The Chosen Few: On Identifying Valuable Patterns

Björn Bringmann
Katholieke Universiteit Leuven
Departement Computerwetenschappen
Celestijnenlaan 200a
3001 Heverlee
Bjoern.Bringmann@cs.kuleuven.be

Albrecht Zimmermann
Katholieke Universiteit Leuven
Departement Computerwetenschappen
Celestijnenlaan 200a
3001 Heverlee
Albrecht.Zimmermann@cs.kuleuven.be

Abstract

Constrained pattern mining extracts patterns based on their individual merit. Usually this results in far more patterns than a human expert or a machine learning technique could make use of. Often different patterns or combinations of patterns cover a similar subset of the examples, thus being redundant and not carrying any new information. To remove the redundant information contained in such pattern sets, we propose a general heuristic approach for selecting a small subset of patterns.

We identify several selection techniques for use in this general algorithm and evaluate those on several data sets. The results show that the technique succeeds in severely reducing the number of patterns, while at the same time apparently retaining much of the original information. Additionally the experiments show that reducing the pattern set indeed improves the quality of classification results. Both results show that the approach is very well suited for the goals we aim at.

1. Introduction

Pattern search is one of the main topics in data mining. In the last years different types of pattern languages were developed in order to be able to deal with the ever-present challenge of finding new and hopefully valuable representations for all kind of data. Most algorithms developed handle the usually computationally intensive task in a decent way enabling us to extract millions of patterns from even small data sets. These patterns are then presented to the user or they are used as features such that each instance of the original database can be converted into a binary vector, each bit encoding the presence or absence of the pattern.

Due to the fact that *interestingness* (i.e. constraint satisfaction) of patterns is evaluated for each pattern individually, the amount of patterns to be considered by a user is often too large. Furthermore, when presenting the binary vector data to a machine learning technique, an overabundance of features does not help in the learning task, possibly even "confusing" the algorithm, leading to overfitting.

The aim of our work is to reduce the set of patterns returned by a data mining operation to a subset that is small enough to be inspectedby a human user. To be of maximal benefit to the user this set should show little redundancy while retaining as much information as possible encoded in the full pattern set. In our approach, the information of a pattern set is determined by the partition it induces on the examples, with all examples containing the same set of patterns belonging to one and the same block. Thus, information is obtained from the composition of *all* patterns. This is in contrast to e.g. the notion of sets of *closed* [12] patterns, which only takes care that no two individual patterns induce the same partition. It does not take into account complementary information, i.e. patterns that are mutually exclusive. Additionally, especially on dense data sets the number of closed patterns is still significantly larger than anything humans can be reasonably expected to peruse.

Since the high amount of patterns leads to a very large search space of possible subsets, we develop a heuristic technique for solving the problem. In order to accommodate different notions of how to traverse the set and how to select patterns, we keep the main algorithm rather general. Based on our intuition about redundancy between patterns we develop several selection measures and combine them with straight-forward ordering strategies. When evaluated on several data sets the techniques reduce the set of closed patterns severely.

The rest of paper is structured as follows: in the next Section we introduce notions used throughout this work and describe the general formulation of our technique. In Section 3 we describe several instantiations of the general algorithm, motivating the different selection strategies. In Section 4 we perform experiments on several data sets, showing the effectiveness of the used selection techniques, and comparing and discussing the resulting reduced pattern sets. In Section 5 we discuss related work and finally, we conclude in Section 6.

2. The General Algorithm

As stated above, the main goal of our work is presenting both human user and machine learning technique with a set of patterns that is small enough to be easily processed. Small size is not a virtue in itself, however, if there is little information encoded in the pattern set. Thus it is important that much of the characteristics of the data described by the complete set of patterns is retained, and it is desirable that the redundancy between patterns in the set is low.

To achieve selection of patterns beyond that done in the mining operation, additional knowledge is required. The cheapest and often only form of background knowledge available is the database $\mathcal{T}$ which the patterns were extracted from. By evaluating subsets of patterns *jointly* on this database, we exploit the knowledge the mining operation itself did not use.

Hence, our goal is the following: Given a *set* $\mathbb{S}$ of *patterns* p_i, the database $\mathcal{T}$, and a redundancy measure Φ, select a subset $\mathbb{S}^* \subseteq \mathbb{S}$, such that $\mathbb{S}^*$ satisfies the following requirements:

1. $|\mathbb{S}^*|$ is small, such that a human expert could inspect it.
2. members of $\mathbb{S}^*$ have low redundancy w.r.t. $\mathcal{T}$ according to Φ.
3. members of $\mathbb{S}^*$ describe characteristics of $\mathcal{T}$.

