Pattern Recognition 47 (2014) 1664–1673

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Efficient feature size reduction via predictive forward selection

Matthias Reif [a,*], Faisal Shafait [b]

[a] German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany
[b] The University of Western Australia (UWA), Perth, Australia

ARTICLE INFO

Article history:
Received 12 November 2012
Received in revised form
3 June 2013
Accepted 8 October 2013
Available online 18 October 2013

Keywords:
Feature selection
Meta-learning
SVM
Classification
Forward selection

ABSTRACT

Most of the widely used pattern classification algorithms, such as Support Vector Machines (SVM), are sensitive to the presence of irrelevant or redundant features in the training data. Automatic feature selection algorithms aim at selecting a subset of features present in a given dataset so that the achieved accuracy of the following classifier can be maximized. Feature selection algorithms are generally categorized into two broad categories: algorithms that do not take the following classifier into account (the filter approaches), and algorithms that evaluate the following classifier for each considered feature subset (the wrapper approaches). Filter approaches are typically faster, but wrapper approaches deliver a higher performance. In this paper, we present the algorithm – Predictive Forward Selection – based on the widely used wrapper approach *forward selection*. Using ideas from meta-learning, the number of required evaluations of the target classifier is reduced by using experience knowledge gained during past feature selection runs on other datasets. We have evaluated our approach on 59 real-world datasets with a focus on SVM as the target classifier. We present comparisons with state-of-the-art wrapper and filter approaches as well as one embedded method for SVM according to accuracy and run-time. The results show that the presented method reaches the accuracy of traditional wrapper approaches requiring significantly less evaluations of the target algorithm. Moreover, our method achieves statistically significant better results than the filter approaches as well as the embedded method.

1. Introduction

Pattern recognition is often defined as the task of assigning a label to a given input data instance described by a set of measured or calculated values called *features*. If the task is to assign a label from a set of pre-defined labels, it is termed as *classification*. Pattern classification has a wide range of applications ranging from Astronomy to Molecular Biology and DNA sequence analysis. Owing to the broad applications base, a large number of pattern classification algorithms based on different theoretical foundations have been proposed in the literature. The no-free-lunch theorem [41] tells us that there is no pattern classification algorithm that can be uniformly better than all other algorithms on each problem instance. However, Support Vector Machines (SVM) [17] have shown outstanding performance on many widely accepted benchmark datasets originating from different domains, making them one of the most favorite pattern classification algorithms today. One common limitation of many pattern classification algorithms, including SVMs, is that their performance degrades if the feature vector contains irrelevant or redundant features [40]. For most of the pattern classification problems, many potentially good features are extracted during data collection. The actual usefulness of each extracted feature is not known at that stage. Automatic feature selection methods are usually used to reduce the number of features to be fed to the following pattern classification method for training. Removing irrelevant ore redundant features not only improves overall classification accuracy, but also reduces the dimensionality of the data thereby shortening the training and application time of the learning scheme. Furthermore, shorter feature vectors help the classifier in better coping with the curse of dimensionality.

Two general approaches for automatic feature selection are filter methods and wrapper methods [8,24]. Filter methods select features based on their discriminative power according to the target variable without taking the actual learning scheme into account. Measures based on correlation or information theory are usually used to determine the quality of a feature [21,33]. The top-ranked features according to the used measure construct the final set of features. The number of best features used for learning is a typical parameter of filter approaches. Therefore, the best value is often empirically found by examining multiple values.

Since filter approaches typically treat each feature independently, they are not able to remove redundant features. The minimum redundancy-maximum relevance (MRMR) feature selection method tries to select features that are maximally

* Corresponding author. Tel.: +49 631205754185.
E-mail addresses: matthias.reif@dfki.de (M. Reif), faisal.shafait@uwa.edu.au (F. Shafait).

0031-3203/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2013.10.009

relevant to the target variable but also minimally redundant among themselves [28]. The Fast Correlation Based Filter (FCBF) also tries to remove irrelevant and redundant features [42]. Features with a symmetrical uncertainty according to its class below a given threshold are removed because they are considered as irrelevant. Additionally, only features that do not have any approximate Markov blanket in the current set of remaining features are kept in order to reduce redundancy. Mao [26] presented a filter approach by selecting features in an orthogonal space. Different class separability measures are used to select the most promising features.

Although filter approaches are typically fast, one of their weaknesses is that the actual target classification algorithm is not taken into consideration. Therefore, the same feature subset is selected regardless of the following classification algorithm being used. Different classification algorithms have different degrees of sensitivity to spurious features. Hence, it is useful to take the target classifier into account while doing feature selection to optimize the true objective function (i.e. the accuracy of the classifier on new data). To solve this problem, wrapper approaches involve the target algorithm in the feature selection process [24]. The quality of a feature subset is determined by evaluating the actual classifier using the selected feature subset (i.e. training the classifier on the subset of the training data containing the selected features only, and testing it on the similarly extracted subset of validation data). This has the advantage of including the target classification algorithm into the feature selection procedure which leads to the most suitable feature subset for the given classifier. The sequence and the number of feature set evaluations are specified by the actual search strategy used for feature selection. The simplest strategy is a brute-force approach, that evaluates all possible subsets of features. This is computationally very expensive due to its combinatorial complexity and becomes infeasible when a large number of features are involved.

Forward selection is a frequently used wrapper approach. It starts with an empty set of features and iteratively adds unused features. In the first iteration, all features are evaluated individually. During each following iteration, one feature is added to the feature subsets of the previous iteration and the newly created feature subsets are evaluated again. In order to reduce the number of evaluations, only the k best feature subsets are kept after each iteration. If k is low, the run-time decreases, but the algorithm might miss better feature subsets. One iteration of forward selection is illustrated in Fig. 1: Starting with two feature subsets ($k=2$) of size three, all possible subsets of size four are evaluated. Afterwards, the two best subsets are kept as the basis for the next iteration. A stopping criterion that is often used in forward selection is that the algorithm stops if the accuracy of the target algorithm using the best selected subset does not improve in m consecutive iterations.

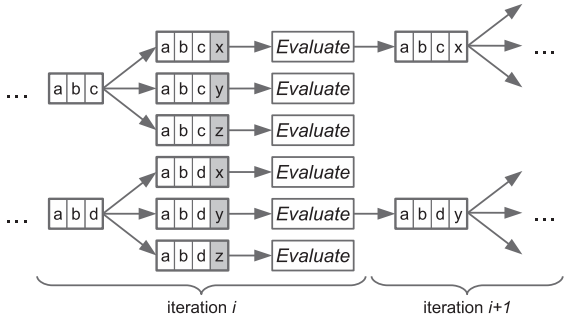

Fig. 1. One iteration of *forward selection*: Each unused feature before the start of current iteration is added to the previously selected feature subsets. All resulting subsets are evaluated by applying the learner and the k best subsets are used as the basis for the next iteration ($k=2$ in this illustration).

Since forward selection is a greedy method (only keeps the k best feature subsets from the previous iteration), it does not guarantee to find the optimal feature subset. The method *backward elimination* works analogically, but starts with the complete set of features and removes one feature in each iteration.

