Fast, Effective Molecular Feature Mining by Local Optimization

Albrecht Zimmermann[1], Björn Bringmann[1], and Ulrich Rückert[2]

[1] Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Leuven, Belgium
{albrecht.zimmermann,bjoern.bringmann}@cs.kuleuven.be
[2] UC Berkeley, EECS Department, 750 Sutardja Dai Hall #1776, Berkeley, CA 94720-1776, USA
rueckert@eecs.berkeley.edu

Abstract. In structure-activity-relationships (SAR) one aims at finding classifiers that predict the biological or chemical activity of a compound from its molecular graph. Many approaches to SAR use sets of binary substructure features, which test for the occurrence of certain substructures in the molecular graph. As an alternative to enumerating very large sets of frequent patterns, numerous pattern set mining and pattern set selection techniques have been proposed. Existing approaches can be broadly classified into those that focus on minimizing correspondences, that is, the number of pairs of training instances from different classes with identical encodings and those that focus on maximizing the number of equivalence classes, that is, unique encodings in the training data. In this paper we evaluate a number of techniques to investigate which criterion is a better indicator of predictive accuracy. We find that minimizing correspondences is a necessary but not sufficient condition for good predictive accuracy, that equivalence classes are a better indicator of success and that it is important to have a good match between training set and pattern set size. Based on these results we propose a new, improved algorithm which performs local minimization of correspondences, yet evaluates the effect of patterns on equivalence classes globally. Empirical experiments demonstrate its efficacy and its superior run time behavior.

1 Introduction

The field of *structure-activity relationships* deals with the problem of predicting the biological or chemical activity of molecules. For example, a researcher might want to learn a model predicting whether or not a particular compound inhibits tumor growth. Such a model could then be used to reduce the amount of in-vitro experimentation. Since the molecular structure of a molecule can be easily encoded by a labeled graph, structure-activity learning problems have been a fertile field for structured data mining. Typically, data mining in this context starts from a data set of molecular compounds that fall into two activity classes, e.g. inhibiting cancer growth or not inhibiting it. Then, a first mining step generates a set of patterns. The patterns in this set are subsequences, subtrees, or

subgraphs of the molecular graphs in the database. This means that one can use the patterns as description features for classification. In this representation a molecular graph is re-encoded as a bit-vector where each bit indicates presence or absence of a specific pattern. Finally, this propositionalized data representation can then be handled by established machine learning techniques such as *Support Vector Machines* (SVM) to learn a classifier.[1]

The traditionally first approaches to the feature generation problem involved general substructure mining tools. For instance, some systems just generate all patterns whose occurrence frequency exceeds some predefined threshold. This was motivated by the use of *fingerprints* in bio-chemistry. In more recent developments, a variety of techniques [10, 14, 8, 2, 4, 16, 1, 5] have been proposed to avoid the unnecessarily large feature sets and long runtimes of these early approaches. Newer approaches usually either post-process a set of patterns to select a relatively small subset or iteratively refine a candidate pattern set so that some quality measure is optimized.

Finding smaller pattern sets is motivated by the observation that popular similarity measures fail to work well, if the training instances are represented by many redundant features that check for highly similar substructures.

There is, however, less agreement on a second question connected to finding good pattern sets for the encoding of molecules: Is it more important that

1. instances belonging to different classes are encoded differently from each other *or that*
2. instances are generally encoded differently from each other, no matter the class label?

The approach whose selection strategy is driven strongest by the former option is the FCORK technique introduced in [16] while the post-processing techniques proposed in [10] and [2] mainly focus on the latter one. Other techniques consider the two extremes to differing degrees. More generally, it is not very well understood which quality measures a pattern set should optimize in order to lead to classifiers with heigh predictive accuracy.

Our contribution to deciding this question in this paper is two-fold:

i We describe and compare several state-of-the-art approaches in Section 3 and evaluate the generated feature sets according to various quality measures in Section 4. In particular, we investigate whether the number of correspondences, a class-dependent measure, or the number of equivalence classes, a class-agnostic measure, are better indicators of good prediction accuracy.

ii Based on the obtained insights (and the identification of the currently most successful technique), we propose a new iterative mining technique in Section 5 and show that it improves on existing techniques in a variety of ways in Section 6.

The quality measures used in the paper to rate pattern sets are introduced in Section 2 and we present our conclusions in Section 7.

[1] Some approaches to structural prediction [7, 17, 1] integrate the mining step and learning step.

2 Quality Measures for Pattern Sets

Let us start by introducing the notions and quality measures for pattern sets used in the rest of the paper. Given a data set $\mathbb{D}$, a pattern language $\mathcal{L}$ and a pattern constraint c, the result of a constrained mining operation is a *theory* $\mathcal{T}h(\mathbb{D}, \mathcal{L}, c)$, namely a set of patterns satisfying the constraint(s) c (cf. [11]). This is e.g. the usual approach in *frequent* pattern mining. If the pattern constraint employed is a measure ϕ and the goal is to mine the top-k patterns according to this measure, the resulting theory is denoted by $\mathcal{T}h_k(\mathbb{D}, \mathcal{L}, \phi)$.