We will concretize these requirements in the next section. To give a formal description of the general technique, we introduce the following notions:

2.1. Notions

Let each pattern p be associated with a function $p : \mathcal{T} \rightarrow \{true, false\}$. We define $p(t) = true$ if p matches t, and $p(t) = false$ otherwise. In general a pattern does not have *values* as such but can only be present or absent. Associating a pattern with this boolean function does allow us to consider each pattern as a binary feature though.

A set of patterns $\mathbb{S} = \{p_1, \ldots, p_n\}$ is called a *pattern set*. Given a pattern set $\mathbb{S}$ we define an equivalence relation $\sim_{\mathbb{S}}$ on the set $\mathcal{T}$ of transactions as

$$\sim_{\mathbb{S}} \; = \{(t_1, t_2) \in \mathcal{T} \times \mathcal{T} \mid \forall_{p \in \mathbb{S}}\, p(t_1) = p(t_2)\}$$

Thus, two transactions are considered to be equivalent under $\mathbb{S}$ if they share exactly the same patterns. Using the equivalence relation $\sim_{\mathbb{S}}$ the *partition* or quotient set of $\mathcal{T}$ over $\mathbb{S}$ is defined as

$$\mathcal{T}/\sim_{\mathbb{S}} = \bigcup_{x \in \mathcal{T}} \{a \in \mathcal{T} \mid a \sim_{\mathbb{S}} x\},$$

with the equivalence classes also called *blocks*.

To get an intuition why partitions are central to our technique, consider the following: For a machine learning algorithm a subset $\mathbb{S}^*$ that induces the same partition as $\mathbb{S}$ will be of the same usefulness as the complete set since the separability of instances is equally well possible. The actual syntactical composition and support of the patterns are *not* of interest in this case.

For a human user with elaborated knowledge about the domain the situation might be somewhat different, but by defining the total order $< \subset \mathbb{S} \times \mathbb{S}$ in which to process patterns the user can control to some extend which patterns are considered for selection. Additionally, considering a subset of patterns that induces a partition on the entire data is preferable to browsing hundreds or even thousands of patterns or looking through the k most *interesting* which possibly contain more or less the same information.
Furthermore, we define a measure Φ as

$$\Phi(\mathcal{T}, \mathbb{S}^*, p) \rightarrow [0, 1] \subset \mathbb{R}$$

The measure is supposed to rate the value of adding p to $\mathbb{S}^*$. The higher the value of Φ, the more valuable it would be adding p to $\mathbb{S}^*$.

To motivate the usage of an additional measure, consider that the minimal number of patterns needed to induce a partition is $\log_2 |\mathcal{T}/\sim_{\mathbb{S}^*}|$. While this ideal number will hardly be reachable, it is still necessary to rate patterns regarding their contribution towards approaching it since this also minimizes redundancy between (combinations of) patterns.

2.2. The idea

Given the goals stated above and the intuition about meaningful pattern sets we have given in the preceding section, our aim is now to select a subset $\mathbb{S}^* \subset \mathbb{S}$ which comes close to recovering the partition induced by $\mathbb{S}$ while being of low cardinality.

Since the number of patterns is rather high, the brute force approach of testing any possible subset $\mathbb{S}^* \subseteq \mathbb{S}$ will quickly become infeasible. A possible solution to this lies in limiting the size of $\mathbb{S}^*$ to a user-defined k [6]. The choice of this k is not straight-forward though, and furthermore not, as in the approach presented here, governed by the data.

Using the definitions given above we can now give a rather general heuristic algorithm to compute $\mathbb{S}^*$ given $\mathbb{S}$,

$\mathcal{T}$, the order $<$, Φ, and some threshold $t \in [0, 1] \subset \mathbb{R}$. It is shown as Algorithm 1.

Algorithm 1 The general algorithm

```
1: $\mathbb{S}^* = \{p_1\}$
2: for $i = 2$ to $|\mathbb{S}|$ do
3:     if $|\mathcal{T}/\sim_{(\mathbb{S}^* \cup p_i)}| > |\mathcal{T}/\sim_{\mathbb{S}^*}|$ then
4:         if $\Phi(\mathcal{T}, \mathbb{S}^*, p_i) \geq t$ then
5:             $\mathbb{S}^* = \mathbb{S}^* \cup \{p_i\}$
6: return $\mathbb{S}^*$
```

The algorithm iterates over the patterns in $\mathbb{S}$ in the order given by $<$. In each step the partition induced by $\mathbb{S}^* \cup p_i$ is created and compared to the partition induced by $\mathbb{S}^*$. If there is no change, the pattern is rejected, thus reducing redundancy (subgoal 1). If there is change, the measure Φ is evaluated against the threshold and if the threshold is passed the pattern chosen.