In this paper, we present a novel algorithm that significantly reduces the time of the wrapper method *forward selection* without a statistical significant decrease of accuracy. Knowledge about previous feature selection runs of other datasets is used within a novel estimation step introduced into *forward selection*. Although, the presented method is independent of the target classification algorithm, we chose the SVM classifier [17] for experiments in this work. SVM is a non-linear maximum margin classifier using a kernel function. A typical kernel is the radial basis function (RBF). The performance of an RBF-kernal SVM heavily depends on the values of its sensitive parameter C and γ. The parameter C of SVMs controls the trade-off between minimizing the number of wrongly labeled training samples and maximizing the margin, whereas the parameter γ controls the width of the RBF kernel. A more detailed description of the SVM classifier including a discussion on its parameters can be found in [14]. The choice of using SVMs as the target classifier is not only motivated by the wide-spread use of the SVM classifier, but also due to computationally demanding training of SVMs.

Besides the two general categories of wrapper and filter approaches, embedded methods for feature selection have been proposed that perform feature selection within the classifier. These approaches are specific to the particular classification algorithm that is modified to inherently perform feature selection. It should be mentioned here that some classification algorithms have feature selection built in their original algorithm. One example for such classifiers are decision trees, such as CART [13] or Random Forests [12]. Although SVMs do not inherently perform feature selection, embedded methods for SVMs have been proposed. Penalized SVMs use an additional penalty term within the optimization step for selecting the features. Different definitions of this penalty term have been investigated. The L1-penalty was proposed by Bradley and Mangasarian [9], which is applicable to linear kernels only. Zhang et al. [43] combined the smoothly clipped absolute deviation (SCAD) penalty with the SVM classifier. A mixture of a wrapper and an embedded approach was presented by Weston et al. [40] by optimizing the feature weights using a gradient descent method for SVMs. However, embedded methods are classifier specific and cannot directly adapted to any other classifier, which is one of their biggest drawbacks.

The rest of this paper is structured as follows. First, we explain the proposed approach in detail in Section 2. The evaluation of the method is given in Section 3. The final section concludes the paper with a summary.

2. Predictive feature selection

Consider a dataset D consisting of N samples $D=\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N\}$ and p features such that each sample $\vec{x}_i \in \mathbb{R}^p$. Let t be the target variable that needs to be predicted when a new sample $\vec{x}$ arrives. For regression problems, t is considered a continuous variable whereas for classification problems t is discrete. The goal of feature selection is to find from the p-dimensional observation space $\mathbb{R}^p$, a subspace of q features ($q \le p$) $\mathbb{R}^q$, that maximized the prediction accuracy for the target variable t. Since the total number of subspaces is 2^p-1, exhaustively searching for the optimal subspace becomes computationally infeasible even for moderate values of p.

Let D_q represent a subset of the original dataset D such that it contains the same number of samples $D_q = \{\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_N\}$, but each sample is reduced to q-dimensions: $\vec{y}_i \in \mathbb{R}^q$ and $\mathbb{R}^q \subseteq \mathbb{R}^q$. Forward selection starts in the first iteration with $q=1$ and applies the learning scheme Γ that maps the input feature vector $\vec{y}_i$ to the output value t_i:

$$\Gamma : \vec{y}_i \in \mathbb{R}^p \rightarrow t_i \in \mathbb{R} \tag{1}$$

for each candidate data subset D_q. For a classification function Γ_c, the output value t_i belongs to a predefined set of classes $\{\omega_1, \omega_2, \ldots, \omega_C\}$. Let $\mathcal{P}_{D_q}$ represent the performance (classification accuracy) achieved by the learning scheme using the data subset D_q. For the next iteration, only those k data subsets are kept that lead to the highest accuracy using the learning algorithm Γ_c.

Since in each iteration all candidate data subsets are evaluated, forward selection requires *repeated application* of the learning algorithm. The performance achieved by most of the learning algorithms (including SVMs) on a given dataset highly depends on the specific values of their parameters used during training. Therefore, to achieve the best possible results, sensitive parameters of these algorithms need to be optimized on each candidate data subset. For instance, a common strategy employed while training SVMs on a given dataset is to do a search for its sensitive parameters γ and C over a 15×15 grid [22]. This leads to a huge computational demand from the feature selection process.

The key idea behind our work is to avoid the computationally expensive step of training the target classifier on all candidate data subsets. Since the main aim of training the classifier on a data subset D_q is to find its accuracy $\mathcal{P}_{D_q}$ on D_q, we propose to *predict* the accuracy $\hat{\mathcal{P}}_{D_q}$ of the classifier on D_q instead of computing it. This prediction is done by computing different *meta-features* of D_q that characterize intrinsic properties of the dataset itself (a brief discussion of different meta-features used in this work is given in Section 2.1). Hence, all candidate data subsets in a particular iteration can be ranked based on the *predicted* accuracy of the target classifier. Finally, only a certain fraction of the top-ranked data subsets are actually used to train and subsequently compute the accuracy of the classifier.

Fig. 2 illustrates one iteration of the presented approach. The newly introduced step of quality estimation (i.e. accuracy prediction) is performed during each iteration and controls which feature set will actually be evaluated. Under the assumption that predicting the quality is much faster than evaluating the target algorithm, the run-time of the overall forward selection can be decreased dramatically. The number of feature sets that are actually evaluated is a parameter of the proposed method. Either a fixed number of feature sets or a fraction of all subsets can be used. We prefer to use a certain fraction of the candidate data subsets since this is more applicable for differently sized datasets. During the later evaluation, we refer to this parameter as the *evaluation rate* η. Since the quality estimation might not deliver perfect results, this parameter controls the trade-off between speed-up and accuracy. If the estimation would deliver a correct ordering of the feature subsets, no actual evaluation would be necessary. If the parameter is set to 100%, the results are equivalent to the traditional *forward selection* since all candidate subsets are evaluated regardless of the quality estimation.

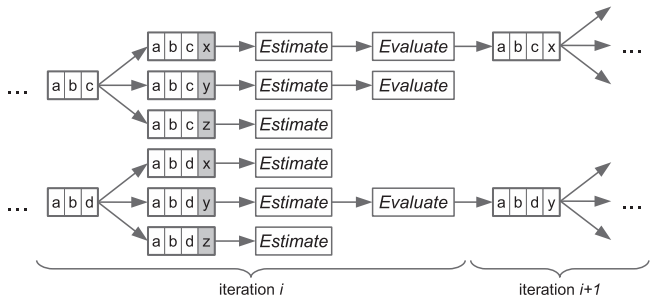

Fig. 2. One iteration of the presented approach: only the most promising feature subsets are evaluated, depending on the quality estimated in a prior step.

