space. The sum of these distances for all datasets is used as the target function for feature selection.

$$d_h = \sum_{x \in X} \sum_{i \in N_x} \|\hat{\theta}_i - \hat{\theta}_x\| \tag{2}$$

The feature selection function $\hat{h}$ that minimizes the distance d_h is chosen as the best candidate. The feature selection approach is also shown as pseudo code in Algorithm 1.

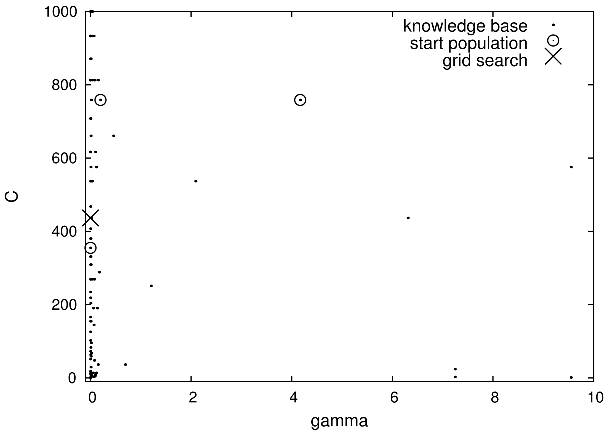
Algorithm 1 Meta-Feature Selection

function META_FEATURE_SELECTION(X, H, k)
  for all feature selection functions $h \in H$ **do**
    $d_h \leftarrow 0$
    for all datasets $x \in X$ **do**
      $N \leftarrow k$-nearest datasets of x in the space of the meta-features selected by h
      $d_\theta \leftarrow$ sum of distances between optimal parameter values of x and N
      $d_h \leftarrow d_h + d_\theta$
  return h that minimizes d_h

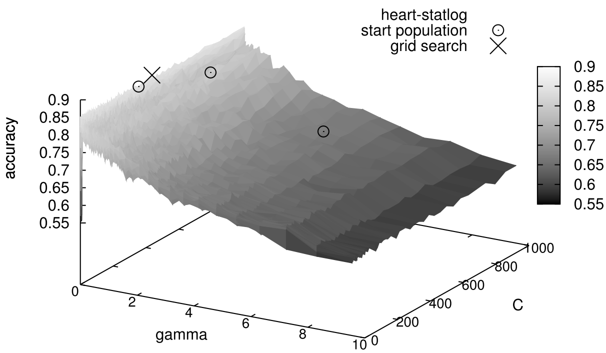
Note that, using 15 meta-features leads to $|H| = 32,767$ subsets in our brute force approach. Since only distances are calculated for a subset evaluation, the computational effort is not very large. In our experiments using 102 datasets and a single-threaded JAVA program, the average run-time of the feature selection method was only about 46 seconds on a 2.3 GHz AMD Opteron running Linux. Since the new dataset is not involved in the feature selection algorithm, the meta-feature selection can be performed off-line, independently from the actual optimization step.

The actual determination of the start population for a new dataset y that is not in the knowledge base uses the determined feature selection function $\hat{h}$. It gets the r nearest datasets according to the selected meta-features and uses the optimal parameter settings of these datasets as start points.

Fig. 2 Not all three selected start points for optimizing the SVM are very close to the result of the grid search for the *heart-statlog* dataset, but just one close point can already speed up the optimization



(a) The knowledge base, three selected start points and the result of a grid search for the *heart-statlog* dataset.

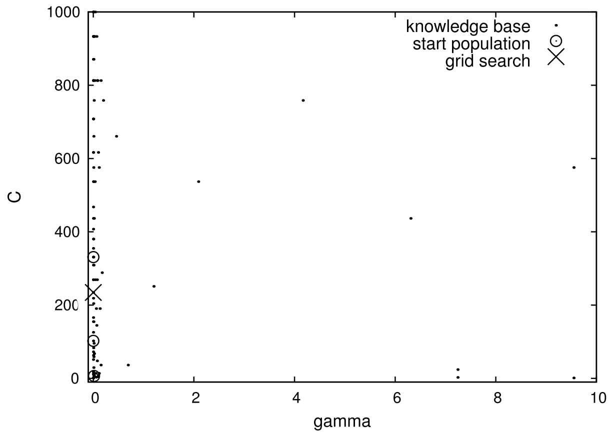


(b) One of the selected start points is already close to the result of the grid search and will therefore deliver a good performance value.

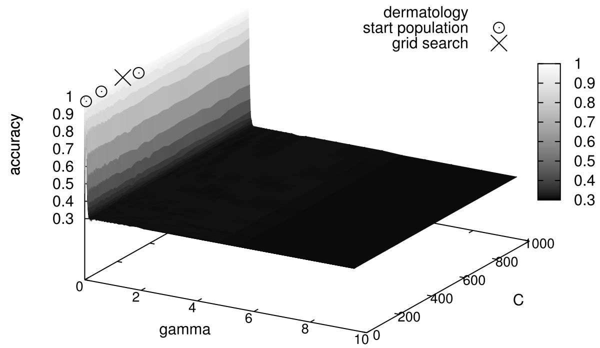
3.4 Examples

As examples for the application of the method, we use the SVM classifier and the same datasets from Fig. 1. Figure 2 illustrates the advantages of the approach for the *heart-statlog* dataset. The optimal parameter settings of the remaining datasets serve as knowledge base and are drawn as dots in Fig. 2(a). The parameter values of the datasets that are most similar to the *heart-statlog* dataset are surrounded by an additional circle. They will be used as start population for the optimization. The desired optimal parameter setting of the *heart-statlog* dataset is marked as a cross. One can see that not all selected datasets deliver very close parameter values, but just one start point near the actual optimum can decrease the optimization time. In Fig. 2(b), the classification accuracy over the parameter space is plotted for the same dataset. The selected start population is shown as circles and the result of a grid search as cross again. Since one selected start point delivers already a good performance value, the result will be quite good even if the search does not find any better solution. The

Fig. 3 If the optimization of the SVM starts with good solutions, the search will not get stuck in flat regions



(a) The knowledge base, three selected start points and the result of a grid search for the *dermatology* dataset.



(b) The complete start population has good accuracy values.

second example uses the *dermatology* dataset and is shown in Fig. 3. Again, not the actual closest start points were selected as shown in Fig. 3(a), but in Fig. 3(b) it is visible that all three points have good performance values, close to the grid search result. The search will not get stuck within the flat, black region of poor solutions.

4 Evaluation

For the evaluation of the presented approach, we used 102 classification datasets from the UCI machine learning repository (Asuncion and Newman 2007) and from StatLib (Vlachos 1998), but artificially created datasets might be used as well (Frasch et al. 2011). The used datasets from different domains contain from 10 to 2310 samples with 1 to 261 nominal and numerical attributes. A list of the datasets can be found in Table 3 in the Appendix. Since the training of the Random Forest classifier on the "cylinder-bands" dataset takes extremely long, we omitted this dataset for evaluations regarding the Random Forest classifier.