To give some intuition about the process, consider Figure 1. The left-hand side shows a small snapshot of $\mathcal{T}$. As can be clearly seen on the left-hand side, p_3's presence depends on the presence of both p_1 **and** p_2, and p_4's presence on the presence of p_1 and absence of p_2. The right-hand side shows the partition induced by the first three patterns on $\mathcal{T}$, and shows that p_3's dependency is mirrored in the way $\mathcal{T}$ is split into blocks.

Note that the rejection on a lack of change in the partition means that the $\mathbb{S}^*$ created by this selection will induce the *same* partition as the *whole* $\mathbb{S}$. Thus, the description of data characteristics that the original set allowed is maintained, satisfying subgoal 3.

Even though there might be cases where it is reasonable otherwise, we consider $\Phi(\mathcal{T}, \emptyset, p_i) = 1$ for all of our experiments. This means that the first pattern $p_1 \in \mathbb{S}$ according to $<$ is always in the reduced set $\mathbb{S}^*$. We will discuss other possibilities in the last section.

Obviously there are – apart from $\mathcal{T}$ – two major points influencing the result $\mathbb{S}^*$. First, there is the measure Φ which so far remains unspecified. Second, the order $<$ of the features in $\mathbb{S}$ may be important. We will discuss several possibilities for those two points in the next section.

3 Instantiations of the Algorithm

To show the applicability of our general algorithm to the task of pattern subset selection we introduce several different instantiations. We start by describing three selection measures used. To give some more motivation for each of those instantiations, we will attempt to give some "meaning" to each selection measure used.

	t_1	t_2	t_3	t_4	t_5
p_1	×		×	×	
p_2	×			×	×
p_3	×			×	
p_4			×		

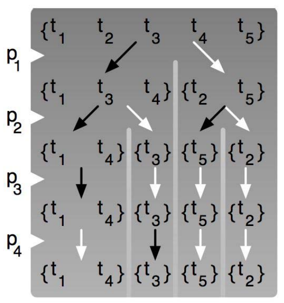
$\{t_1\ t_2\ t_3\ t_4\ t_5\}$
p_1
$\{t_1\ t_3\ t_4\}\ \{t_2\ t_5\}$
p_2
$\{t_1\ t_4\}\ \{t_3\}\ \{t_5\}\ \{t_2\}$
p_3
$\{t_1\ t_4\}\ \{t_3\}\ \{t_5\}\ \{t_2\}$
p_4
$\{t_1\ t_4\}\ \{t_3\}\ \{t_5\}\ \{t_2\}$

Figure 1. Partitions induced by patterns. The left table shows which patterns p_i occur in the transactions t_j. The right shows how the patterns split up the data set. Four binary patterns can induce at most 16 blocks. This combination of patterns and instances yields only four blocks, however.

3.1 Partition size quotient Φ_Q

While rejection of patterns that don't change the partition will effectively cut down on the number of patterns retained already, there is the possibility that adding a pattern will affect only a few blocks. While this may be acceptable in early steps of the selection process when not many patterns are used and only few blocks formed, in later steps this corresponds to only a small gain in new information. Let's assume for instance that exactly one of the existing blocks is split into two sub-blocks when a new pattern is added. This means that the total number of blocks is raised by one. Depending on the number of already existing blocks, this e.g. corresponds to 33% for two blocks, but only 0.9% for 100 blocks.

The crudest way of measuring this lies in defining the measure $\Phi_Q(\mathcal{T}, \mathbb{S}^*, p_i) = 1 - \frac{|\mathcal{T}/\sim_{\mathbb{S}^*}|}{|\mathcal{T}/\sim_{(\mathbb{S}^* \cup p_i)}|}$. We can now define a threshold on what we perceive to be an acceptable increase in the number of blocks and use it for additional pattern selection. The main advantage of this criterion is that it is easy to evaluate. A possible disadvantage might be that focusing solely on the number of blocks without considering which blocks are split and which instances are contained in the new sub-blocks is not enough.