2.1. Meta-features

Different measures can be used to extract certain characteristics of a dataset. Each measure calculates one value describing one property of the whole dataset. One simple example of such a measure is the number of classes contained in a dataset. Multiple measures can be combined to form a feature vector that provides a compact but meaningful characterization of the dataset. Since this vector is computed from the features of the dataset and it is itself a feature vector as well, it is called a meta-feature vector of the dataset. The properties of a dataset that are represented by the individual elements of the meta-feature vector are called meta-features. The actual measures included into the meta-feature vector can be arbitrary. There is no fixed definition of the meta-feature vector since the choice of features to be used strongly depends on their application in a subsequent learning scheme (often termed as *meta-learning*). A proper choice of meta-features is a crucial point in developing meta-learning systems. Therefore, several types of meta-features using different theoretical foundations have been proposed in the literature. Based on their underlying foundations, meta-features can be categorized into five groups: simple, statistical, information-theoretic, model-based, and landmarking. Here we provide a brief overview about each of these meta-feature types to give the reader a better understanding of what constitutes the meta-feature space.

2.1.1. Simple meta-features

Simple meta-features are the most straightforward characteristics of a dataset. Examples of such features are the number of samples, the number of features, or the number of classes [23,27]. Additionally, simple ratios such as the dimensionality of a dataset defined by the division of the number of samples by the number of features or the ratio of numerical/nominal features might be calculated. Due to their simplicity, there is almost no computational effort calculating them, but their descriptive power is very limited as well. However, in combination with more sophisticated meta-features, simple measures can improve the performance of a meta-learning method.

2.1.2. Statistical meta-features

Statistical meta-features use measures and analysis methods from statistics. The single features of a dataset are treated as probability distributions and well known statistical measures are applied. Two frequently used examples of this group of meta-features are the kurtosis and skewness [27,19]. The kurtosis measures the "peakedness" of a probability distribution. The skewness is a measure of the asymmetry of the probability distribution. Other commonly used statistical measures are the standard deviation ratio, the correlation between pairs of features, the canonical correlation for the best single combination of features, and the normalized first eigenvalue of the canonical discriminant matrix [27]. Statistical meta-features are computationally more intensive as compared to simple meta-features. Especially the computation of measures that are defined between pairs of features, such as the correlation, can be very time consuming. Since they are computed for each possible feature pair, their complexity grows quadratically with the number of features. Therefore, such measures might not be suitable for datasets with a very high number of features.

2.1.3. Information-theoretic meta-features

Information-theoretic meta-features are based on the Shannon-entropy for a discrete random variable. The entropy is a measure for the average uncertainty of the random variable and it is calculated for each feature and the class label. Castiello et al. [15] propose to further normalize these entropy values by dividing them by the logarithm of the number of samples. Often used meta-features based on entropy are the joint entropy and the mutual information of the class label and a feature as well as the equivalent number of attributes and the signal-to-noise ratio [27,34].

2.1.4. Model-based meta-features

Model-based meta-features do not use well-known measures such as statistical tests or entropy-measures but features particularly developed for meta-learning. Features are extracted using a two-stage approach. First, a decision tree is trained on the dataset. Then, different properties of the created tree are used as meta-features [5,29]. Bensusan et al. [5] proposed simple properties such as the ratio of the number of nodes to the number of features, the ratio of the number of nodes to the number of samples but also more complex characteristics such as measures for the shape, homogeneity, imbalance, and internal symmetry of the decision tree. Peng et al. [29] also used the maximum, minimum, mean, and standard deviation of different measures, such as the number of nodes at a level, the length of the branches, and the number of occurrences of attributes in the tree.

2.1.5. Landmarking meta-features

In landmarking, a simple and computationally light learning algorithm is training on the target dataset. The resulting performance of the learning algorithm is directly used as a meta-feature. Such a simple learner is typically called a *landmarker*. Algorithms often applied as landmarkers are Naive Bayes, Decision Nodes, and Linear Discriminant [30,6]. Decision Nodes split the data using one feature only and can be viewed as a decision tree containing just one split. The splits of the Decision Nodes are created according to some splitting criterion, e.g., the information gain. The accuracies of the best and the worst node according to the splitting criterion are used as meta-features as well the accuracy of a random node and the average accuracy of all decision nodes.

2.2. Accuracy prediction

Several methods have been proposed in the literature for classifier selection or accuracy estimation for a given dataset. These can be broadly categorized into three main groups:

1. Predicting the best classifier for the given dataset [4,1].
2. Predicting a ranked list of classifiers for the given dataset [10,11,38].
3. Predicting the accuracy of a particular classifier on the given dataset [20,37,25,7,32].

Among these methods, predicting the accuracy of the target classifier [32] for the given dataset D_q can be directly used to get the accuracy estimate $\hat{\mathcal{P}}_{D_q}$. In a traditional meta-learning scenario, the goal is to find the best classifier for a given dataset. We use a different view here. In our scenario, we do not want to select the best algorithm for a given dataset, but to select the best data subset $\hat{D}_q$ for a given target algorithm. Keeping the target algorithm fixed (SVM in this work), we use a regression model that is able to predict the accuracy of the target algorithm for the data subsets. The creation of the regression model is based on the experience knowledge gathered from previous feature selection runs of other

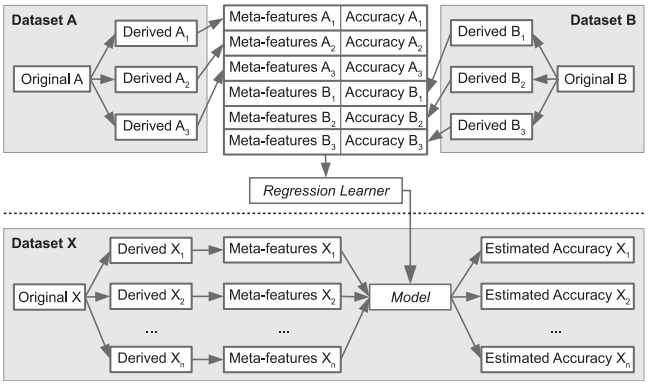

Fig. 3. The meta-features and the accuracy of the datasets derived from datasets A and B construct a meta-dataset that is used to create a regression model (top). The model is then applied on the meta-features of the datasets derived from a new dataset X during the feature selection process (bottom).

datasets. Since multiple feature subsets are evaluated during past feature selection runs, the corresponding data subsets including the achieved accuracies are used as the knowledge base.

A dataset containing only a subset of features has different characteristics than the corresponding original dataset containing all available features. Meta-features are able to describe these characteristics of a dataset and, therefore, are used to represent the data subsets. A regression model learns the relation between these characteristics of (derived) datasets and their accuracy according to a classifier. Note that although the dimensionality of each data subset might be different, the same meta-features are extracted from each of them. Let m be the dimensionality of the meta-feature space, such that each data subset D_q is represented as a single vector $\vec{d}_q \in \mathbb{R}^m$ in that meta-feature space. This property enables us to train a regression model g in the m-dimensional meta-feature space $\mathbb{R}^m$, independent of the dimensionality of the original dataset D_q.