After the calculation of the meta-features, all datasets were preprocessed to meet the requirements of both classifiers. Missing values were replaced by the average value, nominal features were converted to numerical features, and finally, all features were normalized to the range $[-1; 1]$. The optimal parameter settings for every dataset were determined by a grid search and used to construct the knowledge bases for the evaluation. Regarding the SVM classifier, the search interval for γ was $[0.0001; 10]$ with 85 logarithmic steps and $[0; 1000]$ with 100 logarithmic steps for C. Due to the computational complexity, the number of steps for the parameters of the Random Forest classifier were smaller. We used five steps for the number of trees and six steps for the remaining four parameters. This leads to 6480 classifier evaluations during the grid search, which is already a huge computational effort.

The experiments were performed by utilizing only open source libraries. LibSVM (Chang and Lin 2001) served as SVM implementation and PARF (Skala 2004) as implementation for the Random Forest classifier. GAlib (Wall 1996) was used for the genetic algorithm. We modified this genetic algorithm implementation to use a defined start population. The mutation operation was a Gaussian mutation with a step size of 1.0. Its probability was selected by applying the rule of Bäck (1993): $p_m = \frac{1}{j}$, where j is the number of variables of an individual. This leads to $p_m = 0.5$ for SVM and $p_m = 0.2$ for Random Forest. Additionally, uniform cross-over was performed with a probability of $p_{co} = 0.9$.

We compared the presented approach with five different optimization approaches:

grid search: For the SVM classifier, we applied a typical 15×15 grid used by Hsu and Lin (2002): $\gamma = [2^{-10}, 2^{-9}, \ldots, 2^{4}]$ and $C = [2^{-2}, 2^{-1}, \ldots, 2^{12}]$. For the Random Forest classifier, the same grid with 6480 parameter combinations for creating the knowledge base is used.

std. GA: The standard genetic algorithm uses a random start population. It also utilizes GAlib as implementation.

CMA: For the Covariance Matrix Adaptation Evolution Strategy (Hansen and Ostermeier 2001) we used the implementation of the BEAGLE framework (Gagné and Parizeau 2006).

DIRECT: We used the NLopt (Johnson 2009) framework as implementation for the Dividing Rectangles algorithm (Jones et al. 1993).

GSS: Generating Set Search (Kolda et al. 2003) is a type of pattern search that can also handle linear constraints. HOPSPACK (Plantenga 2009) was the used implementation.

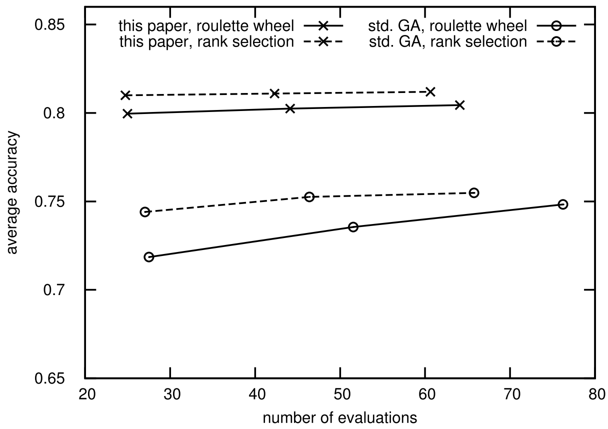
We applied all optimization methods on each of the 102 datasets using a leave-one-out cross-validation. The optimization intervals for the parameters of the SVM and Random Forest were the same as used for creating the knowledge base. Every classifier evaluation used a ten-fold cross-validation. Since the optimization runs contain random factors, 30 independent runs of each algorithm were performed on every dataset. The mean accuracy of an optimization method is the average over these 30 runs and all datasets.

All optimization methods were stopped after s generations/iterations without improvement. Since this parameter has a strong influence on performance and the run-time of the algorithm, its choice is important. The influence of this parameter is discussed in more detail in Sect. 4.2.

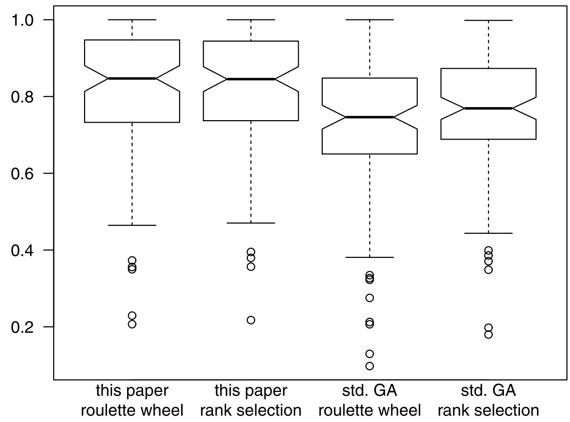
The population-based methods contain the population size r as an additional parameter. Since the main goal of the presented approach is obtaining good solutions in a short time, small populations and rigorous stopping criteria are primarily considered. Thereby, we assure that the run-time of the genetic optimization is shorter than a grid search, even though the standard GA typically requires a bigger population. The presented approach additionally contains the parameter k within its meta-feature selection step (see Sect. 3.3).

An additional parameter for genetic algorithms is the used selection scheme. As already mentioned, a typical selection scheme is Roulette Wheel selection but rank selection may

Fig. 4 Comparison of the presented method and the standard GA optimizing the parameters of an SVM using different selection schemes and a population size $r = 7$: both approaches achieve higher accuracy values when using rank selection instead of roulette wheel selection. It is noticeable that the improvement is bigger for the standard GA



(a) Average accuracy over 102 datasets using different stopping criteria.



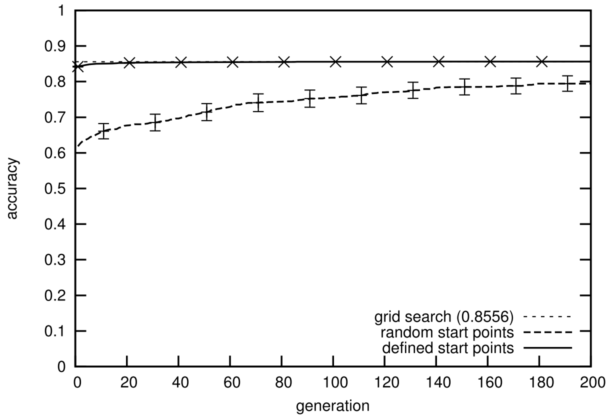
(b) Box plot over the datasets using a stopping criterion of $s = 2$.

work better for parameter optimization. Therefore, we investigated both selection schemes first. Figure 4(a) shows the average accuracy of the SVM classifier achieved by the standard GA and the presented approach using different stopping values and a population size of 7. Figure 4(b) shows a box plot of the two approaches for a stopping criterion of $s = 2$. It is visible that the rank-based selection scheme achieves better results for both methods. Therefore, we decided to use this selection scheme for further evaluations of the genetic algorithms. It is also visible that the improvement by using rank selection is higher for the standard GA.