Let each pattern p be associated with a function $p : \mathbb{D} \mapsto \{true, false\}$. We define $p(t) = true$ if p matches t, and $p(t) = false$ otherwise. Associating a pattern with this boolean function does allow us to consider each pattern as a binary feature. In addition, we assume a binary class labeling function $c : \mathbb{D} \mapsto \{+, -\}$. Given a pattern set $\mathbb{S} \subseteq \mathcal{L}$, we define an equivalence relation $\sim_{\mathbb{S}}$ on $\mathbb{D}$ as:

$$\sim_{\mathbb{S}} \equiv \forall t_i, t_j \in \mathbb{D} : t_i \sim_{\mathbb{S}} t_j \Leftrightarrow \forall p \in \mathbb{S} : p(t_i) = p(t_j)$$

Thus, two data instances are considered to be equivalent under $\mathbb{S}$ if they share exactly the same patterns. Using the equivalence relation $\sim_{\mathbb{S}}$, an *equivalence class* or *block* is defined in the following way:

$$[t] =: \{t' \in \mathbb{D} : t' \sim_{\mathbb{S}} t\}$$

The *partition* or *quotient set* of $\mathbb{D}$ over $\mathbb{S}$ finally, is defined as:

$$\mathbb{P} = \mathbb{D}/\sim_{\mathbb{S}} = \{[t] : t \in \mathbb{D}\}$$

The partition thus induced by a pattern (and therefore feature) set corresponds to the number of different bit-strings that can be presented to a machine learning technique attempting to build a classifier based on them. Instances assigned to the same block are effectively indistinguishable and will therefore be classified in the same way. Some systems therefore maximize the number of equivalence classes in the pattern sets:

$$\mathrm{eq}_{\mathbb{D}}(\mathbb{S}) = \mid \{\mathbb{D}/\sim_{\mathbb{S}}\} \mid$$

While this might be slightly worrying in case of large blocks consisting of only one class (lump judgments can be problematic to generalize), it is clearly counterproductive if both classes are present in a block since this introduces unavoidable errors. The authors of [16] define the concept of *correspondence*: Two data instances t_1, t_2 form a *correspondence* under the equivalence relation $\sim_{\mathbb{S}}$ iff $c(t_1) \neq c(t_2) \wedge t_1 \sim_{\mathbb{S}} t_2$. Here, $c(t)$ denotes the class label of a training instance. With this, the number of correspondences is:

$$\mathrm{corr}_{\mathbb{D}}(\mathbb{S}) = \sum_{\mathbb{B}_i \in \mathbb{D}/\sim_{\mathbb{S}}} \mid \{t \in \mathbb{B}_i : c(t) = +\} \mid \cdot \mid \{t \in \mathbb{B}_i : c(t) = -\} \mid$$

A pattern set which minimizes the number of correspondences provides more information to the learning algorithm, because it encodes the specific properties

better that make each example differ from the other ones. On the other hand, this might lead to overfitting or large pattern sets. To overcome this, the authors of [10] use joint-entropy as a pattern set quality measure:

$$\mathrm{je}_{\mathbb{D}}(\mathbb{S}) = -\sum_{\mathbb{B}_i \in \mathbb{D}/\sim_{\mathbb{S}}} \frac{\mid \mathbb{B}_i \mid}{\mid \mathbb{D} \mid} \log_2 \frac{\mid \mathbb{B}_i \mid}{\mid \mathbb{D} \mid}$$

Finally, the authors of [14] use a dispersion score to rate the quality of a pattern set:

$$\mathrm{disp}_{\mathbb{D}}(\mathbb{S}) = \frac{1}{\mid \mathbb{D} \mid^2} \sum_{p_i, p_j \in \mathbb{S}} (\mid \{t \in \mathbb{D} : p_i(t) = p_j(t)\} \mid - \mid \{t \in \mathbb{D} : p_i(t) \neq p_j(t)\} \mid)^2$$

Since this score grows with the size of the pattern set, we use the following normalized version to compare pattern sets in Section 4:

$$\mathrm{dispNorm}_{\mathbb{D}}(\mathbb{S}) = \mathrm{disp}_{\mathbb{D}}(\mathbb{S}) / (\mid \mathbb{S} \mid \cdot \mid \mathbb{S} \mid -1)/2$$

To evaluate and compare pattern sets generated by different approaches we employ a SVM with the popular Tanimoto kernel. Given two molecules encoded as bit-vectors $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^d$ using d mined patterns, it is defined as:

$$\mathcal{K}_T(\boldsymbol{x}, \boldsymbol{y}) = \frac{\sum_1^d \min\{x_i, y_i\}}{\sum_1^d \max\{x_i, y_i\} - \sum_1^d \min\{x_i, y_i\}}$$

Obviously, if both vectors' components are mostly 1 and only a few 0s distinguish them, even kernel values for different instance pairs will be close to each other. Such badly scaled kernels are known to be problematic for successful prediction.

3 Existing Approaches

The work concerned with finding good pattern sets for classification falls into two categories: 1) post-processing of a set of patterns, which is similar to feature selection techniques from machine learning, and 2) iterative pattern set mining techniques that extend and improve a candidate pattern set.

The first type of approaches has the advantage that only a single pattern mining run has to be performed. To ensure that the variety of patterns is high enough to enable the extraction of a suitable subset, it is usually necessary to mine a very large amount of patterns. This makes the post-processing step (and possibly the mining step) potentially expensive, especially if some features are not informative on their own, but lead to high predictive accuracy when considered together. In [10], this problem is addressed by assuming a size-constraint that is always present. The authors discuss several desirable measures and discuss their potential mutual exclusivity. In a related work, the authors gave an algorithm for finding pattern sets maximizing the *joint entropy* (*JE*) criterion. JE effectively measures the ability of a pattern set to induce a partition of equally sized equivalence classes.

Most existing systems differ in the search strategy and the criterion used to rate the quality of a pattern set. For instance, the system presented in [4] uses greedy search and a supervised quality measure, which is based on the correlation of a pattern with a class and the similarity to earlier patterns. The search uses a database coverage constraint to control pattern set size. In this manner, it minimizes the expected number of correspondences and maximizes the expected number of equivalence classes. The approaches discussed in [2] are also based on greedy search, but use an unsupervised quality measure, which quantifies how well sets of patterns partition the data, *without* any reference to a class label. The search strategy in [8], finally, is a local search technique, which selects a non-redundant set of patterns from randomly sampled maximal graph-structures.