To estimate the quality of a feature subset, the regression function Γ_g is applied to the m-dimensional meta-feature vector $\vec{d}_q$ representing the following data subset:

$$\Gamma_g : \vec{d}_q \in \mathbb{R}^m \rightarrow \hat{\mathcal{P}}_{D_q} \in \mathbb{R} \tag{2}$$

The regression approach is illustrated in Fig. 3, including training (top) and application (bottom): For each of the two known datasets A and B, three data subsets are derived by using certain subsets of features. The meta-features of these derived datasets as well as the accuracies achieved by the target classifier construct a meta-dataset. Within this dataset, one instance describes one data subset and the corresponding accuracy of the target classifier. Using the accuracy as the target variable, a regression learner is used to learn a model. This training phase is independent of the feature selection process for a new dataset and the created model can be used for selecting features of any new dataset. Furthermore, since no explicit knowledge about the feature selection process is required in the feature subset quality estimation stage, the method can be easily applied in other sequential-search-based methods such as *backward elimination*.

2.3. Application example

In this section, we describe different steps that need to be taken for practical application of our proposed method for feature selection. Note that, since this method is knowledge-based, data from previous feature selection experiments is needed to train the system. In practice, such a knowledge-base can either be built

from scratch using public datasets (e.g. those from the UCI repository) or a public knowledge-base can be directly used.[1]

To build the knowledge-base, the following steps are applied:

Collect multiple datasets $D^1, D^2, \ldots, D^j$.
For each individual dataset D^j:
  For each candidate feature set of D^j:
  - Extract data subset D_q containing only the class label and the candidate features.
  - Train the target classifier on D_q and compute its accuracy $\mathcal{P}_{D_q}$ using cross-validation.
  - *Compute m meta-features* $\vec{d}_q \in \mathbb{R}^m$ *of the data subset* D_q *such that the whole dataset is represented as a single point in the m-dimensional meta-feature space.*
  - *Store accuracy of the target classifier* $\mathcal{P}_{D_q}$ *and the computed meta-features* $\vec{d}_q$ *as one record in meta-dataset.*

Train a regression algorithm on the meta-dataset using $\vec{d}_q$ as input features and $\mathcal{P}_{D_q}$ as the target variable.

After a regression model has been trained, it can be readily used within the forward selection to predict the accuracy of the target classifier on newly created data subsets. To illustrate the steps involved in our *predictive forward selection* algorithm, we consider a toy dataset in which the features $F = \{f_1, \ldots, f_n\}$ are used to represent each data sample. We assume an evaluation rate η of 50% and keep only the best ($k=1$) candidate feature set during each iteration.

1. Start with an empty set of features $M := \varnothing$
2. For each feature $f_j \in F \backslash M$:
   - Extract data subset D_q containing only the class label and the features M and f_j.
   - Compute m meta-features $\vec{d}_q \in \mathbb{R}^m$ of the data subset D_q.
   - Apply regression model Γ_g from Eq. (2) to predict the accuracy $\hat{\mathcal{P}}_{D_q}$.
3. Rank data subsets according to their predicted accuracy.
4. Train and compute the accuracy $\mathcal{P}_{D_q}$ of the target classifier on the $\eta = 50\%$ top-ranked data subsets.
5. Add the feature f_j of the data subset with the highest computed accuracy ($\mathcal{P}_{D_q}$) to the set M of selected features.
6. If stopping criterion is fulfilled, stop. Otherwise, continue at step 2.

3. Experimental setup

We evaluated the presented approach using RBF-kernel SVM as the target classifier. Training the SVM on each data subset involved a 10×10 grid search to find suitable values for its parameters C and γ. Ten-fold cross-validation was performed to get a reliable estimate of the accuracy achieved by SVM. Therefore, the evaluation of one feature subset involved training of 1000 SVMs. Since using two different subsets of the same dataset may have different intrinsic properties, independent parameter optimization for the feature subsets is necessary.

We started the evaluation with 22 rather small datasets containing only 5–14 features. Such small datasets were used because the evaluation of all possible feature combinations is computationally feasible. This is important since it enables us to compare the results of the other approaches to the optimum (result obtained by brute-force search). Additionally, by knowing the optimal accuracy, we can adjust the parameters of the *forward selection* and the presented approach to reach almost optimal accuracy without performing unnecessary feature evaluations. This is important for the comparison of the run-time.

The datasets were randomly selected from the UCI machine learning repository [2], StatLib [39], and the book "Analyzing Categorical Data"[35].

3.1. Learning the regression model

The experience knowledge is created based on the 22 selected datasets. First, meta-features are calculated for all possible feature subsets of each dataset. We computed 51 meta-features from five different groups (see Section 2.1): 17 simple, 5 statistical, 6 information-theoretic, 17 model-based, and 6 landmarking meta-features. The same meta-features as used in Reif [31] were computed using the same R-script. The performance is evaluated by applying the SVM classifier including the parameter optimization.

Since the number of features is different in the selected datasets (varying from 5 to 14 features), the number of feature subsets is different as well. For n features, the total number of possible feature subsets is $2^n - 1$. Each feature subset generates one instance of the training data for the regression model. Therefore, a dataset with 4 features will generate 15 training instances, whereas a dataset with 14 features will generate 16,383 instances. In order to balance the influence of the different datasets while training the regression model, we used at most 511 randomly selected instances per dataset. This is the number of possible feature subsets of a dataset with nine features.

Applying a leave-one-out cross-validation, we created separate regression models for each of the 22 datasets. We used the ε-SVR – the regression variant of the Support Vector Machine [36] – as the meta-learner and libSVM [16] as its implementation. The ε-SVR was used because it is also able to model linear as well as non-linear regression functions. The radial basis function is used as the kernel. In Section 4.5, we present a comparison of different regression algorithms when applied as a meta-learner.

The training data of the regression model for each dataset contains only the experience knowledge gathered from the feature selection runs of the remaining 21 datasets. Therefore, the exact number of training instances slightly differs for different datasets, but the training data is always based on 21 datasets.

All 51 meta-features were normalized to the interval [0; 1] and an automatic feature selection on the meta-features was applied independently for each regression model. We used a *forward selection* with a three-fold cross-validation and a limit of at most 10 selected meta-features. The ε-SVR was applied using default parameters. Using the selected meta-features only, we optimized the sensitive parameters of the ε-SVR, γ and C, using a 15×15 grid and a three-fold cross-validation. The parameter γ controls how the data points are transformed into a higher dimensional space to allow non-linear functions. The parameter C defines the penalty of errors greater than ε. As the last step, the final regression model was trained using the selected meta-features and the selected parameters.

The run-time for creating such a regression model strongly depends on the size of the training data, the parameter optimization,

[1] The knowledge base developed in this paper is publicly available upon request from the authors.

and the feature selection. However, since the model can be created in advance and independently from the actual feature selection task, the run-time is not considered as an important factor. For instance, it is worthwhile investing huge computational effort in creating a model that will be used over a longer duration for many upcoming feature selection tasks.

4. Results and discussion

4.1. Evaluation of accuracy

Besides the presented approach, we evaluated different wrapper, filter, and embedded approaches. All wrapper and filter approaches use exactly the same accuracy for the same feature subsets in order to remove random factors within the cross-validation. The comparison of accuracy and run-time is presented within the next subsections.