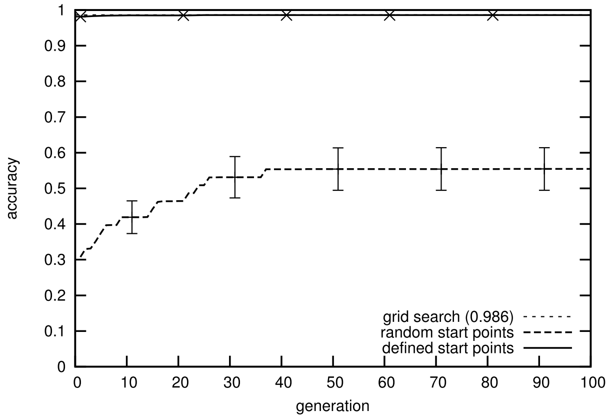
4.1 Results and discussion

For the two datasets from the examples of Sect. 3.4, Fig. 5 shows the achieved accuracy values during the search for the optimal parameter values of the SVM by the standard GA and the presented approach. The values are the mean and the standard error over 30 runs.

Fig. 5 The achieved classification accuracy during the search is plotted for two datasets and a fixed number of generations ($k = 1$, $r = 5$, average and standard error over 30 runs)

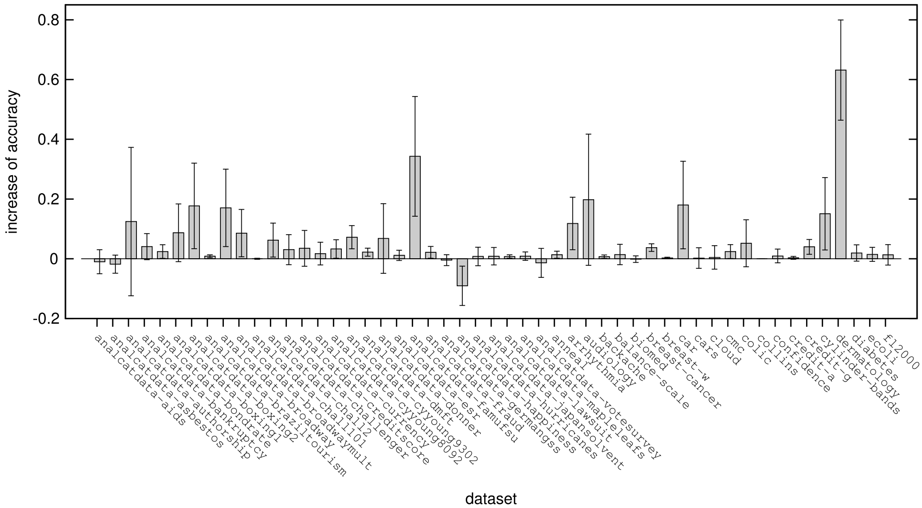


(a) *heart-statlog* dataset: the presented approach already starts with very good solutions and has very low variances. The standard GA needs much more time for finding good solutions and has high standard error.

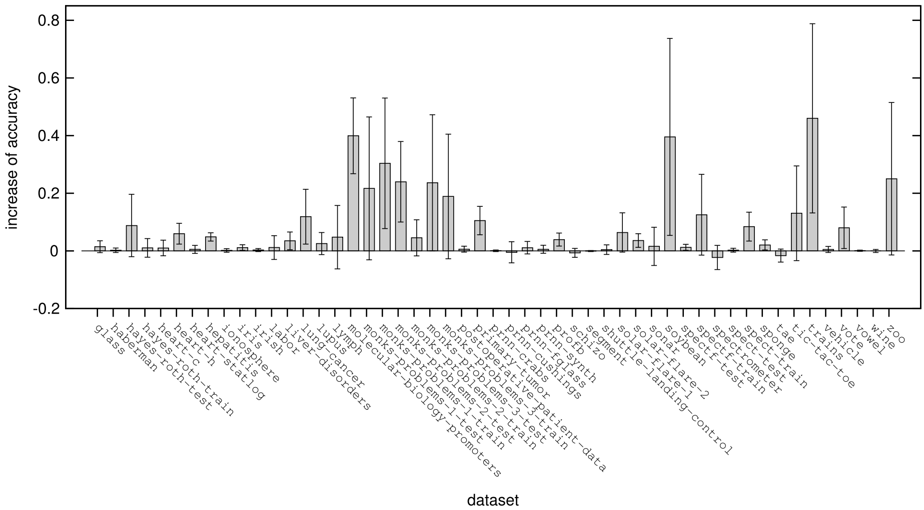


(b) *dermatology* dataset: the presented approach starts with very good solutions again. The risk of getting a bad solution is rather high for the standard approach.

For this illustration, we used a different stopping criterion. The search was stopped after a fixed number of generations and it uses a population size of 5. In both graphs, one can see that the presented approach already starts with good performance values and converges to the accuracy of the grid search in few generations. For the heart-statlog dataset in Fig. 5(a), the standard genetic algorithm also delivers good results, but needs a much higher number of generations. On the contrary, the results of the algorithm with random start points for the *dermatology* dataset in Fig. 5(b) stay below the optimum even after a large number of generations. Additionally, the standard error of the achieved accuracy values over 30 runs is very low using the presented method. This indicates a very low risk of ending up with a bad solution. In contrast to this, the risk of the standard GA is higher as especially noticeable in Fig. 5(b).



(a) Results for the datasets from `a` to `f`.



(b) Results for the datasets from `g` to `z`.

Fig. 6 The increase in classification accuracy of an SVM for each single dataset of the proposed method compared to the standard GA (average of 30 runs including the standard deviation) with the parameters $k = 1$, $r = 5$, and $s = 2$

Figure 6 shows the increase of accuracy for every dataset after the random start population was replaced by a defined population. The standard deviation of this increase is also visible. The plot uses the parameter values $r = 5$, $k = 1$, and $s = 2$. Although the increase in accuracy is not very large for every dataset, the presented method is able to improve results for almost every dataset.

Table 1 shows the average accuracy values of the presented approach and the two population-based methods for different population sizes and stopping criteria. The two other methods, DIRECT and GSS, are omitted in this table since they are not population-based

Table 1 Average classification accuracy of the grid search, the standard genetic algorithm, the CMA algorithm and the proposed method for different parameter combinations. All values are in [%]

Parameter		SVM accuracy				Random forest accuracy			
r	s	Grid search	Std. GA	CMA	This paper	Grid search	Std. GA	CMA	This paper
3	2	81.1	70.4	69.0	79.5	80.7	68.4	68.2	77.8
5	2	81.1	72.9	69.3	80.3	80.7	71.8	68.7	78.7
5	6	81.1	74.2	70.4	80.9	80.7	73.0	69.7	79.2
7	2	81.1	74.4	69.9	81.0	80.7	73.7	69.0	79.2
7	6	81.1	75.5	70.8	81.2	80.7	74.7	69.7	79.7
7	10	81.1	76.4	71.0	81.8	80.7	74.9	70.1	80.0

and a comparison considering population sizes is not easily possible. The accuracy achieved by the grid search is also shown. It is visible that the presented method achieves higher accuracy values than the standard GA and CMA for both classification algorithms. For $r = 7$ and $s = 10$, the presented method achieves an even higher accuracy than grid search for the SVM classifier.

