Extracting Information from Support Vector Machines for Pattern-Based Classification

Madeleine Seeland
Technische Universität München
Institut für Informatik I12
Boltzmannstr. 3
85748 Garching, Germany
madeleine.seeland@in.tum.de

Andreas Maunz
Oncotest GmbH
Am Flughafen 12-14
79108 Freiburg, Germany
andreas@maunz.de

Andreas Karwath, Stefan Kramer
Johannes Gutenberg-Universität Mainz
Institut für Informatik
Staudingerweg 9
55128 Mainz, Germany
{karwath,kramer}@informatik.uni-mainz.de

ABSTRACT

Statistical machine learning algorithms building on patterns found by pattern mining algorithms have to cope with large solution sets and thus the high dimensionality of the feature space. Vice versa, pattern mining algorithms are frequently applied to irrelevant instances, thus causing noise in the output. Solution sets of pattern mining algorithms also typically grow with increasing input datasets. The paper proposes an approach to overcome these limitations. The approach extracts information from trained support vector machines, in particular their support vectors and their relevance according to their coefficients. It uses the support vectors along with their coefficients as input to pattern mining algorithms able to handle weighted instances. Our experiments in the domain of graph mining and molecular graphs show that the resulting models are not significantly less accurate than models trained on the full datasets, yet require only a fraction of the time using much smaller sets of patterns.

1. INTRODUCTION

Two of the most important families of classification methods for structured data (like graphs, trees, and sequences) are kernel-based and pattern-based methods. Kernel-based methods, on the one hand, are often superior in terms of predictivity, but frequently lack the possibility of interpretation and extraction of relevant features. Pattern-based methods, on the other hand, are often less accurate, but offer the advantage of explicit feature spaces, for instance, for the inspection of feature usage or feature weights. As pattern-based classification methods build upon the output of pattern-mining algorithms, they are heavily affected by the typically huge solution sets produced by those algorithms. Despite enormous progress in the area of condensed representations and compression methods for pattern sets (see, e.g., BBRCs [12] and KRIMP [23]), the output size may, in some cases, remain too large for practical application.

Kernel-based methods from the first family of methods, based on support vector machines (SVMs), are often considered as state-of-the-art classification methods in machine learning. An important advantage of SVMs is that their classification decision is based on a subset of the training examples, referred to as the support vectors. However, a drawback of SVMs is their black-box character. The generated non-linear models typically lack interpretability. Therefore, the topic of opening the black-box or making SVMs interpretable has received considerable attention in the literature, e.g., in areas such as credit evaluation, graph reconstruction and others [1, 11, 24]. Martens *et al.* attempted to mimic the behavior of SVM models and to add comprehensibility to them by extracting rules from the trained models [11]. This is accomplished by explicitly making use of key concepts of the SVM: the support vectors, and the observation that these are typically close to the decision boundary. The approach generates additional training examples close to the support vectors. The generated examples are then used with the training data, all provided with a class label by the trained SVM model, to train different decision tree algorithms that learn what the SVMs have learned. Another attempt to extract information from SVMs was made by Bakir *et al.* [1]. The authors trained an SVM regression model to represent the inverse mapping from the feature space to the input space, thus obtaining a pre-image function. This enables sampling of novel instances into the input space. Weston *et al.* [24] considered the task of learning dependencies between a general class of objects. Their approach referred to as Kernel Dependency Estimation (KDE) uses a kernel principal component analysis (PCA) to implicitly model (embed) correlations among both inputs and outputs. KDE decouples output correlations by first applying Kernel PCA over the outputs and then learning the mappings from the input space to dimension-reduced space by ridge regression. A pre-image calculation is required in order to recover the output in the original representation. Whereas pre-image methods are well-established for vectorial data, their usage for graph data seems limited so far. In particular, there is no well-known approach for deriving an explicit graph-based feature representation from a trained SVM with a graph kernel.

In the light of this, we address the following question in this paper: Can we make use of trained SVM models to obtain more compact pattern-based classification models? There are at least two possible ways of doing so: by analyzing a trained SVM model together with the training set, or by using the models as oracles to label instances [5]. In this paper, we study the former of the two approaches. As we would like to take advantage of any SVM with a graph-based kernel and all the information regarding the classification is actually contained in the graph, we do not necessarily assume that the feature space of the graph kernel corresponds to the pattern language of the graph miner.

We investigate this question and a possible solution in the context of graphs and, in particular, molecular graphs. Many graph kernels have been proposed in this domain, including the random walk graph kernel [8, 22], the optimal assignment kernel [7], the shortest-path graph kernel [2], the subtree pattern kernel [20], the neighborhood subgraph pairwise distance kernel [4] and the structural cluster kernel [19]. As mentioned above, a major drawback of kernel approaches is the lack of interpretability, as it may be difficult to figure out which features play an important role in classification. On the other hand, graph pattern-based approaches build graph classifiers based on different types of graph substructure features. The basic idea is to extract frequent substructures, local graph fragments, or cyclic patterns and trees and use them as descriptors to represent the graph data. A major problem with pattern mining is that it usually generates too many patterns on training data, many of which are redundant. When the input datasets attain considerable size, the mining process becomes computationally expensive or simply infeasible. Further, pattern mining algorithms are frequently applied to irrelevant instances. Thus, the interpretation of the results turns out to be a difficult task as the interesting patterns are blurred into the huge amount of outputted patterns.

The approach proposed in this paper aims to overcome some of the aforementioned limitations. More specifically, our approach is motivated by the question whether we can derive a small set of representative patterns from a set of relevant structures extracted from a given dataset such that the classifier trained on this pattern set still leads acceptable prediction performance. To this end we combine graph mining techniques with graph kernel based classifiers. To extract a set of relevant graph instances from a given input dataset we train an SVM on a recently proposed graph kernel, i.e., the structural cluster kernel (SCK). The basic idea of the SCK is to improve a base graph kernel by incorporating similarity information induced by a structural graph clustering algorithm [17, 18]. The extracted graph instances, i.e., the support vectors, and their according weights reflecting the relevance of each support vector are then used as input to a pattern mining algorithm that can cope with weighted instances. More specifically, we employ a modified version of the pattern mining algorithm Backbone Refinement Class Mining (BBRC) [12] that mines compact sets of subgraph descriptors from a set of weighted graphs. By considering only the support vectors for the mining process, i.e., the instances that are critical for classification, we expect that the resulting models achieve similar or at least not significantly worse accuracy than the model trained on the full dataset. Our experiments in the domain of molecular graphs indeed show that on most of our datasets, in particular on datasets comprising structurally more homogeneous graphs, the models trained on the support vectors are not significantly less accurate than models trained on the full datasets, yet require only a fraction of the time while using less patterns.