The second style of structured pattern mining approaches uses an iterative approach, where a candidate pattern set is extended step by step. Iterative mining has the advantage that the mining runs in later iterations can be tailored towards the shortcomings of the previously generated pattern sets. This is in contrast to post-processing, where one assumes (or hopes) that the number of generated patterns is large enough to allow the extraction of an informative subset. On the other hand, there is often no clear way to identify how many iterations will be needed to mine a useful set, and the cumulative computational cost of several pattern mining runs can become rather high. Iterative mining can be performed in two settings: 1) as *sequential mining*: mining is performed strictly sequentially, only one mining process per iteration. This leads to a clear relationship among patterns: each pattern is influenced by those that were mined before it and influences those mined after it. For efficient handling, some algorithms modify the underlying graph database between iterations. The second setting is that of 2) *parallel mining*: in any iteration several mining processes can run in parallel. generally, the results of one mining process are used to split the database into two or more parts, so the mining steps in later iterations work on smaller subsets of data instances.

The two existing sequential approaches differ somewhat. The search strategy in [14] is stochastic local search to maximize the *class-correlated dispersion score* of patterns with regard to patterns mined in earlier iterations. This score, similar to the approach in [4], trades off class-correlation with the similarity of patterns with regard to coverage in the data, or in other words, the minimization of correspondences and the maximization of equivalence classes. In [16], mining for patterns themselves is performed exhaustively using an upper bound and minimum support criterion to heuristically optimize the submodular *correspondence-based quality criterion*. Maximization of equivalence classes, if it happens at all, is only a side effect of the process. The combination of exclusive focus on correspondence minimization and sequential mining leads to *very* compact sets of patterns.

The two approaches falling into the latter group [1, 5], use the usual decision tree induction mechanism: a single pattern is mined using *information gain* and the data split according to matching of the pattern, before the algorithm

recurses on the derived subsets. Since information gain rewards class discrimination, repeated applications will lead to the minimization of correspondences while the fact that mining is performed on subsets of the data can lead to the split of instances from the same class in other parts of the data, thus increasing the number of equivalence classes.

4 Comparison of Systems and Quality Measures

In this section we describe experiments comparing the various approaches outlined earlier and the different quality measures that are used in literature to rate pattern sets for classification. In particular, we investigate *joint entropy*, the normalized and unnormalized *dispersion score*, the number of *correspondences* and the number of *equivalence classes*. The main goal here is to explore which approaches and quality measures lead to pattern sets with high predictive accuracy as measured by the AUC of the final classifier. The trends identified in these experiments can then be used to design fast algorithms generating small feature sets with good predictive accuracy. We investigate the following systems:

- Baseline. The 500 most general (shortest) free graph patterns mined under a minimum support threshold of 5%.
- PICKER*. This is a technique introduced in [2], using the inference measure and no threshold. The underlying pattern set consists of all free graph patterns mined with minimum support 5%, i.e. a superset of the baseline.
- DISP. The stochastic local search technique from [14], optimizing class-correlated dispersion score.
- FCORK as introduced in [16], whose authors supplied us with an executable. Since unconstrained experiments using this technique did not terminate on the NCI data and the cancer data set, a 5% minimum support constraint was used.
- DTM. This is an implementation similar to MBT, introduced in [5]. We were unfortunately not able to obtain an executable of the algorithm from the authors of this paper. Since we had published a similar technique in 2005 under the name TREE^2 [1], however, we extended our implementation to work on graph-structured data and discarded the decision tree after mining. Based on past results [3], we chose to mine sequential patterns which perform as well as graph-structured patterns. We will refer to this technique as "decision tree-like miner" (DTM) in the following.

Generally, we chose the parameters so that only small frequency or weak selectivity constraints were imposed. This ensures that the systems have a large number of candidate patterns available for inclusion. Since different methods generate pattern sets of varying sizes, it is often difficult to compare them directly. To allow for a fair comparison, we thus sometimes cut back the number of features to match the number produced by other techniques. Cutting-back is done by keeping the k highest-ranked patterns/those mined in the k first iterations, with k derived from the size of competing techniques' output.

For the experiments, we used the data sets shown in Table 1. This includes the NCI data sets first reported in [15] (specifically those on which fCork with minimum support five percent terminated in reasonable time), as well as the *Blood Brain Barrier*, *NCTRER*, *Yoshida*, and *Cancer* data sets used in [14]. To evaluate the quality of the derived encodings, we performed a 10-fold cross-validation, using an SVM [9] with Tanimoto Kernel, with the C-parameter set to 1.0. This value gave good performance over the entire range of data sets.

Table 1. An overview of the used data sets

Data set	instances	majority class
NCI 786_0	3154	1648
NCI A549_ATCC	3359	1710
NCI CAKI_1	3221	1678
NCI CCRF_CEM	3131	1995
NCI COLO_205	3279	1748
NCI SF_539	3045	1728
Blood Brain Barrier	373	248
NCTRER	208	125
Yoshida	238	143
Cancer	30796	15590

As a first experiment we investigated how the number of features affects classification accuracy. If prediction accuracy increases with the number of features, this would indicate that strict pattern set selection is misguided and permissive feature generation approaches should be preferred. In Figure 1 we plot the average number of features selected per fold against the AUC, labeled by data set. The plot shows that accuracy increases moderately or not at all with the number of features. It is remarkable that the Pearson correlation coefficient between number of features and AUC is actually positive for the larger datasets (0.4 for NCI and 0.57 for Cancer), but negative for the smaller data sets (-0.02 to -0.12 for Blood Brain Barrier, NCTRER, Yoshida). While this is not statistically significant, it seems to be consistent with the results in [13]: pattern set size should increase with the number of training instances, but overfitting is not as severe as in many other classification settings.