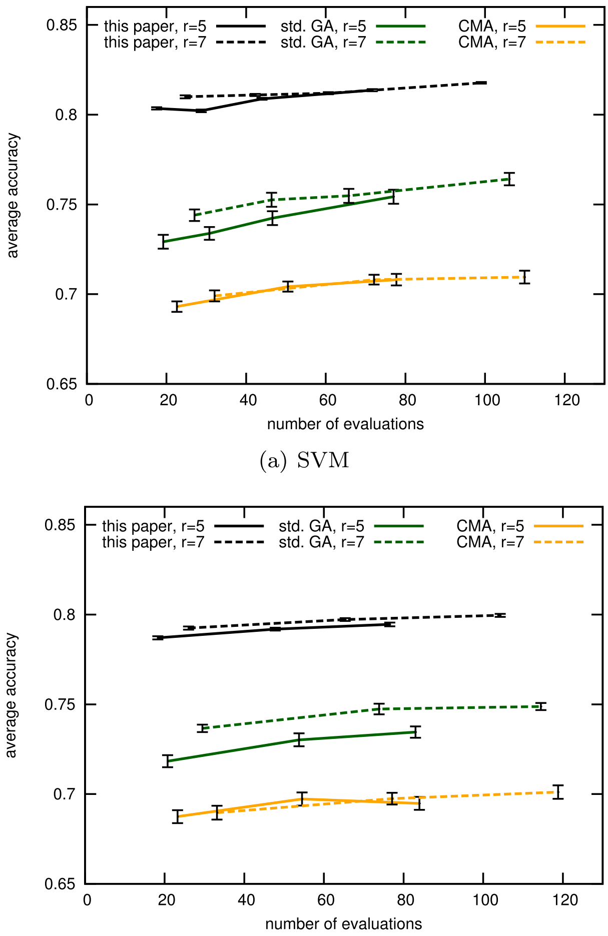
To decide whether these differences in accuracy are statistically significant or not, we applied statistical tests over the 102 datasets and compared the presented approach with each competing method. To do this, the paired t-test might be used. However, this parametric test assumes that the accuracy differences are normally distributed. Furthermore, outliers in the tested population decreases the power of the t-test (Demšar 2006). Since the results particularly contain outliers, we used the non-parametric Wilcoxon test instead. For all considered parameter combinations and a confidence level of 95 %, the differences between the proposed method and the grid search are not statistically significant whereas the differences between the proposed method and the std. GA as well as the CMA are indeed significant.

In addition to the accuracy, we compared the run-time of the different approaches. Figure 7 shows the accuracy depending on the number of evaluations for two different population sizes. The presented approach achieves higher accuracies for both classifiers for a different number of classifier evaluations. The standard GA and the CMA method are unable to achieve the same accuracy even for higher number of evaluations. Surprisingly, the CMA algorithm is worse than the standard GA for approximately the same number of evaluations.

Summarizing, the presented method is able to reach performance values of a grid search, but only needs the time of a genetic algorithm. For example, using an AMD Opteron with 2.3 GHz, the presented approach needs only about 45 minutes for optimizing the SVM parameters of all datasets using the parameter values $r = 7$ and $s = 2$. The grid search needs about six hours that is consequently more than seven times longer. However, the difference of the achieved average accuracy values of 81.0 % by the presented method and 81.1 % by the grid search is statistically not significant.

Since the two optimization methods DIRECT and GSS do not use any population, the classifier is evaluated only once during an iteration of the search. Therefore, a large number of iterations without improvement were allowed to assure a fairer comparison. However, as visible in Fig. 8, both methods do not reach the accuracy values of the presented approach for a comparable number of classifier evaluations. Although the GSS algorithm delivers good results for the SVM classifier, the achieved accuracy values are lower than the std. GA for the Random Forest classifier.

Fig. 7 The average accuracy (including the 95 % confidence interval) achieved by the presented approach and the competing population-based methods depending on the number of classifier evaluations

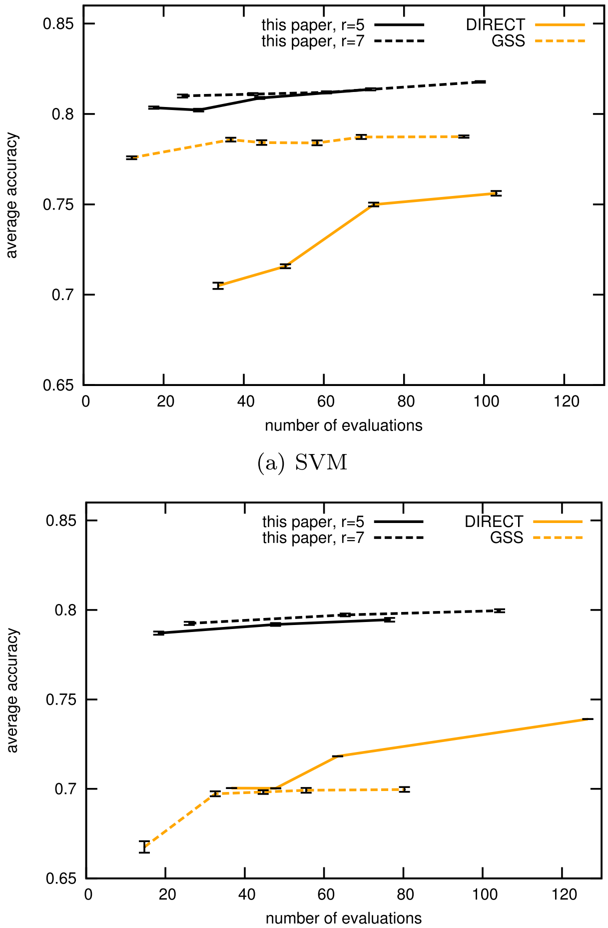


(a) SVM

(b) Random Forest

For the Wilcoxon tests of the accuracy differences over the datasets, we used those of the considered parameter combinations of the algorithms on which they achieved their highest average accuracy. With a confidence level of 95 %, the differences between the presented method and GSS as well as DIRECT are statistically significant. Only the difference for SVM with GSS is statistically not significant.