4.1.1. Comparison to wrapper approaches

We applied five different wrapper approaches to all 22 datasets: using all features, evaluating all possible feature subsets following a brute force approach, traditional forward selection, the presented approach, and using random selection within the presented approach. For the last method, the quality estimation of the presented method is replaced by a random guessing in order to show the effectiveness of the regression-based prediction.

For a fair and convincing comparison, we adjusted the parameters of the traditional *forward selection* such that there is no statistical significance compared to the brute force approach while requiring as few as possible classifier evaluations. We used the Wilcoxon signed-rank test with a confidence level of 95% as recommended by Demšar [18]. As a result, only the two best subsets at each iteration were kept and the search was stopped after one iteration without improvement. The algorithms have a parameter, defining the number of allowed generations without any improvement. Here, the parameter was increased by one, allowing one generation without improvement more. The evaluation rate η was set to 10% (only 10% of the candidate subsets are actually evaluated at each iteration).

Fig. 4 shows the accuracy values achieved for each single dataset by the wrapper approaches. The comparison of the results using all features and applying a brute force search shows that reducing the set of features increases the accuracy on most of the datasets.

In Fig. 5, the same comparison is shown as a boxplot. In both plots, it is visible that the presented method achieves accuracy values comparable to the results of the traditional *forward selection*. In order to test if the differences are statistically significant, we again applied a Wilcoxon signed-rank test with a confidence level of 95%. The difference of accuracy between the traditional *forward selection* and the presented method was not found to be statistically significant.

Using all available features and the presented approach with random selection achieved lower accuracy. The differences in accuracy between the proposed method and these two methods were found to be statistically significant using the same Wilcoxon signed-rank test.

4.1.2. Comparison to filter and embedded approaches

Besides the wrapper approaches, we compared the presented approach with two state-of-the-art filter approaches: the minimum redundancy-maximum relevance (MRMR) method [28] and the fast correlation-based filter (FCBF) [42]. Since both filter approaches contain the desired number of features to be selected as a parameter, we applied these methods with all possible feature set sizes and determined the best feature set by evaluating the target classifier. Therefore, the number of classifier evaluations in the filter approaches is equal to the number of features in the considered dataset.

Furthermore, the penalized SVM using the SCAD penalty was evaluated. The important parameter of this method λ was optimized in the interval $[2^{-10}, 2^{10}]$. We used the existing implementation of penalized SVMs by Becker et al. [3] that includes also the optimization of λ. Because the algorithm contains a singular value decomposition, it fails on datasets with a zero within-class variance. This happens if any feature of a dataset is constant for one class within one fold of the cross-validation, which is often the case for the used datasets. Therefore, we regularized these folds by adding small random terms to these features, but keeping them perfectly separable. However, using this regularization, the

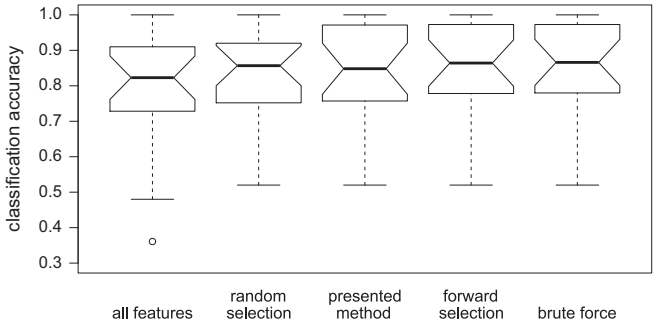

Fig. 5. Boxplot of the accuracies on the 22 datasets achieved by the different wrapper methods.

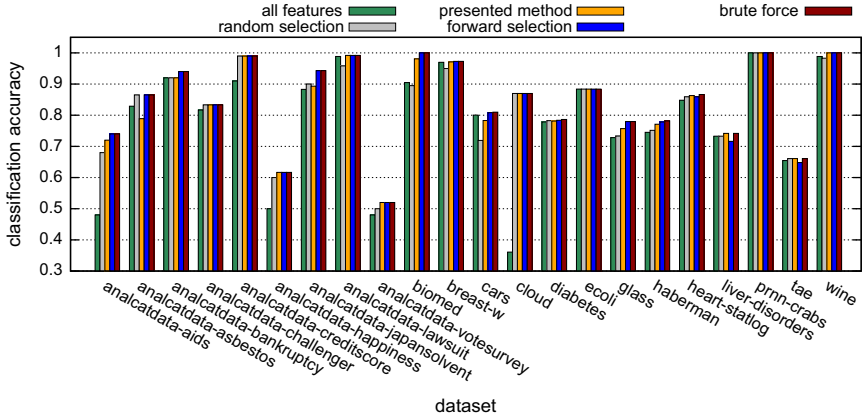

Fig. 4. The accuracies achieved on each dataset by using all available features, the presented method, a traditional *forward selection*, and a brute force search.

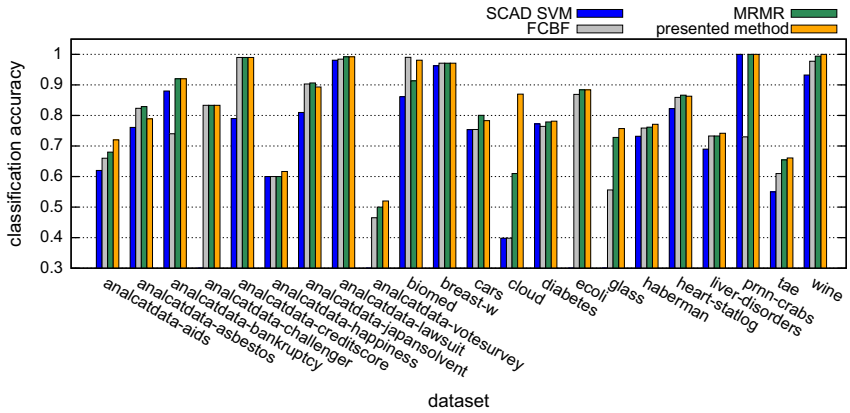

Fig. 6. The accuracy of the presented method compared to the accuracy values achieved by the two filter approaches MRMR and FCBF, and the embedded method SCAD SVM.

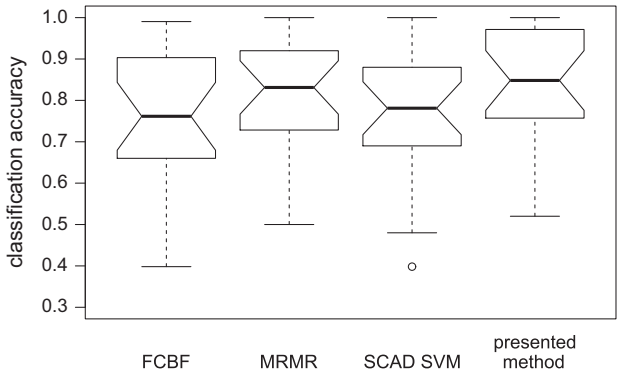

Fig. 7. Boxplot of the accuracies on the 22 datasets achieved by the presented approach and three filter and embedded methods.

implementation still failed on three datasets. For the boxplot, the accuracy achieved by using all features was used for these datasets instead.