The paper is organized as follows: We present the problem definition in Section 2. Section 3 presents our approach to information extraction from SVMs for pattern-based classification along with necessary background information about the employed methods. In Section 4, we give an overview of the datasets and baseline methods used in the paper as well as a description of the experimental setup. Further, we present and discuss experimental results quantitatively and qualitatively. Finally, we conclude in Section 5.

2. PROBLEM DEFINITION

Our proposed approach investigates the question whether we can employ trained SVM models to obtain more compact pattern-based classification models. To do so, our approach extracts information from trained support vector machines, i.e., their support vectors and their relevance according to their coefficients (weights). It uses the support vectors along with their coefficients as input to pattern mining algorithms able to handle weighted instances. The motivation for this is the following: (a) Only the support vectors, i.e., the training instances relevant for the classification according to the trained SVM, are being subjected to the pattern mining. (b) These training instances enter the graph mining not with uniform weight, but a weight corresponding to their contribution to the classification according to the SVM's objective and loss function.

Formally, we frame the problem as follows. Consider the binary classification scenario, and a training dataset $D_{Trg} = \{(x_1, y_1), \ldots, (x_t, y_t)\}$, where $x_i \in X$ represent the data points and $y_i \in \{-1, 1\}$ their corresponding class labels. Further, let $D_{Tst} = \{x_{t+1}, \ldots, x_n\}$ denote the set of test instances. By training an SVM on the training data D_{Trg}, we obtain the support vectors $s = \{s_1, s_2, \ldots, s_k\} \subseteq D_{Trg}$. Given the set of support vectors s along with their corresponding weights $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$, the goal of our approach is to extract a small set of representative subgraph patterns $\{q_1, q_2, \ldots, q_m\}$ ($m \in \mathbb{N}$) correlated with the class labels from only the support vectors, such that the classifier trained on the reduced pattern set still yields acceptable, i.e., not significantly worse, classification accuracy.

3. METHOD

In this section, we first provide some background information on the employed methods before we present our approach to information extraction from SVMs for pattern-based classification in detail.

3.1 Structural Cluster Kernel

This section gives a short overview of the structural cluster kernel (SCK) proposed by Seeland *et al.* [19] that is used by our approach to extract a set of relevant graph instances from a given input dataset. The SCK incorporates similarities induced by a structural graph clustering algorithm [17, 18] to improve state-of-the-art graph kernels. The approach taken is based on the idea that graph similarity can not only be described by the similarity between the graphs themselves, but also by the similarity they possess with respect to their structural neighborhood. The main idea is to change the similarity metric of a base kernel so that the relative similarity between two graph objects is higher if the objects are in the same cluster. The kernel uses a combination of two similarity measures: (1) a base kernel that computes structural similarity between pairs of graphs and (2) a cluster based similarity measure that describes how close examples are to each other in terms of the similarities between the clusters they belong to. The similarity between two clusters is computed by taking the average of the similarities between the cluster instances. In this paper, the neighborhood subgraph pairwise distance kernel (NSPDK) [4] is used as base kernel to compute the pairwise similarities between graphs. The NSPDK is based on exact matching between pairs of small subgraphs. More precisely, the NSPDK decomposes a graph into all pairs of neighborhood subgraphs of radius r whose roots are at distance d in a given graph. The similarity between two graphs is defined in terms of the number of identical pairs of neighboring graphs of radius r at distance d between two graphs.

3.2 Backbone Refinement Class Mining

This section introduces a modified version of Backbone Refinement Class (BBRC) Mining [12] which is used in this work to mine compact sets of subgraph descriptors from a set of weighted graphs. BBRC Mining is an algorithmic approach to mining compact sets of subgraph descriptors in the search space of chemical structure graphs, creating compressed representations of chemical structure. It can be applied to class-labeled 2D graph databases and combines feature generation and feature selection into one step. The modified version employed in this work, allows to weight individual instances of the graph database according to their importance. BBRC mining creates a sparse selection from the search space of frequent and significant subtrees, based on a weighted minimum frequency, structural and statistical constraints. In the work presented here, weights (β) for individual instances can be positive real numbers and zero and are not normalized in any way. Furthermore, the minimum frequency constraint can be set to any positive real number, reflecting each instance's importance in the dataset and hence also of the induced patterns. It has very high compression potential, which has been shown theoretically [12] from the unweighted version of BBRCs. Empirical results confirmed the compression results in practice, while retaining good database coverage. Moreover, it has been shown that the structural constraints produce structurally diverse features with low co-occurrence rates. BBRC descriptors compare favorably to other compressed representations in the context of classification models. In classification tasks with either nearest-neighbor or SVM models, the accuracy of models based on BBRC descriptors was equal to models with the complete set of frequent and significant subtrees, but significantly better than that of other compressed representations. The algorithm performs substructure selection with regard to the endpoint under investigation, and calculates substructure associations to the endpoint in the form of p-values from a chi-squared distribution test. In the following, we recapitulate the basic concepts of BBRC in more detail.

First, we start with the definition of Backbone Refinement Classes. Undirected, labeled graphs are partially ordered via the refinement relation. Here, only acyclic graphs (paths and trees) are considered. Specifically, any path has a sequence, and sequences can be lexicographically ordered. Accordingly, any tree has a backbone, which is defined as the longest path with the lexicographically lowest sequence within the tree. An immediate tree refinement is an addition of a node and an edge to a tree such that it remains a tree, i.e., not possesses a cycle. We are considering the *Backbone Refinement Classes*, where the members of each such class are tree refinements of each other and share the same backbone. We denote the backbone refinement class relation by $\preceq_b$. Note that the classes are not disjoint for the same backbone (but they are across different ones). For example, in Figure 1, q_1 and q_3 are in different classes, but q_2 is in the respective classes of both q_1 and q_3.