In the second experiments, we examine which pattern set criterion is a good indicator of prediction accuracy. Figures 2 – 5 give the scatter plots for the normalized dispersion score, average joint entropy, number of equivalence classes and number of correspondences. The results are mixed, but there are a few interesting insights. First of all, joint entropy and the number of equivalence classes appear to be fairly well correlated to prediction accuracy. Indeed, the following correlation coefficients are significant on the 99% significance level: The correlation between joint entropy and AUC is between 0.8 and 0.95, with the only outlier at 0.72 for the NCTRER data set. For the number of equivalence classes, the Pearson correlation coefficient is large for the bigger data sets (>

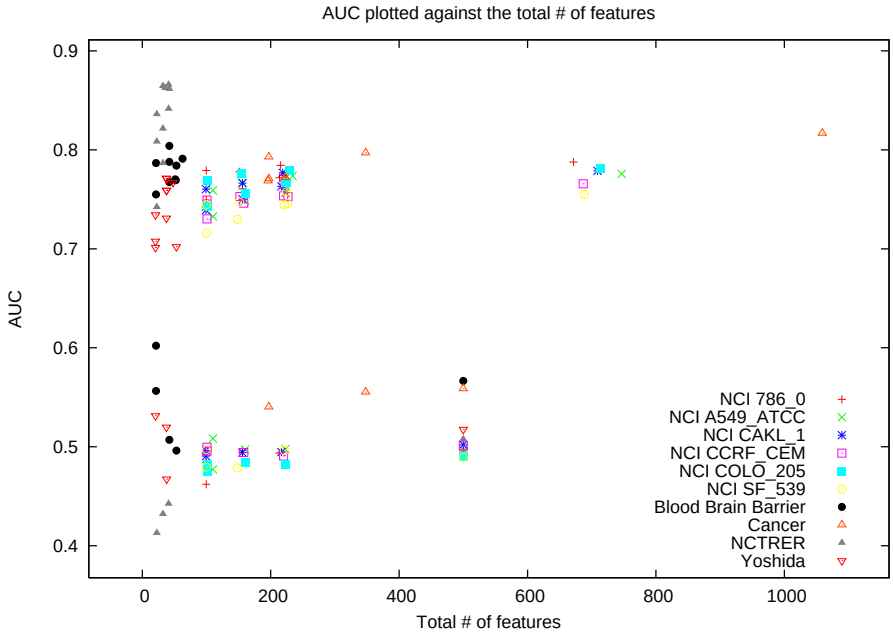

Fig. 1. AUC plotted against the average number of features selected

0.95 for Cancer and NCI), but still reasonable for the smaller ones (0.63-0.8 for NCTRER, Blood Brain Barrier, Yoshida). Looking at Figure 5, one can see that a large number of correspondences always indicates bad AUC performance. This means a pattern mining system needs a way to minimize the number of occurrences to be successful. Unfortunately, it is clearly not enough to only optimize for correspondences: there are a number of settings where pattern sets with low number of correspondences still lead to terrible predictive accuracy. It is clear that successful systems also need to optimize the variety within the instances of the same class. This is what joint entropy and the number of equivalence classes measure. Finally, the dispersion score is only slightly correlated with AUC. The dispersion score varies between around -0.10 for the smaller data sets and around +0.25 for the larger ones. It is noteworthy, though, that the dispersion score improves if class information is included.

Overall, one can conclude that supervised pattern mining scores tend to be better indicators of prediction accuracy and that the characteristics change between larger and smaller datasets. This means that successful techniques should include information about the class label during pattern generation and that the number of generated patterns should be in relation to the number of training instances. To see how the evaluated methods succeed in this regard, Table 2 shows the average rankings of different methods w.r.t. the four quantitative measures and AUC. Highest-ranked in terms of AUC is DTM, which also ranks very high