Furthermore, we investigated how much accuracy could be achieved purely by meta-learning without any evolutionary optimization. For that purpose, we evaluated the performance of the parameter predictions directly. Predicted parameter settings are the optimal parameter values of the r most similar datasets. The performance of the predictions is the highest accuracy achieved with one of those unmodified settings. Table 2 shows the performance of the predictions for the SVM classifier. Besides, the predicted parameter values were used as start points for the genetic algorithm to quantify the improvement in accuracy due to the optimization step. The results show that the following evolutionary optimization is able to improve the performance values. The last column shows the percentage of



(a) SVM

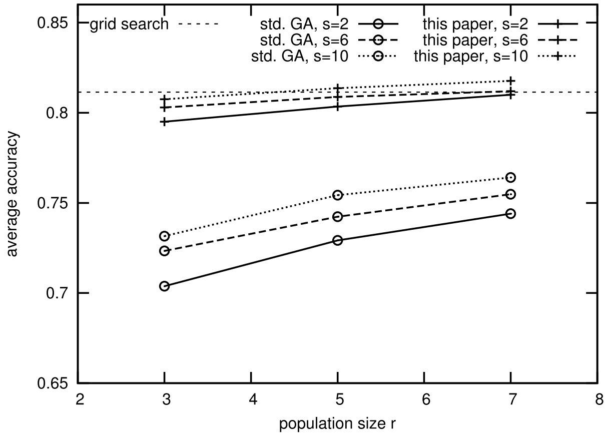
(b) Random Forest

Fig. 8 The average accuracy (including the 95 % confidence interval) achieved by the presented approach, the DIRECT, and the GSS optimization algorithms depending on the number of classifier evaluations

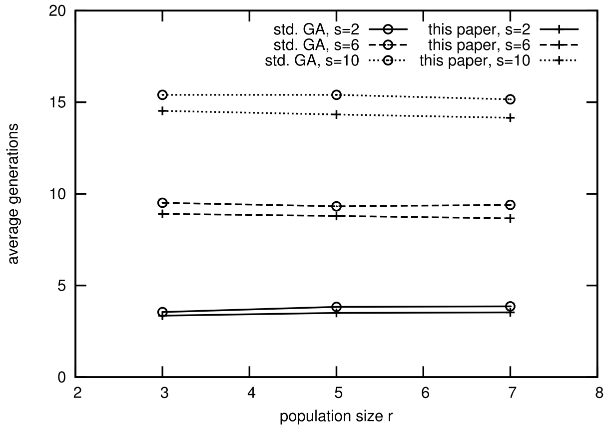
Table 2 Accuracy values of meta-learning and the genetic algorithm with defined start points for SVM. The last column shows the percentage of the datasets, for which accuracy was increased significantly. The parameter r denotes the number of predicted parameter settings by meta-learning and population size in the genetic algorithm. The genetic algorithm was stopped after six generations without improvement ($s = 6$). All values are in [%]

r	Accuracy of metalearning prediction	Accuracy of presented approach	Significantly improved
1	74.5	78.1	73.3
3	77.9	80.3	65.3
5	78.6	80.9	68.3

the datasets, for which accuracy was increased significantly. A Wilcoxon test with a confidence interval of 95 % was used. Whereas the prediction of meta-learning delivers already good results, the accuracy could be increased significantly for more than 65 % of the datasets.



(a) The average accuracy achieved by both methods.



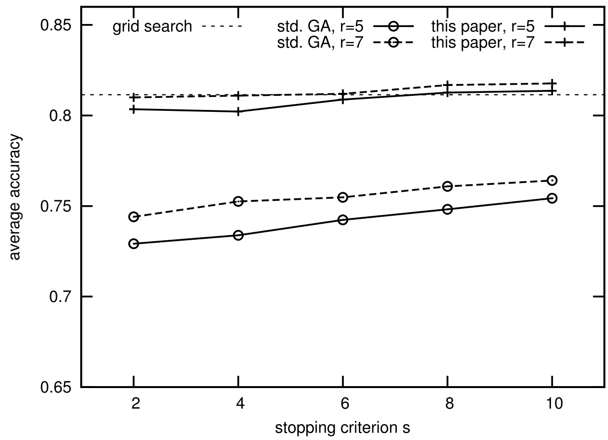
(b) Average number of generations for both methods.

Fig. 9 The influence of the populations size r for three different values of s. Smaller populations sizes already reach good performance values

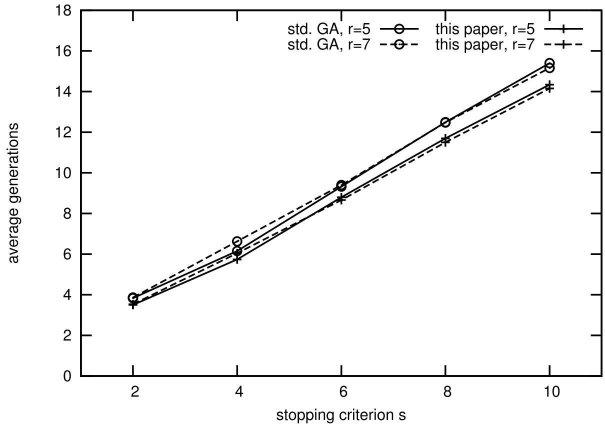
4.2 Effect of changes in parameter values

Previously, only exemplary values of the parameters k, r, and s have been investigated, although the choice of the parameters can have a strong influence on the results of an optimization method. Bartz-Beielstein et al. (2005) presented an approach for optimizing the parameters of search heuristics which can also be used to optimize different parameters of genetic algorithms. However, instead of such an additional optimization, we did an extensive manual analysis of the parameters of the presented method for the SVM classifier.

The parameter k is the number of neighbors, whose distance is minimized during the meta-feature selection step. Since only one good prediction is sufficient for improving the genetic algorithm, a value of one seems to be a good choice, independently from the other parameters. Therefore, we omitted the genetic algorithm and computed only the accuracy values achieved by the potential start points. Our experiments showed that for small values



(a) The average accuracy achieved by both methods.



(b) Average number of generations for both methods.

Fig. 10 The influence of the parameter s of the stopping criterion for two different population sizes. Rigorous stopping criteria (small values of s) only slightly reduce the average accuracy of the presented approach

of k, the results differ not much and $k = 1$ seems to be a good choice in general. So we stick to this value for the evaluation of the two other parameters.

The population size r is an important parameter of genetic algorithms. Smaller population sizes reduce the computation time, but may miss good results as well. The average accuracy values achieved by the SVM with different population sizes are plotted in Fig. 9(a) for both approaches. Although very small population sizes are uncommon for genetic algorithms, the presented approach already reaches good performance results using these settings. The performance of the standard approach stays below the presented method for each considered population size.

Figure 9(b) shows the average number of generations for both methods. It is visible that an increase in the population size does not lead to fewer generations in general. However, the average number of generations for the presented method were lower than those for the standard genetic algorithm.