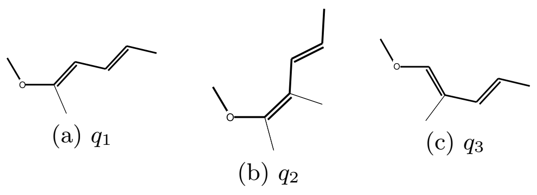


Figure 1: Three example trees with the same backbone (bold). Its sequence is 'c:c:c-C=C-O-C' (reflecting that the fragments include part of an aromatic ring). It also holds that $q_1 \preceq_b q_2$ and $q_3 \preceq_b q_2$, but neither $q_1 \preceq_b q_3$ nor $q_3 \preceq_b q_1$. Therefore, q_1 and q_3 are not in the same Backbone Refinement Class.

We also assume a categorical target class labeling function for the graphs in the database, enabling significance tests on trees by calculating their weighted occurrences in the database and therefore in the weighted target classes.

The problem of weighted BBRC Mining can be defined as follows. Given a graph database, corresponding weights, a user-defined minimum support and user-defined minimum χ^2 value, for all backbone refinement classes within the database, find the smallest (according to $\preceq_b$) of the most significant trees in each class that is *frequent* and *significant* with respect to their weighted occurrences in the target classes. The complexity of BBRC mining is upper-bounded by the complexity of regular tree mining [15]. However, running times are significantly lower for practical applications.

For significance testing, BBRC employs the χ^2 distribution test using weighted instances. Given a subgraph q, the original weights β, and I disjoint target classes G_i, whose weighted member graphs make up the database, we seek a $I \times 2$ contingency table that lists q's weighted support values per target class in the first column and the overall weighted distribution of target classes in the second column, as in Table 1.

	q	*all*
class 1	k_1	$\|G_1\|$
class 2	k_2	$\|G_2\|$
...	...	...
class I	k_I	$\|G_I\|$
Σ	k	$\|G\|$

Table 1: Contingency table for subgraph q.

These data serve to check whether q's weighted support values differ significantly from the overall weighted class distribution. The χ^2_d function is used for distribution testing, defined as

$$\chi^2_d(x, y) = \sum_{i=1}^{I} \frac{(k_i - E(k_i))^2}{E(k_i)}, \qquad (1)$$

where $k_i = \sum_{x_j \in G_i} cover(q, x_j)\beta_{x_j}$ with $cover(q, x_j) \in \{0, 1\}$ denoting whether q satisfies x_j (1) or not (0). Furthermore, $E(k_i) = \frac{|G_i|k}{|G|}$ is the expected value of the weighted k_i, calculates the sum of squares of deviations from the expected weighted support for all target classes G_i (where $|G_i|$ is given by $|G_i| = \sum_{x_j \in G_i} \beta_{x_j}$, i.e., the weighted occurrence of all instances for class G_i. Similarly, $|G|$ reflects the weighted occurrences of all instances in the database. The function value is then compared against the χ^2 distribution function to conduct a significance test with $I - 1$ degrees of freedom and obtain a p-value $p(q)$. It is possible to calculate an upper bound for the χ^2 values of refinements of a pattern [13], which can be used for antimonotonic pruning. Using a static, user-defined upper bound threshold is referred to as *static upper bound pruning*. To speed up the search, we may increase this threshold (*dynamic upper bound adjustment*). For any frequent subtree q, let $\chi^2(q, R)$ and $\chi^2_u(q, R)$ denote the χ^2 value for q and χ^2 upper bound for refinements of q, respectively. Let $u_{max}(q) = max\{\chi^2(p, R) \mid p \preceq_b q\}$. Then, if $u_{max}(q) > u$, u may be increased to $u_{max}(q)$, since we only search for the maximum class element.

3.3 ν-SVM

In this paper, we use a version of SVM referred to as ν-SVM [16] to build a classification model using the SCK. The original SVM formulations for classification use the parameter $C \in [0, inf)$ to apply a penalty to the optimization for points which were not correctly predicted. The ν-SVM was developed to automatically adjust the penalty parameter C in the original SVM formulation by an alternative parameter, $\nu \in [0, 1]$, which applies a slightly different penalty. ν has a much more concrete interpretation than C [16] representing an upper bound on the fraction of training samples which are errors and a lower bound on the fraction of samples which are support vectors. Thus, ν-SVM enables us to control the number of support vectors in the SVM trained with the SCK.

3.4 Graph Mining On Support Vectors

This section introduces our approach for extracting information from support vector machines for pattern-based classification. The approach extracts information from trained support vector machines, in particular their support vectors and their relevance according to their coefficients (weights). Both the support vectors and their corresponding weights are then used as input to the pattern mining algorithm BBRC (see Section 3.2 for a detailed description) that can handle weighted instances. BBRC reflects the weights in its distribution test expressing individual importance of the instances. The intuition behind this approach is the following. It is known that only the support vectors determine the final decision boundary of SVMs, whereas instances other than support vectors have no contribution to determine the decision boundary. The corresponding weights reflect the relative importance of a graph instance in discriminating the classes. The higher the weight of an instance, the more influential it is. Thus, by incorporating only the instances that are important for classification along with their weights in the pattern mining process, we expect that the resulting models yield similar predictive performance compared to the models trained on the full dataset.

Formally, let $D_{Trg} = \{(x_1, y_1), \dots, (x_t, y_t)\}$ denote the training data, where x_i represent the graph instances and $y_i \in \{-1, 1\}$ their class labels, respectively. Further, let $D_{Tst} = \{x_{t+1}, \dots, x_n\}$ represent the test instances. To extract relevant information from the given training data, our approach employs the structural cluster kernel [19] described in Section 3.1. The graph kernel is based on the assumption that graph similarity can not only determined by the similarity of the graphs alone, i.e., their structure, but also by the similarity of the graphs' structural neighborhood. Following this idea, the SCK incorporates similarity information induced by a structural graph clustering algorithm [17, 18] to improve a base kernel. In this paper, we use the NSPDK as base kernel. Our approach starts by constructing the SCK from the training data D_{Trg} and by training an SVM on the resulting kernel. Given the model trained on the SCK, we extract the graph instances whose corresponding SVM coefficients are nonzeros, i.e., the support vectors, along with their weights.