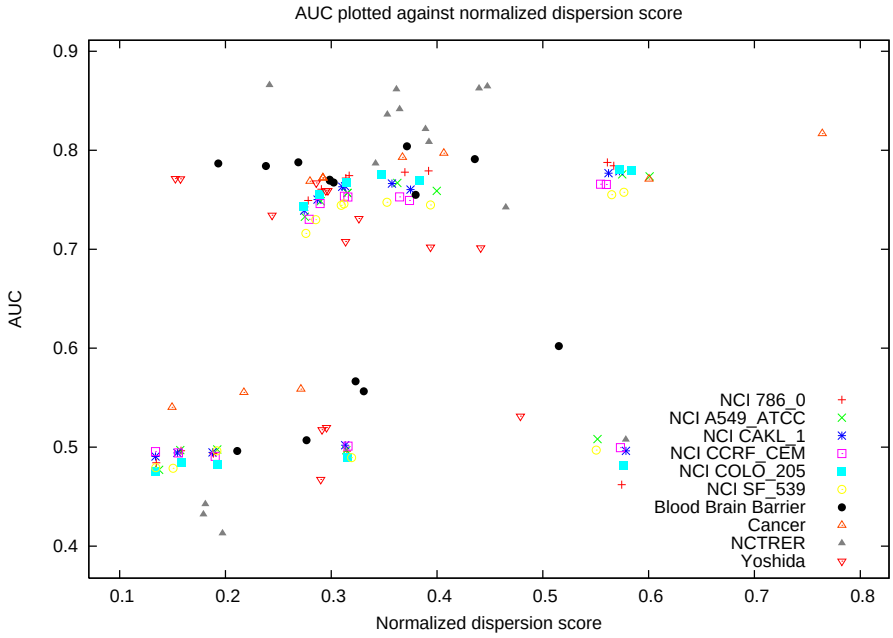

Fig. 2. AUC plotted against the average normalized dispersion score

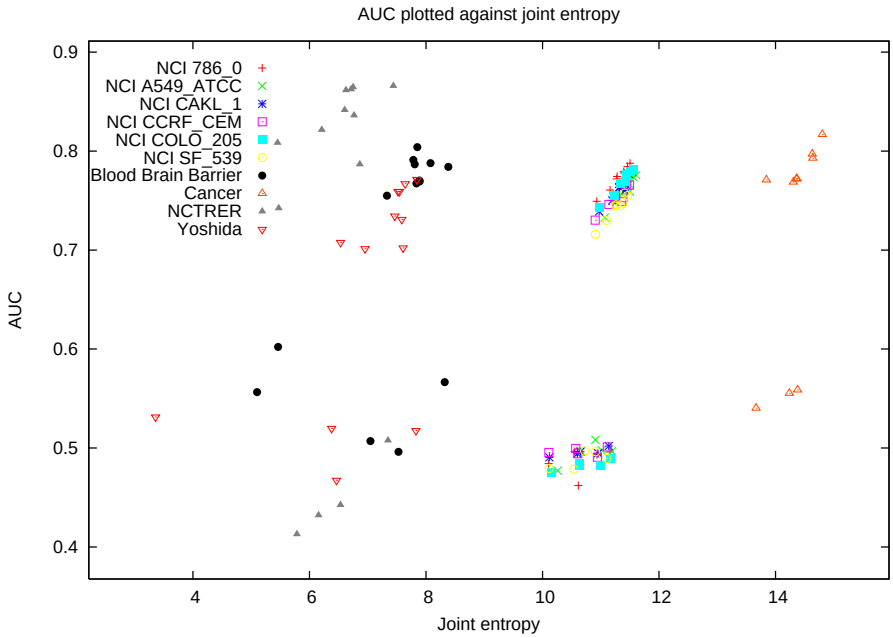

Fig. 3. AUC plotted against the average joint entropy

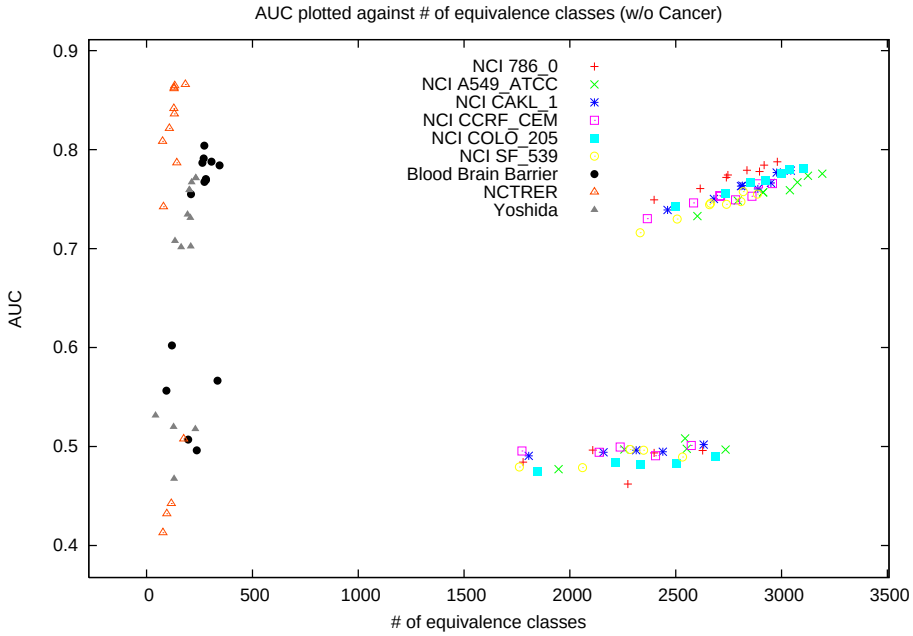

Fig. 4. AUC plotted against the average number of equivalence classes (w/o Cancer)

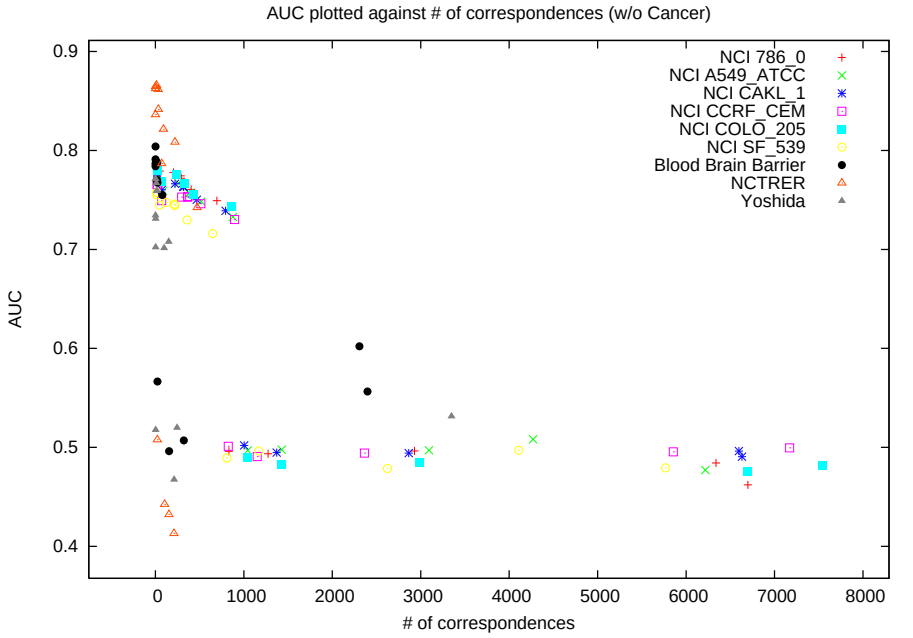

Fig. 5. AUC plotted against the average number of correspondences (w/o Cancer)

Table 2. The respective ranking of techniques for different criteria

AUC	# equivalence classes	# correspondences	dispersion score	joint entropy
DTM (1.44)	PICKER* (1.7)	DTM (1.6)	FCORK (1)	PICKER* (1.9)
PICKER* (2)	DTM (2)	FCORK (1.7)	PICKER* (2.1)	DTM (2)
FCORK (3.22)	FCORK (3.4)	PICKER* (2.8)	DISP (3)	FCORK (3.1)
DISP (3.33)	DISP (3.8)	DISP (4.2)	Baseline (4.3)	DISP (4)
Baseline (5)	Baseline (4.1)	Baseline (4.7)	DTM (4.6)	Baseline (4)