However, both filter methods and the embedded method do not reach the accuracy of the presented method for many datasets as visible in Fig. 6. The comparison using a boxplot is additionally shown in Fig. 7. To determine whether these differences are statistically significant, the same significance test as before was applied. While the presented method is statistically significantly better than FCBF and SCAD SVM, the difference to MRMR is not significant. However, if one more iteration without improvement is allowed in our method, the difference from MRMR becomes statistically significant.

It is notable that especially the “cloud” dataset seems to contain irrelevant features decreasing the SVM performance. While *forward selection* and the presented approach achieve the optimal accuracy, SCAD SVM achieved an accuracy only slightly higher than using all features. Surprisingly, the SCAD SVM did not achieve the accuracy of using all features for multiple datasets, for example for the “tea” and “wine” datasets.

4.2. Evaluation of computation time

We counted the number of SVM evaluations during the traditional *forward selection* as well as during the presented method since this is the most time-consuming part and its comparison across different datasets is much more clearer than comparing absolute run-times on a particular machine.

The number of SVM evaluations for MRMR, the traditional *forward selection* as well as the presented approach is plotted in Fig. 8. As expected, the standard *forward selection* requires much more evaluations than the filter approach. The number of evaluations performed in the presented method is also statistically significantly lower than those for the standard *forward selection* and approximately the same as for MRMR. Since only 10% of the feature subsets are actually evaluated during each iteration, this result is not surprising. However, the used stopping criterion (stop after a defined number of successive iterations without improvement) might lead to a higher number of iterations. If the proposed method selects different feature sets during an iteration than the traditional *forward selection* does, the stopping criterion might be fulfilled after a different number of iterations.

4.3. Scalability on datasets with a larger number of features

We also evaluated the presented feature selection method on datasets containing a larger number of features. We wanted to evaluate how the approach scales up in comparison to standard *forward selection* and if the previously shown advantages still hold.

We randomly selected 37 additional datasets with up to 78 features. Applying wrapper methods on datasets with several hundreds or thousands of features is computationally unfeasible, especially if the parameter optimization of the target classifier is performed for each candidate subset.

The evaluation on the selected datasets is the same as that for the 22 smaller datasets. However, the evaluation on the bigger datasets is completely separated from the previously used 22 datasets and no knowledge from these datasets is used.

Fig. 9 shows the comparison of the presented method and the standard *forward selection* for these relatively bigger datasets. For this evaluation, both methods use exactly the same parameter values: they keep the two best subsets during each iteration and will be stopped after two successive iterations without improvement. Different evaluation rates were investigated for the presented approach. This parameter is important and controls the trade-off between accuracy and run-time. A rate of 100% will lead to a traditional *forward selection*.

As expected, higher evaluation rates lead to better results. However, the differences are not very large, which is an indication of the effectiveness of the predictions: increasing the number of evaluated subsets does not lead to significantly better results which means that the actually best subsets were already selected using the lower evaluation rate. The difference between the presented approach and the standard *forward selection* is already for an evaluation rate of 10% statistically not significant.

The required evaluation rate is mainly influenced by the precision of the regression model. If the prediction quality can

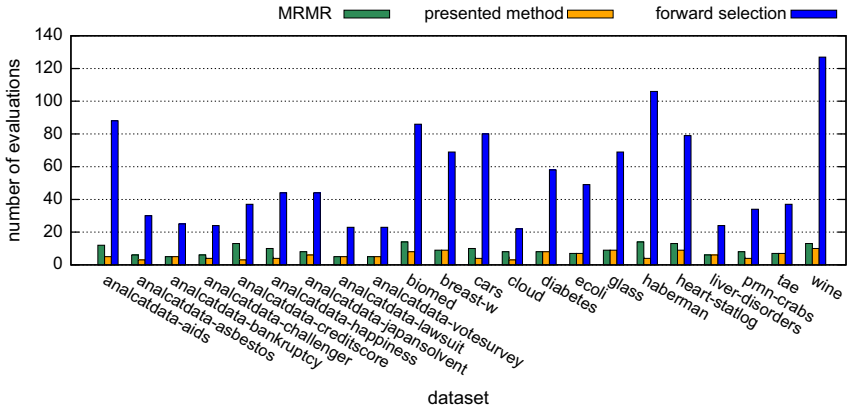

Fig. 8. The number of required evaluations of the SVM within the MRMR filter approach, the presented method, and the traditional *forward selection*.

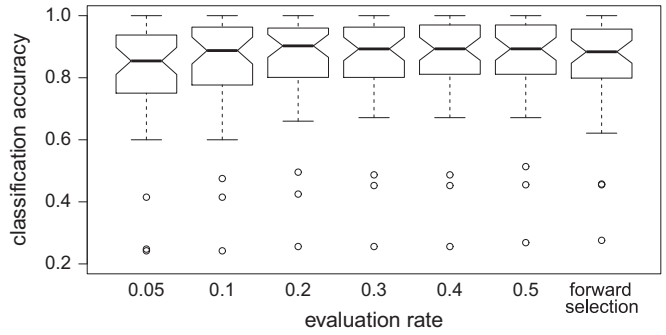

Fig. 9. Boxplot of the accuracies on the 37 bigger datasets achieved by the presented method using different evaluation rates and the standard *forward selection*.

be further increased, e.g., by using different meta-features, more training data, or using the method with a different target classifier, a lower evaluation rate might be sufficient.

4.4. Effect of cardinality of the base datasets

To investigate whether the cardinality (number of features) of the base datasets has an influence on the prediction accuracy of the learned regression function, we swapped the regression models applied on the small-sized datasets with those applied on the larger sized datasets. That means, for feature selection on a small dataset, the regression model trained on the bigger datasets is applied and vice versa.

Fig. 10 shows the boxplots of the results. While using the experience about bigger datasets for the feature selection of datasets with less features does not have a clear influence, using experience only about datasets with few features slightly worsens the results for the bigger datasets. This is because the model trained on bigger datasets also contains the knowledge about smaller feature sets which is sufficient for the datasets with less features. However, the model based on small datasets misses the knowledge about feature sets with a higher number of features resulting in a lower accuracy.

4.5. Influence of the choice of the meta-learning algorithm

So far, the results and discussion were based on using the ε-SVR as the meta-learning scheme. In this section, we study how the choice of the meta-learner influences the overall performance of the proposed predictive feature selection method. For this purpose, we compare ε-SVR with several other regression algorithms for use as a meta-learner. These algorithms include linear regression, Gaussian process, and a k-Nearest Neighbor (k-NN) algorithm. We used a sigmoid kernel for the Gaussian process and weighting based on the distance for k-NN. Also, k was optimized in the interval $[1;20]$ using a brute-force approach.