Formally, let $s = \{s_1, s_2, \dots, s_k\}$ denote the set of extracted support vectors associated with the training data D_{Trg} and let $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_k\}$ represent their corresponding weights. The weights are normalized such that the sum of all weights equals the number of training data, i.e., each weight α_j, $1 \le j \le k$, is normalized according to $\beta_j = \frac{|D_{Trg}|}{\sum_l \alpha_l}$. Next, we use the support vectors and the modified weights β as input for BBRC to derive a set of class-correlated subgraph patterns $\{q_1, q_2, \dots, q_m\}$. As described in Section 3.2, BBRC incorporates weights for instances in the χ^2 distribution test expressing individual importance of the instances. Given the resulting subgraph patterns, we construct a feature vector for each support vector representing the mined subgraph patterns. Formally, we represent each instance from the set of support vectors, s_i, by a binary feature vector $f_{s_i} = [f^1_{s_i}, f^2_{s_i}, \dots, f^m_{s_i}]$ corresponding to the set of mined subgraph patterns $\{q_1, q_2, \dots, q_m\}$. Each element $j \in \{1, \dots, m\}$ in the feature vector f_{s_i} indicates the presence and absence of the corresponding subgraph pattern g_j in the graph object s_i. Next, we train a linear SVM on the support vectors using the feature vectors constructed from the derived subgraph patterns.

Given a test set D_{Tst}, we match the subgraph patterns $\{q_1, q_2, \dots, q_m\}$ obtained by performing graph mining on the support vectors back onto the test instances. For each test instance we create a feature vector indicating the occurrence of each subgraph pattern in the test instance. The selected features are then evaluated by the accuracy of classification.

To summarize, our approach performs the following steps:

1. Construct the SCK on the training data D_{Trg} and train an SVM with the SCK.
2. Extract the support vectors $s = \{s_1, s_2, \dots, s_k\}$ associated with D_{Trg} and their corresponding weights $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_k\}$ from the trained SVM.
3. Normalize each weight α_j, $1 \le j \le k$, according to the formula $\beta_j = \frac{|D_{Trg}|}{\sum_l \alpha_l}$ and use the extracted support vectors s and their weights β as input for BBRC to derive a set of subgraph patterns $\{q_1, \dots, q_m\}$.
4. For each support vector s_i, create a feature vector $f_{s_i} = [f^1_{s_i}, f^2_{s_i}, \dots, f^m_{s_i}]$ corresponding to the set of derived subgraph patterns $\{q_1, \dots, q_m\}$.
5. Use the features vectors associated with the support vectors to train a linear SVM.

6. For each test instance $x_j \in D_{Tst}$:

   (a) Match the subgraph patterns $\{q_1, \ldots, q_m\}$ back onto x_j.

   (b) Create a feature vector $f_{x_j} = [f^1_{x_j}, f^2_{x_j}, \ldots, f^m_{x_j}]$ corresponding to the matched subgraph patterns.

   (c) Classify the test instance x_j according to the prediction of the model built in step 5.

Figure 2 illustrates the steps of our approach in a flowchart.

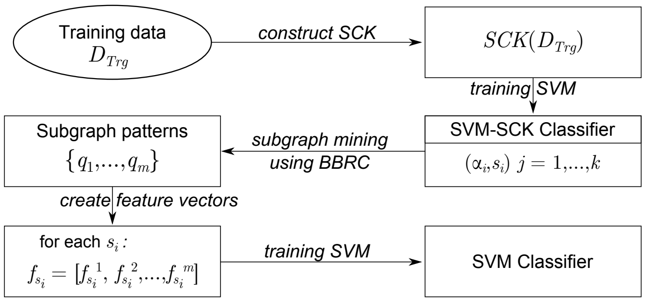


Figure 2: Flowchart of our approach for extracting information from support vector machines.

4. EXPERIMENTAL RESULTS

In this section, we study the performance of our proposed method. The goal is to investigate whether reducing the entire training data to a set of k support vectors that have the highest contribution to classification still yields acceptable classification performance. The section presents the baseline methods, the datasets, the experimental setup and the results.

4.1 Baseline Methods

To investigate the effectiveness of our approach in terms of its ability to reduce the training set size while maintaining the generalization performance, we compared our approach against a method that calculates a set of subgraph patterns from the entire training data D_{Trg}. The derived subgraph patterns are then used as features to build a classification model using a linear SVM. Similar to our proposed approach, the method applies BBRC for computing the subgraph patterns.

We further compared our approach against a method that builds a classification model on a reduced feature set obtained by conducting feature selection on the training data. We used the following two approaches for feature selection. The first approach to feature selection, referred to as $\mathrm{FS}_{\text{top-k}}$, produces a ranked list of attributes using a linear SVM as attribute evaluator and specifies a number of top-ranked attributes to retain. In our experiments, the number of top-ranked attributes corresponds to the number of features obtained by our approach by using 60% of the support vectors for model building. The second approach, referred to as FS, also produces a ranked list using a linear SVM as attribute evaluator. It then steps through this list evaluating each subsequently larger subset (i.e., top attribute, top two attributes etc.) using an SVM based subset evaluator. For both approaches, we train a linear SVM over the training data represented by the reduced feature set.

4.2 Datasets

We employed the chemical domain as our application area by using real datasets of molecular graphs. Table 2 provides an overview of the datasets. All datasets are associated with a classification endpoint, e.g., carcinogenicity.

Table 2: Overview of the datasets. n denotes the number of graphs and Tan. Sim. the mean pairwise Tanimoto similarity using ChemAxon's chemical fingerprints (default parameters).