Table 3. Number of features divided by joint entropy of the feature set

Data set	Baseline	PICKER*	DISP	FCORK	DTM
NCI 786_0	44.89±0.0579	13.68±0.0048	19.71±0.1086	8.74±0.3560	58.39±0.7455
NCI A549_ATCC	44.77±0.1198	13.92±0.0071	19.66±0.1658	9.56±0.3645	64.33±1.1459
NCI CAKI_1	44.98±0.1417	13.64±0.0071	19.79±0.0934	8.68±0.3641	61.49±1.1933
NCI CCRF_CEM	45.08±0.1046	13.85±0.0041	20.18±0.0840	8.87±0.4358	59.75±1.1532
NCI COLO_205	44.84±0.114	14.05±0.0078	19.89±0.0798	8.89±0.1390	61.66±0.7693
NCI SF_539	45.11±0.0663	13.02±0.0047	20.17±0.1445	8.79±0.2561	60.07±0.9540
Blood Brain Barrier	60.20±0.1036	6.32±0.0088	6.80±0.4355	2.74±0.1302	8.05±0.4570
NCTRER	68.17±0.3723	5.63±0.0403	6.52±0.1542	3.34±0.1906	5.33±0.3608
Yoshida	64.013±0.0963	4.84±0.0065	6.44±0.2702	2.74±0.1100	6.96±0.5243
Cancer	34.84±0.0059	23.78±0.0033	15.52±0.8953	13.46±0.2409	

in terms of equivalence classes *and* correspondences, coming second in terms of equivalence classes only to PICKER* that directly optimizes those. Apparently, DTM's approach to class-sensitive pattern generation and its ability to relate feature set size to training set size lead to good overall performance. DTM's bad dispersion score actually supports this point: the unnormalized dispersion score we report increases in the number of patterns involved. The fact that DTM finishes last in terms of it shows that it generates more features than most competing methods on the large datasets. This is also illustrated in Table 3. Joint entropy can be considered as giving the number of bits that are needed to encode the partition. Since each feature acts in fact as a bit in the encoding, the table essentially gives the average number of patterns per bit of information. One can see that DTM adapts better to the varying training set sizes than for instance FCORK or the Baseline. In fact, it adapts a bit too well and the drawback of its large pattern set sizes can be seen in Table 7, which gives the runtimes. Here, DTM performs worst by a large margin. For the cancer dataset, we had to cancel the run after 100 hours.

In order to find a highly predictive, yet fast technique, it is therefore desirable to keep the advantages of DTM – focus on supervised partition generation and sensitivity to the size and characteristics of the underlying dataset – while doing away with the drawbacks – the overly large number of features and long running times.

Table 4. Running times per fold for different techniques (the techniques that are not listed had negligible running times)

Data set	fCork	Disp	DTM
NCI 786_0	4h40m±1h46m	1h2m±7m54s	4h46m±36m48s
NCI A549_ATCC	4h47m±1h53m	1h3m±6m39s	11h31m±4h0m
NCI CAKI_1	3h55m±1h7m	1h5m±6m28s	11h39m±3h17m
NCI CCRF_CEM	4h24m±1h7m	1h1m±9m55s	12h46m±4h55m
NCI COLO_205	3h22m±1h34m	1h5m±7m33s	10h43m±3h33m
NCI SF_539	5h54m±4h16m	1h0m±5m11s	11h37m±4h44m
Blood Brain Barrier	20m12s±11m26s	6.14s±0.98s	7m 2s±1m5.65s
NCTRER	1h10m±38m	1.52s±0.18s	4m31.6s±1m3.75s
Yoshida	1m42s±1m4s	3.16s±0.42s	7m7.5s±1m23.65s
Cancer	10h41m±1h	13h8m±20m	100h+

5 The ReMine Algorithm

DTM, shown in Algorithm 1 iteratively mines patterns in the same way as a binary decision tree is induced on class-labeled data: first, a single test that scores best according to *information gain* is extracted (line 2). In a standard decision tree, such a test would be an attribute-value pair, while in DTM a graph-structured pattern is mined. The data is then split into two subsets, one on which the test matches, the second on which it does not (line 4), and the operation repeated on the derived subsets (line 5). A problem is that information gain, in contrast to e.g. minimum frequency, is not anti-monotone. However, it is possible to calculate an upper bound for information gain and use it for forward-pruning to make mining for the best pattern according to information gain ($\mathcal{T}h_1(\mathbb{D}, \mathcal{L}, \phi)$) feasible [12]. Information gain rewards patterns that sepa-

Algorithm 1 The DTM algorithm

DTM($\mathbb{D}$)
1: $\mathcal{F} = \emptyset$
2: $\mathcal{F}_{new} = \mathcal{T}h_1(\mathbb{D}, \mathcal{L}, \phi)$ – this yields zero or one feature
3: **if** $\mathcal{F}_{new} \neq \emptyset$ **then**
4: $\quad \mathbb{P} = \{\mathbb{B}_1, \mathbb{B}_2\} = \mathbb{D}/\sim_{\mathcal{F}_{new}}$
5: $\quad \mathcal{F} = \mathcal{F}_{new} \cup \text{DTM}(\mathbb{B}_1) \cup \text{DTM}(\mathbb{B}_2)$
6: **end if**
7: **return** $\mathcal{F}$

rate instances with different class labels from each other, in this way reducing correspondences on the subset on which a pattern is mined. We aim at keeping this basic mining process of DTM, still mining a single best pattern from a subset. The difference lies in how we apply the patterns to derive *new* subsets for the following iteration: Instead of using each pattern to split only the subset

on which it was mined, we use *all* patterns derived so far to induce a partition on the entire data (line 3 in Algorithm 2).