3.2 Agglomerative clustering Φ_C

To alleviate this, one can use an agglomerative clusterer which combines some of the new sub-blocks until the old number of blocks is reached. Let's assume that as in the section before, the addition of a new pattern leads to a split of an existing block into two sub-blocks. These have a dissimilarity of 1 according to the Manhattan distance since they agree in all patterns except the new one. By the same argument the parent block had a distance of at least 1 to the other blocks. Thus one of the sub-blocks will have a distance of 2 to all blocks except its sibling. Non-split blocks might have a smaller dissimilarity to non-sibling blocks than to siblings, depending on the effect of the new pattern.

An agglomerative clusterer will combine two blocks from the new partition into one block since then the old number of blocks is reached again. Given the effects described above, there will very likely be change that can be measured using the *Rand* index, affected by the new pattern over the entire partition, even unchanged blocks. For a bigger increase in the number of blocks this change can be expected to be even more pronounced.

The *Rand* index is defined as follows: assume two partitions $\mathbb{P}, \mathbb{P}'$. For each pair of instances t_i, t_j, two decision variables exist - c_{ij} which is set to 1 if the two instances end up in the *same* block in both $\mathbb{P}$ and $\mathbb{P}'$, 0 otherwise, and $d_{ij} = 1$ if the two instances are assigned to *different* blocks in both partitions, 0 otherwise. The *Rand* index is then:

$$Rand(\mathbb{P}, \mathbb{P}') = \frac{2 * \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} + d_{ij}}{n \cdot (n-1)}$$

We define $\Phi_C(\mathcal{T}, \mathbb{S}^*, p_i) = 1 - Rand(\mathcal{T}/\sim_{\mathbb{S}}^*, \mathcal{T}/\sim_{\mathbb{S}^* \cup p_i})$ and set a threshold t that quantifies what we consider the minimal acceptable number of changes for a pattern to be chosen.

This technique will take longer to evaluate than the one shown before since the clustering process needs at least quadratic running time and for the Rand-index $\frac{n(n-1)}{2}$ pairwise decisions have to be made. It does have the advantage of using information about the size and composition of the blocks and not only about their number though.

3.3 Inference of patterns Φ_I

The first selection technique we showed strictly evaluates whether patterns can be described by a combination of others while the second one evaluates the effect of combining instances that differ in only one pattern. A third option is also possible in which a machine learning technique such as a rule-based learner is used to evaluate the possibility of predicting the presence/absence of a pattern based on the presence of previously chosen patterns. While this will never lead to a perfect model[1], a pattern whose presence or absence is correctly predicted on the majority of instances can be considered as not adding much information.

Given a pattern set $\mathbb{S}^* = \{p_1, \ldots, p_k\}$, a new pattern p_{k+1} and the database $\mathcal{T}$, we identify each transaction t_i with its binary feature vector $\overrightarrow{f_{\mathbb{S}^*}}(t_i) = \langle p_1(t_i), \ldots, p_k(t_i) \rangle$ and label it with $c(t_i) = p_{k+1}(t_i)$. We use a learner[2] to induce a hypothesis $h : X \mapsto \{0, 1\}$ where $X = \{\overrightarrow{f_{\mathbb{S}^*}}(t) | t \in \mathcal{T}\}$ and define the measure $\Phi_I(\mathcal{T}, \mathbb{S}^*, p_{k+1}) = 1 - \frac{|\{t_i | h(\overrightarrow{f_{\mathbb{S}^*}}(t_i)) = c(t_i)\}|}{|\mathcal{T}|}$.

Note that in this case *all* instances are represented as feature vectors including duplicates w.r.t. the feature vectors. This means that a feature having only marginal effect on large parts of the instance space will be predicted with high accuracy while a feature that e.g. splits the largest block in half will have far less accuracy, thus having a better chance of being chosen.

3.4 Ordering relations

As we mentioned before, the second important issue in the instantiations of our general approach is the ordering relation used. When working with frequent itemsets two simple *types* of orderings are possible based on support and on length of the itemsets, respectively. In each case we can use two *directions* of ordering, either ascending or descending. This leads to a total of four different orderings: support ascending ($s^{\uparrow}$), support descending ($s^{\downarrow}$), length ascending ($l^{\uparrow}$), and length descending ($l^{\downarrow}$) evaluated in the experimental section.

4. Experimental Evaluation

To evaluate the presented approach we used pattern sets from five UCI itemset mining tasks, harvested using an APRIORI implementation [1] with different minimum support thresholds. We obtained closed pattern sets of a size as shown in Table 1.

Table 1 also reports the minimum and the maximum number of patterns needed to induce the same partition as is induced by the full set of patterns $\mathbb{S}$. The low number is produced by either the $s^{\downarrow}$ or $l^{\uparrow}$ order, the large number by one of the other two.