Dataset	n	Tan. Sim.	Proportion positive class
Fontaine [6]	435	0.461	0.64
er_tox [21]	446	0.416	0.41
bloodbarr [10]	413	0.237	0.67
kazius [9]	4337	0.159	0.41
dhfr [21]	303	0.428	0.32
NCI_AIDS [3]	1000	0.305	0.42

4.3 Experimental Setup

In all experiments, classification was performed using SVM classifiers. To build a classification model using the SCK, we used ν-SVM as classifier [16] (see Section 3.3). We controlled the fraction of support vectors by choosing ν such that the number of support vectors covers a specific fraction of the training samples. In this paper, we require the number of support vectors to cover 50% and 60% of the training data, respectively. For each dataset, we selected three values for the minimum frequency (MF) parameter of BBRC. For *NCI_AIDS*, *dhfr* and *bloodbarr* we chose $MF = 6\%$, 8% and 10%, respectively. Due to the structural diversity of the *kazius* dataset, we decided to set smaller values for the MF, since higher values result in too few patterns. For the structural homogeneous datasets *er_tox* and *Fontaine*, we chose higher values for the MF, since for smaller values BBRC generates too many features resulting in an increased computational complexity. To build a classification model on the subgraph patterns, we used a linear SVM. The trade-off between training error and margin, C, was optimized by internal cross-validation selecting from $\{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$. The parameter resulting in the the highest accuracy was then used for building the final model. All other SVM parameters were left at their default values. Performance estimates were obtained using 15 times hold-out validation using the same data for training and testing for all comparison methods. According to Nadeau and Bengio [14], a 15 holdout run provides good power, in terms of the probability of rejecting the null hypothesis when it is false, with reasonable computational effort, whereas going beyond 15 gives little additional power and is probably not worth the computational effort. To quantify predictive accuracy, we chose the classification accuracy, which is a standard measure in classification settings. We tested for significant differences between the methods using the corrected resampled t-test [14] at the 5% significance level. We further investigated the number of computed subgraph patterns as well as the runtime for model building and prediction.

4.4 Results

Table 3 shows the detailed experimental results in terms of classification accuracy, number of computed subgraph patterns and runtime performance (time for model building and prediction) for the various methods on all datasets. We indi-

Table 3: Classification accuracies (ACC), number of attributes (#Feats) and runtime (in sec). MF denotes the minimum frequency parameter of BBRC.

Fontaine		MF=22%	MF=20%	MF=18%
D_{Trg}	ACC	91.26 ± 1.90	92.30 ± 1.96	92.34 ± 2.86
	#Feats	4864.3 ± 3486.3	4923.6 ± 3483.7	5868.6 ± 2952.2
	Time	25.5 ± 22.7	60.0 ± 69.9	21.9 ± 18.1
$SV_{60\%}$	ACC	90.59 ± 2.11	91.12 ± 1.87	91.40 ± 2.52
	#Feats	140.6 ± 119.0	237.3 ± 151.5	291.9 ± 169.5
	Time	2.8 ± 4.3	3.7 ± 5.5	4.5 ± 5.6
$SV_{50\%}$	ACC	89.14 ± 2.73	89.28 ± 2.66 ○	89.64 ± 3.16 ○
	#Feats	113.8 ± 120.3	143.7 ± 153.5	176.45± 159.3
	Time	4.0 ± 9.2	2.8 ± 5.2	4.7 ± 7.5
$FS_{top\text{-}k}$	ACC	86.26 ± 8.05 ○	90.26 ± 3.40	90.96 ± 2.57
	#Feats	140.6 ± 119.0	237.3 ± 151.5	291.9 ± 169.5
	Time	763.4 ± 871.8	1893.2 ± 3418.9	1135.4 ± 2086.8
FS	ACC	90.63 ± 1.72	91.08 ± 2.47	91.36 ± 1.84
	#Feats	65.7 ± 24.2	104.8 ± 45.3	144.8 ± 132.1
	Time	8771.4 ± 9873.8	9998.2 ± 13405.2	18068.5 ± 25696.9
er_tox		MF=12%	MF=10%	MF=8%
D_{Trg}	ACC	75.31 ± 2.62	75.26 ± 2.78	75.00 ± 2.75
	#Feats	7927.5 ± 2135.3	7998.0 ± 2171.3	8131.2 ± 1867.0
	Time	74.9 ± 31.1	75.0 ± 29.7	75.5 ± 32.1
$SV_{60\%}$	ACC	75.18 ± 2.68	75.18 ± 3.37	75.35 ± 3.87
	#Feats	5826.3 ± 2730.1	6796.1 ± 2823.1	7474.5 ± 3045.7
	Time	30.0 ± 18.9	31.9 ± 22.5	37.5 ± 23.1
$SV_{50\%}$	ACC	70.79 ± 5.49	71.14 ± 8.17	72.63 ± 3.88
	#Feats	2000.4 ± 3321.0	3473.6 ± 4487.6	3833.8 ± 5150.2
	Time	5.3 ± 13.0	7.6 ± 14.0	12.7 ± 26.4
$FS_{top\text{-}k}$	ACC	74.66 ± 2.39	74.76 ± 2.46	74.83 ± 2.89
	#Feats	5826.3 ± 2730.1	6796.1 ± 2823.1	7474.5 ± 3045.7
	Time	2056.1 ± 1557.4	2882.7 ± 3329.3	3646.1 ± 2924.7
FS	ACC	74.61 ± 1.22	74.91 ± 1.25	75.10 ± 1.99
	#Feats	501.0 ± 159.4	727.0 ± 161.0	971.0 ± 280.0
	Time	28460.2 ± 23104.2	30508.4 ± 28020.3	34460.2 ± 30104.2
bloodbarr		MF=10%	MF=8%	MF=6%
D_{Trg}	ACC	71.87 ± 2.88	70.83 ± 4.00	71.87 ± 2.64
	#Feats	73.4 ± 27.0	141.3 ± 66.8	321.2 ± 173.4
	Time	15.4 ± 9.2	18.3 ± 7.6	29.0 ± 67.2
$SV_{60\%}$	ACC	72.81 ± 2.92	72.96 ± 3.10	73.43 ± 2.77
	#Feats	35.8 ± 10.8	52.1 ± 17.5	142.6 ± 137.5
	Time	1.8 ± 3.9	1.4 ± 1.3	2.2 ± 3.9
$SV_{50\%}$	ACC	69.31 ± 3.57	69.22 ± 2.91	70.02 ± 3.34
	#Feats	21.4 ± 10.0	41.8 ± 21.6	87.6 ± 50.7
	Time	1.8 ± 4.4	1.2 ± 2.1	2.2 ± 5.0
$FS_{top\text{-}k}$	ACC	71.89 ± 2.96	70.35 ± 3.84 ○	71.73 ± 3.43
	#Feats	35.8 ± 10.8	52.1 ± 17.5	142.6 ± 137.5
	Time	108.07 ± 282.33	138.65 ± 147.33	140.16 ± 27.19
FS	ACC	72.29 ± 2.98	70.50 ± 3.13	71.73 ± 2.74
	#Feats	42.3 ± 20.9	51.7 ± 25.9	123.3 ± 51.9
	Time	113.7 ± 248.2	137.8 ± 147.5	388.2 ± 533.4
kazius		MF=6%	MF=4%	MF=3%
D_{Trg}	ACC	75.44 ± 1.10	75.82 ± 1.05	77.08 ± 1.00
	#Feats	169.5 ± 9.7	250.5 ± 15.4	350.2 ± 33.8
	Time	3122.5 ± 1456.9	4572.9 ± 1053.0	7016.5 ± 1155.50
$SV_{60\%}$	ACC	70.62 ± 0.80 ○	72.97 ± 1.42 ○	73.46 ± 1.25 ○
	#Feats	54.2 ± 12.1	145.5 ± 21.9	207.1 ± 42.7
	Time	70.3 ± 26.4	154.9 ± 64.5	227.4 ± 91.8
$SV_{50\%}$	ACC	67.56 ± 2.43 ○	67.85 ± 3.36 ○	68.78 ± 2.49 ○
	#Feats	29.3 ± 8.9	54.8 ± 15.6	95.0 ± 37.9
	Time	21.5 ± 13.5	40.5 ± 23.9	71.0 ± 42.3
$FS_{top\text{-}k}$	ACC	74.17 ± 1.00 ●	75.07 ± 1.23 ●	76.31 ± 0.86 ●
	#Feats	54.2 ± 12.1	145.5 ± 21.9	207.1 ± 42.7
	Time	6443.9 ± 5036.9	6870.5 ± 12745.4	14107.0 ± 29832.0
FS	ACC	75.05 ± 0.84 ●	76.04 ± 0.93 ●	76.73 ± 0.73 ●
	#Feats	151.0 ± 10.2	165.2 ± 19.6	265.3 ± 68.59
	Time	12266.9 ± 8266.9	19819.4 ± 15778.5	32730.9 ± 28721.1
dhfr		MF=10%	MF=8%	MF=6%
D_{Trg}	ACC	76.22 ± 2.57	77.16 ± 2.89	77.51 ± 3.63
	#Feats	607.4 ± 161.4	1054.1 ± 280.5	1596.9 ± 366.0
	Time	20.7 ± 7.1	38.5 ± 60.7	41.8 ±41.1
$SV_{60\%}$	ACC	76.82 ± 4.18	77.06 ± 3.39	77.16 ± 3.93
	#Feats	290.0 ± 152.4	362.0 ± 210.0	576.5 ± 315.1
	Time	3.7 ± 3.8	4.8 ± 4.4	5.7 ± 4.6
$SV_{50\%}$	ACC	75.97 ± 1.70	76.12 ± 3.08	76.31 ± 4.06
	#Feats	135.6 ± 85.3	213.5 ± 250.3	497.0 ± 377.0
	Time	2.5 ± 4.1	3.2 ± 4.5	3.0 ± 1.6
$FS_{top\text{-}k}$	ACC	74.73 ± 4.28 ○	76.02 ± 3.47	76.61 ± 3.43
	#Feats	290.0 ± 152.4	362.0 ± 210.0	576.5 ± 315.1
	Time	88.9 ± 91.2	144.4 ± 140.6	303.6 ± 322.6
FS	ACC	74.98 ± 4.40	76.57 ± 3.45	76.96 ± 4.14
	#Feats	127.7 ± 86.9	163.7 ± 55.3	288.3 ± 377.5
	Time	876.0 ± 894.7	1451.4 ± 1544.1	3153.1 ± 3468.6