For the parameter s of the stopping criterion, Fig. 10 shows the average accuracy and number of generations. A more rigorous stopping did not lead to much worse results of the presented method (Fig. 10(a)), but can shorten the search time dramatically (Fig. 10(b)). Again, parameter values that will lead to a shorter run-time (like small values of s) increase the benefit of the presented approach.

5 Conclusion

In this paper, a new approach of using meta-learning in genetic algorithms for parameter optimization of classifiers was presented. Knowledge based on previous evaluations on similar datasets was used to define a better initial population instead of random individuals. The similarity between datasets was computed as a function of distance between their automatically selected meta-features.

Our approach was used for optimizing two parameters of Support Vector Machines (SVM) and five parameters of the Random Forest classifier on over 100 datasets from the UCI and StatLib repositories. The results showed that the presented approach is able to significantly increase the average classification accuracy of both classifiers compared to a standard genetic algorithm. The achieved accuracy values of the SVM were close to those achieved by a typical 15×15 grid search for SVMs at one-seventh of the computational cost. Moreover, the presented approach was compared to three other state-of-the-art optimization algorithms. The results show that it achieves the best average accuracy for both classifiers.

Acknowledgements This work was partially funded by the BMBF (German Federal Ministry of Education and Research), project PaREn (01 IW 07001). We also thank the reviewers for their detailed and helpful comments to the manuscript.

Appendix: Increase of accuracy

Table 3 The increase of accuracy of the presented method compared to the standard GA for each dataset and different parameter combinations (averaged over 30 runs)

Dataset	SVM				Random forest			
	$r=5$		$r=7$		$r=5$		$r=7$	
	$s=2$	$s=6$	$s=2$	$s=6$	$s=2$	$s=6$	$s=2$	$s=6$
Analcatdata-aids	−1.8	−1.0	−0.6	−1.0	8.5	8.0	7.0	9.1
Analcatdata-asbestos	−2.9	−1.8	−2.4	−2.2	1.9	2.8	3.3	2.4
Analcatdata-authorship	10.4	12.5	10.4	4.2	0.9	0.8	0.8	0.9
Analcatdata-bankruptcy	3.9	4.1	4.5	4.3	21.1	20.5	15.9	8.9
Analcatdata-bondrate	2.1	2.4	2.4	1.6	1.0	0.8	0.9	1.0
Analcatdata-boxing1	10.3	8.7	9.6	8.3	9.5	9.2	7.6	5.8
Analcatdata-boxing2	17.5	17.7	11.9	10.3	12.0	6.9	11.2	2.8
Analcatdata-braziltourism	0.7	0.9	1.0	0.8	0.2	−0.1	0.0	0.3
Analcatdata-broadway	12.6	17.0	11.5	9.1	8.1	4.5	8.4	4.7
Analcatdata-broadwaymult	10.1	8.6	9.0	6.6	−1.4	−1.0	−1.4	−0.1
Analcatdata-chall101	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Analcatdata-chall2	6.2	6.2	7.1	3.0	4.4	1.3	4.1	3.7
Analcatdata-challenger	5.1	3.0	1.7	0.4	7.8	7.0	6.8	7.2
Analcatdata-creditscore	3.7	3.5	0.2	0.1	4.9	6.1	1.6	1.8

Table 3 (*Continued*)

Dataset	SVM				Random forest			
	$r=5$		$r=7$		$r=5$		$r=7$	
	$s=2$	$s=6$	$s=2$	$s=6$	$s=2$	$s=6$	$s=2$	$s=6$
Analcatdata-currency	−0.2	1.7	−0.6	−0.5	1.9	1.7	1.7	2.8
Analcatdata-cyyoung8092	4.7	3.3	4.9	5.4	2.8	3.4	4.0	3.3
Analcatdata-cyyoung9302	6.3	7.2	6.8	5.8	4.0	3.5	3.0	3.0
Analcatdata-dmft	2.4	2.2	2.0	2.4	1.0	0.2	0.3	0.1
Analcatdata-donner	6.7	6.8	17.0	13.0	12.2	10.3	9.1	10.0
Analcatdata-esr	0.9	1.1	0.9	1.7	1.0	1.1	1.9	2.6
Analcatdata-famufsu	36.4	34.3	41.4	33.1	47.2	47.3	38.7	41.5
Analcatdata-fraud	2.2	2.1	3.3	2.5	4.9	3.8	5.1	3.5
Analcatdata-germangss	0.0	−0.5	−0.8	0.2	2.0	1.2	1.5	0.8
Analcatdata-happiness	−10.1	−9.1	−1.4	−1.4	12.8	13.9	13.4	12.2
Analcatdata-hurricanes	1.2	0.8	0.8	0.8	−1.3	−1.8	0.2	0.2
Analcatdata-japansolvent	0.3	0.8	1.7	1.2	17.3	14.9	7.1	10.7
Analcatdata-lawsuit	0.8	0.7	1.1	0.8	0.8	1.1	0.8	1.2
Analcatdata-mapleleafs	0.3	0.9	0.3	0.4	0.1	0.1	−0.1	0.0
Analcatdata-votesurvey	−4.6	−1.4	−3.8	−3.1	8.6	10.0	10.3	4.1
Anneal	0.8	1.4	1.3	1.3	2.1	2.3	6.1	4.0
Arrhythmia	14.0	11.8	10.3	9.3	4.4	3.1	3.5	2.0
Audiology	19.6	19.7	28.2	24.3	19.7	17.4	15.9	15.6
Backache	0.5	0.7	0.6	0.9	0.5	0.7	0.4	0.9
Balance-scale	3.4	1.4	2.5	2.1	2.6	1.5	1.6	0.8
Biomed	−0.1	−0.1	−0.3	−0.2	2.2	1.6	1.9	1.4
Breast-cancer	4.4	3.7	4.1	3.6	−1.1	−1.3	−0.3	−1.5
Breast-w	0.4	0.3	0.4	0.3	0.4	0.2	0.3	0.3
Car	15.7	18.0	10.9	7.1	9.6	6.8	6.0	4.5
Cars	0.4	0.2	1.6	2.3	6.5	8.1	6.6	5.8
Cloud	−0.1	0.4	−0.4	0.3	9.7	9.5	8.0	6.6
Cmc	2.8	2.4	2.3	1.4	0.1	−1.0	−0.7	−0.2
Colic	9.1	5.2	4.7	5.3	0.7	0.3	0.4	0.3
Collins	0.0	0.0	0.2	0.0	5.6	4.6	4.1	4.3
Confidence	0.7	0.9	0.8	1.1	38.9	37.9	23.1	25.7
Credit-a	1.3	0.3	0.4	0.5	0.5	0.7	0.4	0.3
Credit-g	4.7	4.0	4.2	3.9	2.0	1.2	1.1	1.1
Cylinder-bands	14.5	15.1	12.6	11.1	–	–	–	–
Dermatology	49.7	63.2	49.7	56.5	6.1	3.7	2.9	2.0
Diabetes	3.0	1.9	1.7	1.4	0.6	0.1	0.2	−0.1
Ecoli	2.2	1.4	1.7	1.2	1.8	2.2	4.4	3.0
Fl2000	1.9	1.3	1.3	1.3	7.3	10.0	5.9	5.5
Glass	1.7	1.5	0.5	0.9	7.5	8.5	6.9	7.2
Haberman	−0.2	0.2	0.1	−0.1	0.2	0.4	0.2	0.1
Hayes-roth-test	8.1	8.8	6.0	7.3	23.8	15.4	15.1	19.2
Hayes-roth-train	1.2	1.0	1.2	1.2	7.7	5.1	6.0	7.3
Heart-c	3.1	1.0	2.2	0.2	0.4	−0.1	0.2	0.2
Heart-h	8.1	6.0	7.1	6.1	0.0	0.2	0.0	0.6
Heart-statlog	2.3	0.5	0.6	0.2	0.5	0.8	0.5	0.3