This can be illustrated by considering the first three iterations: in the first step, both DTM and ReMine mine the best pattern according to information gain and use it to split the whole data set into two subsets. In the second iteration, two more patterns are mined but when these are used to split the subsets, DTM and ReMine behave differently. DTM splits each subset in two, according to the pattern mined from each, for a maximum of **four** subsets. ReMine, on the other hand, splits each subset according to *both* patterns. This can lead maximally to four *new* subsets from each current one, for a total of maximally **eight** subsets of the data. The maximum number of different blocks that can be encoded by three binary features is eight, which shows that ReMine can use patterns much more efficiently than DTM when it induces a partition. In the third iteration, DTM will therefore mine for four new patterns, while ReMine mines for eight.

Accordingly, the ReMine algorithm consists of a main loop (line 2-10) where in each iteration the whole data set $\mathbb{D}$ is partitioned according to the current set of features $\mathcal{F}$ (line 3). Then for each of the blocks $\mathbb{B}_i$ of the partition $\mathbb{P}$ the feature giving the highest information gain is extracted (line 6) which are then joined with the previous features (line 8). If the new partition $\mathbb{P}'$ induced by the resulting enhanced feature set has not changed, the algorithm terminates and returns the found set of features $\mathcal{F}$.

Algorithm 2 The ReMine algorithm

```
 1: F = ∅ – the initial set of features
 2: repeat
 3:    P = D/ ∼F – partition induced by the current feature set
 4:    F_new = ∅
 5:    for all blocks B_i ∈ P do
 6:       F_new = F_new ∪ Th_1(B_i, L, φ)
 7:    end for
 8:    F = F ∪ F_new
 9:    P' = D/ ∼F – partition induced by the new feature set
10: until P' = P
11: return F
```

6 Experimental Evaluation

Our goal in proposing the ReMine technique lies in keeping DTM's good performance in terms of AUC while at the same time reducing the number of patterns mined which should help in achieving lower running times. We therefore repeat our earlier experiments and compare ReMine's results against those of fCork, Picker*, and DTM. As Table 5 shows, ReMine achieves on average second-

Table 5. Ranking of techniques for different criteria, including ReMine now

AUC	# equivalence classes	# correspondences	dispersion score	joint entropy
DTM (1.89)	DTM (2.2)	ReMine (1.7)	fCork (1)	DTM (2.2)
ReMine (2)	Picker* (2.4)	fCork (2.4)	Picker* (2.2)	Picker* (2.6)
Picker* (2.78)	ReMine (2.6)	DTM (2.5)	Disp (3.3)	ReMine (2.7)
fCork (4.11)	fCork (4.4)	Picker* (3.6)	ReMine (3.7)	fCork (4)
Disp (4.22)	Disp (4.6)	Disp (5.2)	Baseline (5.2)	Disp (4.7)
Baseline (6)	Baseline (4.8)	Baseline (5.6)	DTM (5.6)	Baseline (4.8)

best AUC (by a small margin), and performs at least as good as DTM in terms of the quantitative criteria, improving on it for the number of correspondences and the dispersion score. This means that we achieved our goal to keep the good performance and the ranking indicates improvements in the number of features as well. Table 6 confirms this indication, showing that while ReMine does not

Table 6. Number of features divided by joint entropy of the feature set, including ReMine now

Data set	Picker*	fCork	DTM	ReMine
NCI 786_0	13.68±0.0048	8.74±0.3560	58.385±0.745544	18.79±1.0739
NCI A549_ATCC	13.92±0.0071	9.56±0.3645	64.33±1.14595	20.23±0.9974
NCI CAKI_1	13.64±0.0071	8.68±0.3641	61.49±1.19338	19.04±0.9252
NCI CCRF_CEM	13.85±0.0041	8.87±0.4358	59.75±1.15327	19.28±0.6363
NCI COLO_205	14.05±0.0078	8.89±0.1390	61.66±0.76938	19.85±0.7491
NCI SF_539	13.02±0.0047	8.79±0.2561	60.07±0.95406	19.68±1.059
Blood Brain Barrier	6.32±0.0088	2.74±0.1302	8.05±0.4570	5.36±0.4502
NCTRER	5.63±0.0403	3.34±0.1906	5.33±0.3608	4.77±0.2409
Yoshida	4.84±0.0065	2.74±0.1100	6.96±0.5243	4.95±0.2858
Cancer	23.78±0.0033	13.46±0.2409		71.55±2.5414

mine as few patterns as Picker* and fCork, it strongly reduces the size of the feature set compared to DTM. The final aspect to be evaluated is that of running times and as Table 4 shows, along with the reduction in features mined comes a reduction in running times that is so pronounced that ReMine runs even faster than fCork while mining more features (and leading to better AUC).

7 Conclusions and Future Work

In this paper we evaluated the importance (in terms of AUC) of several selection criteria for pattern set mining. We found that minimizing correspondences is necessary, but not sufficient for predictive classifiers, whereas general partitioning scores are good overall indicators of pattern set quality. We also found that successful methods need to adapt the pattern set size to the characteristics of the datasets. Unfortunately, large pattern sets require many expensive mining steps

Table 7. Running times per fold for different techniques (the techniques that are not listed had negligible running times)