●,○ statistically significant improvement, or degradation

NCI_AIDS		MF=10%	MF=8%	MF=6%
D_{Trg}	ACC	83.20 ± 2.44	83.63 ± 1.84	84.86 ± 1.88
	#Feats	321.1 ± 83.3	632.6 ± 142.0	1071.6± 224.9
	Time	399.0 ± 113.0	471.7 ± 145.7	210.8 ± 95.9
$SV_{60\%}$	ACC	82.31 ± 2.05	82.90 ± 1.61	84.20 ± 1.65
	#Feats	219.9 ± 72.4	389.4 ± 107.8	675.5 ± 222.4
	Time	15.9 ± 6.4	20.7 ± 5.8	19.4 ± 3.2
$SV_{50\%}$	ACC	77.94 ± 2.77 ○	78.35 ± 1.44 ○	79.22 ± 1.86 ○
	#Feats	72.2 ± 22.2	96.8 ± 37.6	154.3 ± 79.9
	Time	6.5 ± 5.1	7.3 ± 5.2	8.6 ± 5.6
$FS_{top\text{-}k}$	ACC	82.13 ± 1.94	82.42 ± 1.85	83.90 ± 1.83
	#Feats	219.9 ± 72.4	389.4 ± 107.8	675.5 ± 222.4
	Time	602.8 ± 1167.9	1010.1 ± 2155.3	1450.0 ± 2416.3
FS	ACC	82.26 ± 2.03	82.95 ± 1.89	84.25 ± 1.66
	#Feats	137.0 ± 61.1	238.1 ± 81.4	437.7 ± 98.9
	Time	4559.9 ± 12782.3	4408.7 ± 4050.0	13997.9 ± 13719.4

●,○ statistically significant improvement, or degradation

cate whether our approach using both 50% and 60% SVs is significantly better or worse than the method employing all the training data at $p < 0.05$ using the corrected resampled t-test. Further, in the same table we report whether the feature selection approaches, $FS_{top\text{-}k}$ and FS, are significantly better or worse than our approach using 60% SVs.