Table 3 (*Continued*)

Dataset	SVM				Random forest			
	$r=5$		$r=7$		$r=5$		$r=7$	
	$s=2$	$s=6$	$s=2$	$s=6$	$s=2$	$s=6$	$s=2$	$s=6$
Hepatitis	5.0	4.9	4.9	4.8	1.9	1.1	2.5	2.3
Ionosphere	1.2	0.2	0.7	0.2	3.3	1.4	1.6	1.3
Iris	1.4	1.2	1.3	1.4	5.8	5.8	5.0	0.1
Irish	0.4	0.3	0.3	0.2	1.9	0.1	0.4	0.0
Labor	8.4	1.2	7.4	1.7	7.9	9.9	13.7	5.3
Liver-disorders	3.6	3.5	2.7	2.6	0.7	0.6	1.7	1.1
Lung-cancer	13.5	11.9	11.9	14.6	12.8	13.3	13.6	11.9
Lupus	3.4	2.5	1.8	1.3	1.1	3.2	1.1	0.5
Lymph	12.7	4.8	9.1	2.7	4.1	3.5	4.4	4.8
Molecular-biology-promoters	35.8	39.9	32.9	27.5	4.5	6.4	2.3	2.5
Monks-problems-1-test	30.0	21.7	13.3	18.3	23.0	14.3	13.8	10.2
Monks-problems-1-train	37.7	30.4	30.6	23.3	20.1	16.9	14.9	16.8
Monks-problems-2-test	31.8	24.0	26.2	22.2	20.2	17.9	20.3	18.0
Monks-problems-2-train	5.7	4.5	9.7	10.6	3.8	4.6	4.4	5.7
Monks-problems-3-test	18.2	23.6	12.6	17.3	8.5	2.3	4.6	2.1
Monks-problems-3-train	30.5	18.9	24.6	17.3	11.4	4.9	5.0	2.3
Postoperative-patient-data	0.4	0.6	0.8	0.9	0.2	0.0	0.1	0.3
Primary-tumor	9.0	10.5	9.9	7.4	3.4	2.8	3.1	4.1
Prnn-crabs	0.4	0.1	0.2	0.1	14.2	8.8	8.2	7.2
Prnn-cushings	−0.1	−0.5	0.6	1.2	20.4	21.1	21.1	16.6
Prnn-fglass	1.4	1.1	0.8	0.5	10.6	8.7	11.7	8.5
Prnn-synth	0.3	0.6	0.2	0.1	1.0	−0.4	−0.2	−0.3
Profb	4.5	3.9	4.7	4.4	−0.8	−0.3	−0.4	−0.1
Schizo	−0.3	−0.7	−0.1	−0.2	4.6	4.3	1.6	4.3
Segment	0.0	−0.1	0.0	−0.1	1.5	1.0	1.1	1.0
Shuttle-landing-control	0.2	0.4	0.0	0.2	−0.5	2.0	1.0	−2.2
Solar-flare-1	10.3	6.4	8.8	8.6	2.5	0.8	0.8	0.0
Solar-flare-2	3.6	3.6	2.7	2.7	0.5	0.4	0.3	0.2
Sonar	4.2	1.6	0.8	1.9	5.2	6.7	3.7	3.2
Soybean	48.7	39.5	44.0	37.3	10.9	7.4	4.5	8.0
Spectf-test	1.3	1.2	1.4	1.0	−0.3	0.0	−0.4	0.2
Spectf-train	11.8	12.5	9.0	8.0	9.7	11.7	8.5	6.8
Spectrometer	0.7	−2.3	3.9	1.7	10.6	11.6	7.8	9.1
Spect-test	0.3	0.2	0.1	0.2	0.1	0.2	0.3	0.2
Spect-train	9.4	8.4	8.0	6.8	11.4	8.8	4.3	4.9
Sponge	2.2	2.1	2.1	1.5	2.3	1.8	1.6	1.2
Tae	−2.8	−1.6	−1.3	−2.6	14.0	13.4	10.9	12.8
Tic-tac-toe	26.8	13.1	15.5	14.2	12.5	10.6	7.8	7.8
Trains	40.7	46.0	43.3	29.3	33.0	42.3	40.7	33.3
Vehicle	1.5	0.5	0.9	0.2	3.8	2.8	3.8	2.2
Vote	9.5	8.0	8.0	6.9	2.1	1.1	1.0	1.0
Vowel	0.2	0.1	0.0	0.1	17.7	19.4	11.8	9.4
Wine	0.2	0.0	0.1	0.1	3.8	3.3	3.1	3.5
Zoo	35.0	25.0	21.3	26.0	14.4	8.8	13.3	14.1

References

Ali, S., & Smith, K. A. (2006). On learning algorithm selection for classification. *Applied Soft Computing*, *6*, 119–138.

Asuncion, A., & Newman, D. (2007). UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Ayat, N., Cheriet, M., & Suen, CY (2002). Optimization of the SVM kernels using an empirical error minimization scheme. In *Proc. of the international workshop on pattern recognition with support vector machine* (pp. 354–369). Berlin: Springer.

Bäck, T. (1993). Optimal mutation rates in genetic search. In *Proceedings of the fifth international conference on genetic algorithms* (pp. 2–8). San Francisco: Morgan Kaufmann.

Bartz-Beielstein, T., Lasarczyk, C. W. G., & Preuss, M. (2005). Sequential parameter optimization. *2005 IEEE congress on evolutionary computation* (Vol. 1, pp. 773–780). New York: IEEE Press

Bensusan, H., & Giraud-Carrier, C. (2000a). Casa batló is in passeig de gràcia or how landmark performances can describe tasks. In *Proceedings of the ECML-00 workshop on meta-learning: Building automatic advice strategies for model selection and method combination* (pp. 29–46).

Bensusan, H., & Giraud-Carrier, C. G. (2000b). Discovering task neighbourhoods through landmark learning performances. In *PKDD'00: Proceedings of the 4th European conference on principles of data mining and knowledge discovery* (pp. 325–330). London: Springer.