As performance measure, we used the Pearson Product Moment Correlation Coefficient. Thereby, we measure the correlation between the predicted $\hat{\mathcal{P}}$ and computed accuracies $\mathcal{P}$ of a classifier on a feature set:

$$\xi_{\mathcal{P},\hat{\mathcal{P}}} = \frac{E[(\mathcal{P}-\mu_{\mathcal{P}})(\hat{\mathcal{P}}-\mu_{\hat{\mathcal{P}}})]}{\sigma_{\mathcal{P}}\sigma_{\hat{\mathcal{P}}}}. \tag{3}$$

where $\mu_{\mathcal{P}}$ and $\mu_{\hat{\mathcal{P}}}$ represent the mean of the computed and predicted accuracies respectively, and $\sigma_{\mathcal{P}}$, $\sigma_{\hat{\mathcal{P}}}$ represent the corresponding standard deviations. This measure returns values in the interval $[-1;1]$. A value of 1 (-1) indicates a perfect positive (negative) relationship. If the correlation is 0, the two input variables are independent.

We selected this measure because not only the numerical difference between the predicted and computed accuracies is relevant, but also the ordering of the feature sets is important. A perfect correlation ($\xi_{\mathcal{P},\hat{\mathcal{P}}}=1$) would indicate a perfect ordering of the feature sets according to the predicted quality. Consequently, the actually best feature sets are selected for evaluation. Furthermore, with a perfect correlation, the evaluation step would be needless. No correlation ($\xi_{\mathcal{P},\hat{\mathcal{P}}}=0$) would indicate that the predictions made by the regression model are not better than random guessing.

For calculating the correlation of a meta-learner, we predicted the quality of at most 511 random feature subsets for each of the bigger datasets. The prediction was done by using a regression model trained on at most 511 random feature subsets of each remaining dataset.

Table 1 shows the results for the four considered regression algorithms. While ε-SVR achieved the highest correlation and linear regression a comparable one, the correlation of Gaussian process and k-NN are lower. We therefore conclude that ε-SVR is a suitable regression algorithm for this task.

Additionally, we also applied the different regression learners within the actual feature selection process. Fig. 11 shows the final accuracies achieved by the different regression learners for an evaluation rate of 0.1. As expected, the results of ε-SVR and linear regression are comparable while the performance of the two other algorithms is lower.

5. Conclusion

In this paper, we presented an approach for decreasing the computation time of wrapper based feature selection. We chose

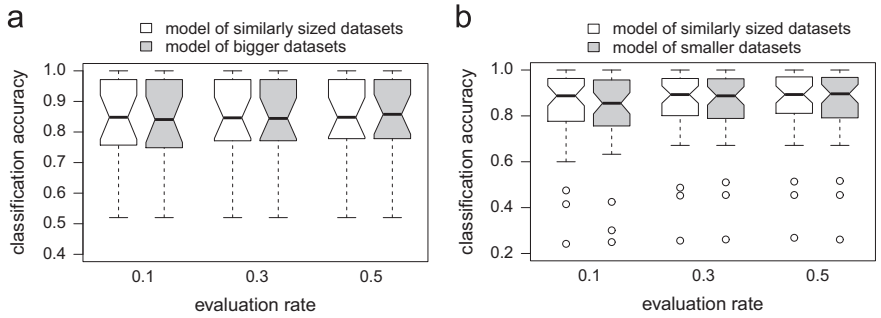

Fig. 10. Results of the presented method applied to the (a) smaller and (b) bigger datasets using the regression model trained on the other set of datasets.

Table 1
The Pearson Product Moment Correlation Coefficient ($\xi_{\mathcal{P},\hat{\mathcal{P}}}$) achieved by different regression algorithms.

Meta-learner	$\xi_{\mathcal{P},\hat{\mathcal{P}}}$
ε-SVR	0.801
Linear regression	0.783
Gaussian process	0.727
k-NN	0.723

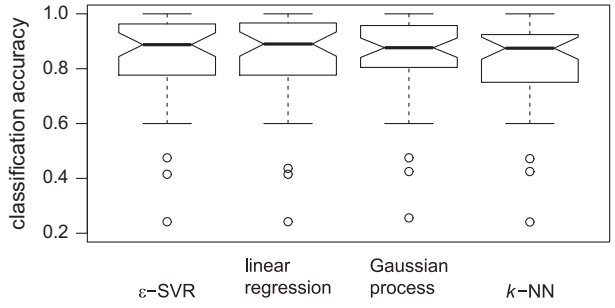

Fig. 11. Boxplot of the accuracies achieved by using four different regression algorithms as meta-learners ($\eta = 0.1$).

forward selection as the feature selection algorithm and Support Vector Machine as the target classifier to demonstrate our approach. The number of required classifier evaluations is reduced by predicting the accuracy of a candidate feature subset in a preliminary step and actually evaluating only feature subsets with the highest predicted quality.

We evaluated the approach on several real-world datasets from different domains. The results were compared to the brute-force approach and traditional *forward selection*. The presented approach achieved a comparable accuracy without a statistical difference whereas the run-time was reduced significantly. Additionally, the evaluation showed that the presented method is able to achieve statistically significant higher accuracy values than the state-of-the-art filter approaches MRMR and FCBF, as well as an embedded approach SCAD-SVM.

The method presented in this paper is, in principal, applicable to any wrapper method, such as backward elimination, random search, and brute-force search. While in random search, each candidate is only evaluated if the estimation reaches a certain threshold, a fixed fraction of all possible subsets might be evaluated based on the estimations in a brute-force approach. Furthermore, unlike embedded feature selection approaches, our method is independent of the choice of the target classifier and can be used in combination with other classifiers such as neural networks or nearest-neighbor classifier.

Conflict of interest statement

None declared.