The results show that our approach achieves the best performances in terms of prediction accuracy on the datasets comprising structurally more homogeneous graphs, i.e., on all datasets except for *kazius*. On these datasets using 60% of the training data as support vectors our approach yields similar predictive performance and at the same time employs less features and requires only a fraction of the time compared to the approach that uses the entire training data to build a model. On the other hand, the results on the *kazius* toxicity dataset show that the models trained on the support vectors are significantly less accurate than models trained on the full datasets. Primarily, we explain this negative effect as a result of the structurally heterogeneity of the examples. On this dataset our approach yet generates a much smaller pattern set and at the same time requires less time. However, the patterns generated from only the support vectors are not sufficient for classification, indicated by significantly worse classification accuracies. Reducing the fraction of support vectors to 50% results in a decrease in performance with respect to prediction accuracy. However, on the *dhfr*, *bloodbarr* and *er_tox* datasets and on *Fontaine* for MF=22% the performance differences with respect to the approach using the full training data for graph mining are not statistically significant. For better illustration, Figure 3 shows the relationship between the number of features and the prediction for our approach with the number of support vectors covering 60% of the training data and the version employing the complete training data.

Comparing our approach using 60% SVs to both feature selection approaches, $FS_{top\text{-}k}$ and FS, we observe that on all datasets except for *kazius* our approach yields similar or even better predictive performance. At the same time, our approach requires far less time for training and testing than $FS_{top\text{-}k}$, while employing the same number of features. Even though the approach named FS produces less features than our approach, it needs far more time for model building and prediction (up to factors of thousands). To summarize, our experimental results demonstrate that the classification models resulting from our approach are on most datasets, more specifically on datasets comprising structurally more homogeneous graphs, not significantly less accurate than the models trained on the full datasets.

5. CONCLUSION

In the work presented here, we proposed an approach for extracting information from support vector machines for pattern-based classification. Our approach extracts information from trained support vector machines, in particular their support vectors and their relevance according to their coefficients. It uses the support vectors along with their coefficients as input to pattern mining algorithms able to handle weighted instances. We evaluated our approach on several real-world datasets of molecular graphs and compared it against a method that builds a classification model on a set of subgraph patterns derived from the full datasets. The results show that the models resulting from our approach are on most datasets, i.e., on datasets containing structurally similar molecular graphs, not significantly less accurate than models trained on the full datasets. At the same time the resulting models tend to be better interpretable, due to the smaller set of patterns generated by the mining process. Compared to using the full dataset in the mining process, our proposed approach allows an analysis, which is up to an order of magnitude faster using often a substantially smaller number of features. This enables feasible parameter and feature optimization for given problems using tens of thousands of experiments in a fraction of the original time required.

6. REFERENCES

[1] G. H. Bakir, A. Zien, and K. Tsuda. Learning to find graph pre-images. In *Pattern Recognition: Proc. 26th DAGM Symposium*, DAGM'04, pages 253–261, 2004.

[2] K. M. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. In *Proc. Fifth IEEE International Conference on Data Mining*, ICDM'05, pages 74–81, 2005.

[3] J. M. Collins. The DTP AIDS Antiviral Screen Program 1999, `http://dtp.nci.nih.gov/docs/aids/aidsdata.html`.

[4] F. Costa and K. De Grave. Fast neighborhood subgraph pairwise distance kernel. In *Proc. 26th International Conference on Machine Learning*, ICML'10, pages 255–262, 2010.

[5] M. Craven and J. W. Shavlik. Using sampling and queries to extract rules from trained neural networks. In *Proc. Eleventh International Conference on Machine Learning*, ICML'94, pages 37–45, 1994.

[6] F. Fontaine, M. Pastor, I. Zamora, and F. Sanz. Anchor-GRIND: Filling the gap between standard 3D QSAR and the GRid-INdependent descriptors. *J. Med. Chem.*, 48(7):2687–2694, 2005.

[7] H. Fröhlich, J. K. Wegner, F. Sieker, and A. Zell. Optimal assignment kernels for attributed molecular graphs. In *Proc. 22nd International Conference on Machine learning*, ICML'05, pages 225–232, New York, NY, USA, 2005. ACM.

[8] T. Gärtner, P. A. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Proc. 16th Annual Conference on Computational Learning Theory*, COLT'03, pages 129–143, 2003.

[9] J. Kazius, R. McGuire, and R. Bursi. Derivation and validation of toxicophores for mutagenicity prediction. *J. Med. Chem.*, 48(1):312–320, 2005.

[10] H. Li, C. W. Yap, C. Y. Ung, Y. Xue, Z. W. Cao, and Y. Z. Chen. Effect of selection of molecular descriptors on the prediction of blood-brain barrier penetrating and nonpenetrating agents by statistical learning methods. *J. Chem. Inf. Model.*, 45(5):1376–1384, 2005.

[11] D. Martens, B. Baesens, and T. Van Gestel. Decompositional rule extraction from support vector machines by active learning. *IEEE Trans. on Knowl. and Data Eng.*, 21(2):178–191, 2009.

[12] A. Maunz, C. Helma, and S. Kramer. Efficient mining for structurally diverse subgraph patterns in large molecular databases. *Mach. Learn.*, 83(2):193–218, 2011.

[13] S. Morishita and J. Sese. Transversing itemset lattices with statistical metric pruning. In *Proc. 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '00, pages 226–236, New York, NY, USA, 2000. ACM.

[14] C. Nadeau and Y. Bengio. Inference for the generalization error. *Mach. Learn.*, 52(3):239–281, 2003.

[15] S. Nijssen and J. N. Kok. The gaston tool for frequent subgraph mining. *Electron. Notes Theor. Comput. Sci.*, 127(1):77–87, Mar. 2005.

[16] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Comput.*, 12(5):1207–1245, May 2000.

[17] M. Seeland, S. A. Berger, A. Stamatakis, and S. Kramer. Parallel structural graph clustering. In *Proc. 2011 European Conference on Machine Learning and Knowledge Discovery in Databases*, ECML PKDD'11, pages 256–272, 2011.

[18] M. Seeland, T. Girschick, F. Buchwald, and S. Kramer. Online structural graph clustering using frequent subgraph mining. In *Proc. 2010 European Conference on Machine Learning and Knowledge Discovery in Databases*, ECML PKDD'10, pages 213–228, 2010.