Bensusan, H., & Kalousis, A. (2001) Estimating the predictive accuracy of a classifier. In L. De Raedt, & P. Flach (Eds.), *Lecture notes in computer science: Vol. 2167. Machine learning: ECML 2001* (pp. 25–36). Berlin: Springer.

Berrer, H., Paterson, I., & Keller, J. (2000). Evaluation of machine-learning algorithm ranking advisors. In *Proceedings of the PKDD-2000 workshop on datamining, decision support, meta-learning and ILP: Forum for practical problem presentation and prospective solutions*.

Brazdil, P., Gama, J., & Henery, B (1994). Characterizing the applicability of classification algorithms using meta-level learning. In: *ECML-94: Proceedings of the European conference on machine learning* (pp. 83–102). Secaucus: Springer.

Brazdil, P., Soares, C., & da Costa, J. P. (2003). Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results. *Machine Learning*, *50*(3), 251–277.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32.

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*, 121–167.

Chang, C. C., & Lin, C. J. (2001). LIBSVM: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Chapelle, O., Vapnik, V., Bousquet, O., & Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. *Machine Learning*, *46*, 131–159.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

Eitrich, T., & Lang, B. (2006). Efficient optimization of support vector machine learning parameters for unbalanced datasets. *Journal of Computational and Applied Mathematics*, *196*(2), 425–436.

Engels, R., & Theusinger, C. (1998). Using a data metric for preprocessing advice for data mining applications. In: *Proceedings of the European conference on artificial intelligence* (pp. 430–434). New York: Wiley.

Frasch, J. V., Lodwich, A., Shafait, F., & Breuel, T. M. (2011). A bayes-true data generator for evaluation of supervised and unsupervised learning methods. *Pattern Recognition Letters*, *32*(11), 1523–1531.

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

Gagné, C., & Parizeau, M. (2006). Genericity in evolutionary computation software tools: Principles and case study. *International Journal on Artificial Intelligence Tools*, *15*(2), 173–194.

Gama, J., & Brazdil, P. (1995). Characterization of classification algorithms. In C. Pinto-Ferreira & N. Mamede (Eds.), *Lecture notes in computer science: Vol. 990. Progress in artificial intelligence* (pp. 189–200). Berlin: Springer.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning* (1st ed.) Reading: Addison-Wesley.

Hansen, N., & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, *9*(2), 159–195.

Hilario, M., Nguyen, P., Do, H., Woznica, A., & Kalousis, A. (2011). Ontology-based meta-mining of knowledge discovery workflows. In N. Jankowski, W. Duch, & K. Grąbczewski (Eds.), *Meta-learning in computational intelligence, studies in computational intelligence* (Vol. 358, pp. 273–315). Berlin: Springer.

Hsu, C. W., & Lin, C. J. (2002). A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, *13*, 415–425.

Johnson, S. G. (2009). The nlopt nonlinear-optimization package. http://ab-initio.mit.edu/nlopt.

Jones, D. R., Perttunen, C. D., & Stuckman, B. E. (1993). Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, *79*(1), 157–181.

Kalousis, A., & Hilario, M. (2001). Feature selection for meta-learning. In D. Cheung, G. Williams, & Q. Li (Eds.), *Lecture notes in computer science: Vol. 2035. Advances in knowledge discovery and data mining* (pp. 222–233). Berlin: Springer.

Kietz, J. U., Serban, F., Bernstein, A., & Fischer, S. (2010). Data mining workflow templates for intelligent discovery assistance and auto-experimentation. In *Proceedings of the ECML/PKDD-10 workshop on third generation data mining: Towards service-oriented knowledge discovery* (pp. 1–12).

King, R. D., Feng, C., & Sutherland, A. (1995). Statlog: Comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence*, *9*(3), 289–333.

Kohonen, T. (1998). Learning vector quantization. In *The handbook of brain theory and neural networks* (pp. 537–540). Cambridge: MIT Press.

Kolda, T. G., Lewis, R. M., & Torczon, V. (2003). Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, *45*, 385–482.

Köpf, C., Taylor, C., & Keller, J. (2000). Meta-analysis: From data characterisation for meta-learning to meta-regression. In *Proceedings of the PKDD-00 workshop on data mining, decision support, meta-learning and ILP*.

Merelo, J. J., Prieto, A., & Morán, F. (2001). Optimization of classifiers using genetic algorithms. In M. Patel, V. Honavar, & K. Balakrishnan (Eds.), *Advances in the evolutionary synthesis of intelligent agents* (pp. 91–108). Cambridge: MIT Press.

Merz, C. (1995) Dynamical selection of learning algorithms. In *Learning from data, artificial intelligence and statistics*. Berlin: Springer.

Momma, M., & Bennett, K. P. (2002). A pattern search method for model selection of support vector regression. In *Proceedings of the SIAM international conference on data mining*.

Pfahringer, B., Bensusan, H., & Giraud-Carrier, C. (2000). Meta-learning by landmarking various learning algorithms. In *Proceedings of the seventeenth international conference on machine learning* (pp. 743–750). San Mateo: Morgan Kaufmann.

Plantenga, T. D. (2009). *Hopspack 2.0 user manual* (Tech. Rep. SAND2009-6265). Sandia National Laboratories, Albuquerque, NM and Livermore, CA.

Shafait, F., & Breuel, T. M. (2011). The effect of border noise on the performance of projection-based page segmentation methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(4), 846–851.

Skala, K. (2004). PARF—Parallel RF algorithm. Centre for Informatics and Computing of Ruđer Bošković Institute. http://www.irb.hr/en/research/projects/it/2004/2004-111/.

Smith-Miles, K. A. (2008). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys*, *41*(1), 1–25.

Soares, C., & Brazdil, P. B. (2006). Selecting parameters of SVM using meta-learning and kernel matrix-based meta-features. In *SAC'06: Proceedings of the 2006 ACM symposium on applied computing* (pp. 564–568). New York: ACM.

Soares, C., Brazdil, P. B., & Kuba, P. (2004). A meta-learning method to select the kernel width in support vector regression. *Machine Learning*, *54*(3), 195–209.

Sohn, S. Y. (1999). Meta analysis of classification algorithms for pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21*(11), 1137–1144.

Vilalta, R., Giraud-carrier, C., Brazdil, P., & Soares, C. (2004). Using meta-learning to support data mining. *International Journal of Computer Science and Applications*, *1*(1), 31–45.

Vlachos, P. (1998). StatLib datasets archive. Department of Statistics, Carnegie Mellon University. http://lib.stat.cmu.edu/l.

Wall, M. (1996). Galib: A C++ library of genetic algorithm components. Mechanical Engineering Department, Massachusetts Institute of Technology.

Yao, X. (1993). Evolutionary artificial neural networks. *International Journal of Intelligent Systems*, *4*, 539–567.