Data set	fCork	DTM	ReMine
NCI 786_0	4h40m±1h46m	4h46m±36m48s	59m27s±13m56s
NCI A549_ATCC	4h47m±1h53m	11h31m±4h0m	59m27s±13m56s
NCI CAKI_1	3h55m±1h7m	11h39m±3h17m	2h36m±1h57m
NCI CCRF_CEM	4h24m±1h7m	12h46m±4h55m	2h20m±1h51m
NCI COLO_205	3h22m±1h34m	10h43m±3h33m	2h47m±2h17m
NCI SF_539	5h54m±4h16m	11h37m±4h44m	2h59m±1h54m
Blood Brain Barrier	20m12s±11m26s	7m 2s±1m5.65s	2m16.4s±24.2s
NCTRER	1h10m±38m	4m31.6s±1m3.75s	1m48.5s±15.9s
Yoshida	1m42s±1m4s	7m7.5s±1m23.65s	1m53.9s±11.9s
Cancer	10h41m±1h	100h+	13h51m±2h14m

and large runtimes.We thus designed a faster and more efficient adaptation of the DTM algorithm. ReMine performs the local pattern mining step in the same way as DTM, but uses *all* patterns instead of a single pattern per iteration on the *entire* data to induce the new partition. Our experimental evaluation showed that ReMine retains the good performance of DTM in terms of AUC, has similar characteristics in the number of equivalence classes and correspondences, reduces the number of patterns mined, and has lower runtimes.

The aforementioned partial redundancy among patterns mined by ReMine could actually go as far as producing completely redundant, e.g. complementary, patterns. Whether removing such redundant patterns or other redundancy control methods would improve or impair the usefulness of the derived encoding is an open question. There is also the issue that decision-tree like iterative miners like ReMine (or MbT and Tree^2) can be parallelized, giving them a further speed advantage over sequential miners like fCork. Finally, we were only concerned with graph-structured data in this work, other representations, including unstructured data such as itemsets, remain a topic for future work.

Acknowledgements

We thank Marisa Thoma for making an executable of the fCork algorithm available to us. We thank Luc De Raedt for helpful comments and discussion, as well as the anonymous reviewers for their valuable input. The work presented here was partially supported by the European Commission under the 7th Framework Programme FP7-ICT-2007-C FET-Open, contract no. BISON-211898 and by Deutsche Forschungsgemeinschaft, contract no. RU 1589/1-1.

References

1. Bringmann, B., Zimmermann, A.: Tree^2 - Decision trees for tree structured data. In: Jorge, A., Torgo, L., Brazdil, P., Camacho, R., Gama, J. (eds.) 9th European Conference on Principles and Practice of Knowledge Discovery in Databases. pp. 46–58. Springer (2005)

2. Bringmann, B., Zimmermann, A.: One in a million: picking the right patterns. Knowledge and Information Systems 18(1), 61–81 (2009)
3. Bringmann, B., Zimmermann, A., De Raedt, L., Nijssen, S.: Don't be afraid of simpler patterns. In: Fürnkranz et al. [6], pp. 55–66
4. Cheng, H., Yan, X., Han, J., Hsu, C.W.: Discriminative frequent pattern analysis for effective classification. In: Proceedings of the 23rd International Conference on Data Engineering. pp. 716–725. IEEE (2007)
5. Fan, W., Zhang, K., Cheng, H., Gao, J., Yan, X., Han, J., Yu, P.S., Verscheure, O.: Direct mining of discriminative and essential frequent patterns via model-based search tree. In: Li, Y., Liu, B., Sarawagi, S. (eds.) Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 230–238. ACM (2008)
6. Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.): Knowledge Discovery in Databases: PKDD 2006,10th European Conference on Principles and Practice of Knowledge Discovery in Databases, Berlin, Germany, September 18-22, 2006, Proceedings. Springer (2006)
7. Geamsakul, W., Matsuda, T., Yoshida, T., Motoda, H., Washio, T.: Performance evaluation of decision tree graph-based induction. In: Grieser, G., Tanaka, Y., Yamamoto, A. (eds.) Discovery Science, 6th International Conference. pp. 128–140. Springer, Sapporo, Japan (Oct 2003)
8. Hasan, M.A., Chaoji, V., Salem, S., Besson, J., Zaki, M.J.: Origami: Mining representative orthogonal graph patterns. In: Ramakrishnan, N., Zaiane, O. (eds.) ICDM. pp. 153–162. IEEE Computer Society (2007)
9. Joachims, T.: Making large-scale support vector machine learning practical. In: Advances in kernel methods: support vector learning, pp. 169–184. MIT Press, Cambridge,MA,USA (1999)
10. Knobbe, A.J., Ho, E.K.Y.: Pattern teams. In: Fürnkranz et al. [6], pp. 577–584
11. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. Data Mining and Knowledge Discovery 1(3), 241–258 (1997)
12. Morishita, S., Sese, J.: Traversing itemset lattice with statistical metric pruning. In: Proceedings of the 19th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (2000)
13. Rückert, U.: Capacity control for partially ordered feature sets. In: ECML PKDD '09: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 318–333. Springer-Verlag, Berlin, Heidelberg (2009)
14. Rückert, U., Kramer, S.: Optimizing feature sets for structured data. In: Kok, J.N., Koronacki, J., de Mántaras, R.L., Matwin, S., Mladenic, D., Skowron, A. (eds.) 18th European Conference on Machine Learning. Lecture Notes in Computer Science, vol. 4701, pp. 716–723. Springer (2007)
15. Swamidass, S.J., Chen, J.H., Bruand, J., Phung, P., Ralaivola, L., Baldi, P.: Kernels for small molecules and the prediction of mutagenicity,toxicity and anti-cancer activity. In: ISMB (Supplement of Bioinformatics). pp. 359–368 (2005)
16. Thoma, M., Cheng, H., Gretton, A., Han, J., Kriegel, H.P., Smola, A.J., Song, L., Yu, P.S., Yan, X., Borgwardt, K.M.: Near-optimal supervised feature selection among frequent subgraphs. In: Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 – May 2, Sparks, Nevada. pp. 1–12. SIAM (2009)
17. Zaki, M.J., Aggarwal, C.C.: XRules: an effective structural classifier for XML data. In: Getoor, L., Senator, T.E., Domingos, P., Faloutsos, C. (eds.) Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 316–325. ACM, Washington, DC, USA (Aug 2003)