[19] M. Seeland, A. Karwath, and S. Kramer. A structural cluster kernel for learning on graphs. In *Proc. 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'12, pages 516–524, New York, NY, USA, 2012. ACM.

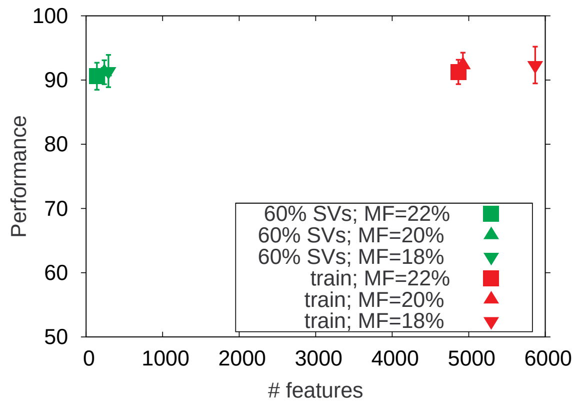
[20] N. Shervashidze and K. M. Borgwardt. Fast subtree kernels on graphs. In *Advances in Neural Information Processing Systems 22*, NIPS'09, pages 1660–1668, Red Hook, NY, USA, 2009. Curran.

[21] J. J. Sutherland, L. A. O Brien, and D. F. Weaver. Spline-fitting with a genetic algorithm: A method for developing classification structure-activity relationships. *J. Chem. Inf. Comput. Sci.*, 43(6):1906–1915, 2003.

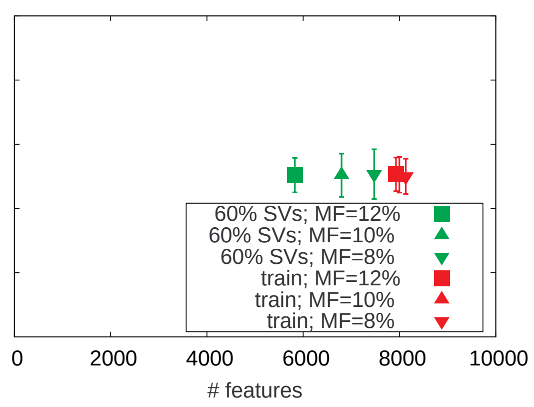
[22] S. V. N. Vishwanathan, K. M. Borgwardt, and N. N. Schraudolph. Fast computation of graph kernels. In *Proc. 20th Annual Conference on Neural Information Processing Systems*, NIPS'06, pages 1449–1456, 2006.

[23] J. Vreeken, M. Leeuwen, and A. Siebes. KRIMP: Mining itemsets that compress. *Data Min. Knowl. Discov.*, 23(1):169–214, 2011.

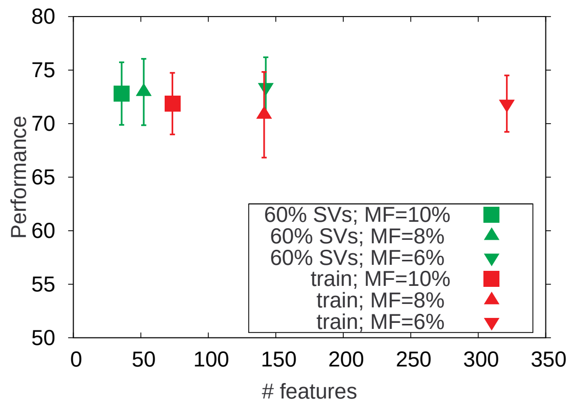
[24] J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik. Kernel dependency estimation. In *Advances in Neural Information Processing Systems*, NIPS'02, pages 873–880, 2002.



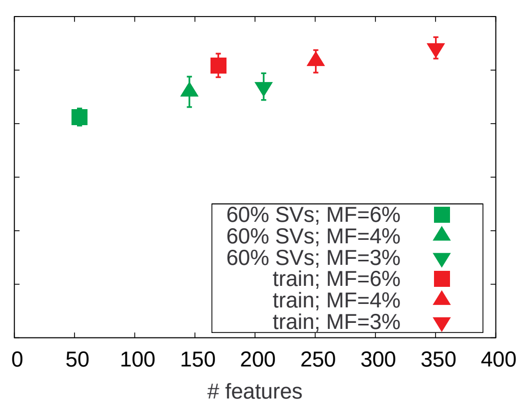
(a) Fontaine



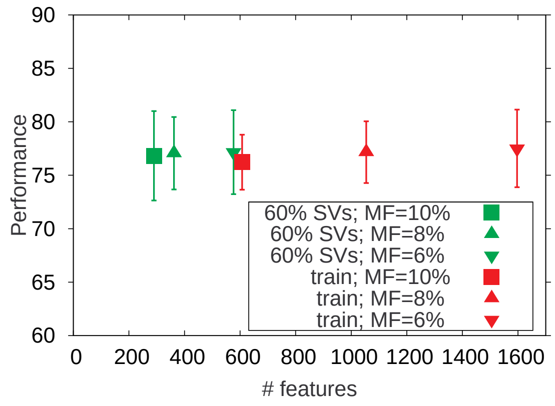
(b) er_tox



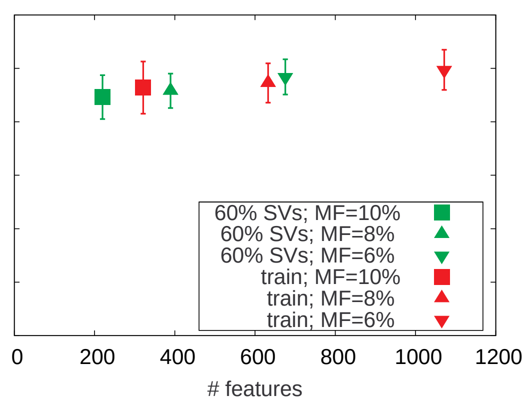
(c) bloodbarr



(d) kazius



(e) dhfr



(f) NCI_AIDS

Figure 3: Relationship between the number of patterns (features) generated by BBRC and the predictive performance of the compared methods on all benchmark datasets. In each figure "train" denotes the method using the full dataset as input for pattern mining, whereas "60% SVs" represents the approach incorporating only the support vectors and their corresponding weights into the mining process.