References

[1] S. Ali, K.A. Smith, On learning algorithm selection for classification, Appl. Soft Comput. 6 (January) (2006) 119–138.
[2] A. Asuncion, D. Newman, UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, ⟨http://www.ics.uci.edu/∼mlearn/MLRepository.html⟩, 2007.
[3] N. Becker, W. Werft, G. Toedt, P. Lichter, A. Benner, PenalizedSVM: a R-package for feature selection SVM classification, Bioinformatics 25 (13) (2009) 1711–1712.
[4] H. Bensusan, C. Giraud-Carrier, 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, 2000, pp. 29–46.
[5] H. Bensusan, C. Giraud-Carrier, C. Kennedy, A higher-order approach to meta-learning, in: Proceedings of the ECML'2000 Workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination, June 2000, pp. 109–117.
[6] H. Bensusan, C.G. Giraud-Carrier, Discovering task neighbourhoods through landmark learning performances, in: PKDD'00: Proceedings of the Fourth European Conference on Principles of Data Mining and Knowledge Discovery, Springer Berlin/Heidelberg, 2000, pp. 325–330.
[7] H. Bensusan, A. Kalousis, Estimating the predictive accuracy of a classifier, in: L. De Raedt, P. Flach (Eds.), Machine Learning: ECML 2001, Lecture Notes in Computer Science, vol. 2167, Springer, Berlin/Heidelberg, 2001, pp. 25–36.
[8] A.L. Blum, P. Langley, Selection of relevant features and examples in machine learning, Artif. Intell. (Special Issue on Relevance) 97 (1997) 235–271.
[9] P. Bradley, O.L. Mangasarian, Feature selection via concave minimization and support vector machines, in: International Conference on Machine Learning (ICML), Morgan Kaufmann, 1998, pp. 82–90.
[10] P.B. Brazdil, C. Soares, Zoomed ranking: selection of classification algorithms based on relevant performance information, in: Proceedings of Principles of Data Mining and Knowledge Discovery, Fourth European Conference, 2000, pp. 126–135.
[11] P.B. Brazdil, C. Soares, J.P. da Costa, Ranking learning algorithms: using IBL and meta-learning on accuracy and time results, Mach. Learn. 50 (3) (2003) 251–277.
[12] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[13] L. Breiman, J. Friedman, R. Olshen, C. Stone, Classification and Regression Trees, Wadsworth, 1984.
[14] C.J. Burges, A tutorial on support vector machines for pattern recognition, Data Mining Knowl. Discovery 2 (1998) 121–167.
[15] C. Castiello, G. Castellano, A.M. Fanelli, Meta-data: characterization of input features for meta-learning, in: V. Torra, Y. Narukawa, S. Miyamoto (Eds.), Modeling Decisions for Artificial Intelligence, Lecture Notes in Computer Science, vol. 3558, Springer, Berlin/Heidelberg, 2005, pp. 295–304.
[16] C.-C. Chang, C.J. Lin, LIBSVM: a library for support vector machines, Software available at ⟨http://www.csie.ntu.edu.tw/∼cjlin/libsvm⟩, 2001.
[17] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[18] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[19] R. Engels, C. Theusinger, Using a data metric for preprocessing advice for data mining applications, in: Proceedings of the European Conference on Artificial Intelligence (ECAI-98), John Wiley & Sons, 1998, pp. 430–434.
[20] J. Gama, P.B. Brazdil, Characterization of classification algorithms, in: C. Pinto-Ferreira, N. Mamede (Eds.), Progress in Artificial Intelligence. Lecture Notes in Computer Science, vol. 990, Springer, Berlin/Heidelberg, 1995, pp. 189–200.
[21] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, J. Mach. Learn. Res. March (3) (2003) 1157–1182.

[22] C.-W. Hsu, C.-J. Lin, A comparison of methods for multi-class support vector machines, IEEE Trans. Neural Networks 13 (2002) 415–425.
[23] R.D. King, C. Feng, A. Sutherland, Statlog: comparison of classification algorithms on large real-world problems, Appl. Artif. Intell. 9 (1995) 289–333.
[24] R. Kohavi, G.H. John, Wrappers for feature subset selection, Artif. Intell. (Special Issue on Relevance) 97 (December) (1997) 273–324.
[25] C. Köpf, C. Taylor, J. Keller, 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, 2000.
[26] K.Z. Mao, Fast orthogonal forward selection algorithm for feature subset selection, IEEE Trans. Neural Networks 13 (5) (2002) 1218–1224.
[27] D. Michie, D.J. Spiegelhalter, C.C. Taylor, Machine Learning Neural and Statistical Classification, Ellis Horwood, 1994.
[28] H. Peng, F. Long, C. Ding, Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy, IEEE Trans. Pattern Anal. Mach. Intell. 27 (8) (2005) 1226–1238.
[29] Y. Peng, P. Flach, C. Soares, P.B. Brazdil, Improved dataset characterisation for meta-learning, in: S. Lange, K. Satoh, C. Smith (Eds.), Discovery Science. Lecture Notes in Computer Science, vol. 2534, Springer, Berlin/Heidelberg, 2002, pp. 193–208.
[30] B. Pfahringer, H. Bensusan, C. Giraud-Carrier, Meta-learning by landmarking various learning algorithms, in: Proceedings of the Seventeenth International Conference on Machine Learning, Morgan Kaufmann, 2000, pp. 743–750.
[31] M. Reif, A comprehensive dataset for evaluating approaches of various meta-learning tasks, in: First International Conference on Pattern Recognition and Methods (ICPRAM), SciPress, February 2012.
[32] M. Reif, F. Shafait, M. Goldstein, T. Breuel, A. Dengel, Automatic classifier selection for non-experts, Pattern Anal. Appl. http://dx.doi.org/10.1007/s10044-012-0280-z, in press.
[33] N. Sánchez-Maroño, A. Alonso-Betanzos, M. Tombilla-Sanromán, Filter methods for feature selection a comparative study, in: H. Yin, P. Tino, E. Corchado, W. Byrne, X. Yao (Eds.), Intelligent Data Engineering and Automated Learning —IDEAL 2007. Lecture Notes in Computer Science, vol. 4881, Springer, Berlin, Heidelberg, 2007, pp. 178–187, URL http://dx.doi.org/10.1007/978-3-540-77226-2_19.
[34] S. Segrera, J. Pinho, M. Moreno, Information-theoretic measures for meta-learning, in: E. Corchado, A. Abraham, W. Pedrycz (Eds.), Hybrid Artificial Intelligence Systems. Lecture Notes in Computer Science, vol. 5271, Springer, Berlin/Heidelberg, 2008, pp. 458–465.
[35] J.S. Simonoff, Analyzing Categorical Data. Springer Texts in Statistics, Springer, Berlin/Heidelberg, 2003 ⟨http://people.stern.nyu.edu/jsimonof/AnalCatData/⟩.
[36] A.J. Smola, B. Schölkopf, A tutorial on support vector regression, Stat. Comput. 14 (August (3)) (2004) 199–222.
[37] S.Y. Sohn, Meta analysis of classification algorithms for pattern recognition, IEEE Trans. Pattern Anal. Mach. Intell. 21 (11) (1999) 1137–1144.
[38] R. Vilalta, C. Giraud-Carrier, P.B. Brazdil, C. Soares, Using meta-learning to support data mining, Int. J. Comput. Sci. Appl. 1 (1) (2004) 31–45.
[39] P. Vlachos, StatLib Datasets Archive, Department of Statistics, Carnegie Mellon University, ⟨http://lib.stat.cmu.edu⟩, 1998.
[40] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, V. Vapnik, Feature selection for SVMs, Adv. Neural Inf. Process. Syst. 13 (2000) 668–674.
[41] D.H. Wolpert, The lack of a priori distinctions between learning algorithms, Neural Comput. 8 (7) (1996) 1341–1390.
[42] L. Yu, H. Liu, Efficient feature selection via analysis of relevance and redundancy, J. Mach. Learn. Res. 5 (2004) 1205–1224.
[43] H.H. Zhang, J. Ahn, X. Lin, Gene selection using support vector machines with nonconvex penalty, Bioinformatics 22 (2006) 88–95.

Matthias Reif received his diploma in Computer Science in 2006 from University of Technology Chemnitz, Germany. He is working as a Researcher at DFKI, Germany and also pursuing his Ph.D. from University of Kaiserslautern, Germany. His research interests include machine learning, pattern recognition, and meta-learning.

Faisal Shafait is working as an Assistant Professor at the University of Western Australia, Perth. He has co-authored over 80 publications in international peer-reviewed conferences and journals in the areas of machine learning, pattern recognition, and document image analysis. He is serving as an Editorial Board member of IJDAR and as a PC member of several international conferences.