For each of the data sets we evaluated two minimum support thresholds σ. Due to a less than efficient implementation, the subset selection for the 10% setting on the mushroom data set, which produced 16000+ closed patterns, didn't finish in time and is not reported here. As Table 1 shows even setting σ to quite large values [3] and restricting the result to *closed* patterns, removing quite some redundancy, the size of the pattern sets are still too large for use by a human expert. It can be observed that reducing the set

[1]Due to the rejection of patterns that do not effect a change in the partition.

[2]The JRip implementation of WEKA

[3]Other work often reports values of only 1%

Dataset	TicTacToe		Mushroom	Breast-Cancer		Vote		Primary-Tumor	
Size	958		8124	286		435		339	
min support	$\sigma = 10$	$\sigma = 5$	$\sigma = 25$	$\sigma = 5$	$\sigma = 1$	$\sigma = 25$	$\sigma = 10$	$\sigma = 25$	$\sigma = 10$
c. patterns	191	951	694	1018	6358	2063	15433	4873	17384
Φ_Q	5	6	3	3	3	5	2	3	3
Φ_I	3	3	3	4	4	3	3	2	6
Φ_C	2	2	2	2	2	2	2	2	2
max number blocks	958		1180	266		337	342	258	275
min number patterns	17		21	23		26		19	
max number patterns	160	78	71	133	231	161	229	88	141

Table 1. Data sets and the size of the according *closed* pattern sets, the smallest reduced sets for each measure, and the minimum and maximum number patterns for the maximal partition as well as its cardinality.

to the patterns needed to recreate the partition gives a large reduction for the "right" orders.

We used the three selection measures and four orders described in the preceding section. The measure Φ_C using *clustering* turned out to be the most most expensive in terms of computing power. The *quotient* measure Φ_Q was rather fast, leaving the measure Φ_I employing a *learning algorithm* in the middle. For each of the three techniques a threshold t has to be supplied which – although indirectly – determines the size of the resulting pattern set $\mathbb{S}^*$. We used the thresholds $\{0, 0.01, 0.03, 0.05, 0.1, 0.2, 0.3, 0.4\}$, obtaining one reduced pattern set per threshold for each of the itemsets. The intervals between different thresholds thus become wider the tighter the thresholds are, since we work on the assumption that tightening the thresholds for relaxed values will have a larger effect.

Figure 2 shows $\frac{|\mathcal{T}\sim^*_{\mathbb{S}}|}{|\mathcal{T}\sim_{\mathbb{S}}|}$ plotted against $|\mathbb{S}^*|$ for each of the selection methods, with each curve corresponding to an order. Additionally a curve $\frac{2^{|\mathbb{S}^*|}}{|\mathcal{T}\sim_{\mathbb{S}}|}$ is shown for comparison. These particular plots were derived on the *voting-record* set with $\sigma = 25\%$. The x-axis is scaled logarithmically to make the differences for the smallest pattern sets more visible since many methods converge there. As these plots compare the ability to recover a partition of equal informativeness as the original pattern set to the size of the reduced set, points/curves closer to the upper left corner can be considered better.

The first observation to be made is that the "high-support" orderings ($s^{\downarrow}$ and $l^{\uparrow}$) perform very similar to each other as do the "low-support" orderings ($s^{\uparrow}$, $l^{\downarrow}$). Additionally the latter induce larger $\mathbb{S}^*$ for the same size of the partition when compared to the former.

Second, the reduction in patterns is for most settings not linear to the decrease in blocks, meaning that our approach doesn't need to sacrifice too much in information about the data set to improve comprehensibility by the user.

Third, there are two very distinct differences between Φ_C, and Φ_I and Φ_Q. On the one hand, in the area where the thresholds are rather loose, leading to relatively large $\mathbb{S}^*$, Φ_I and Φ_Q show a smooth curve that corresponds to the small increases in the threshold setting while Φ_C reduces the cardinality of $\mathbb{S}^*$ rapidly. It also induces a coarser (less blocks but of higher cardinality) partition doing so. On the other hand, once the threshold is raised above a certain value, large changes in the threshold have less effect on Φ_C than on the other two measures, resulting in a smooth curve for small sets for Φ_C and more abrupt changes for Φ_I and Φ_Q. Ultimately, the threshold of 0.4 leads to a reduction of $|\mathbb{S}^*|$ to 2, essentially the lowest reasonable cardinality, while Φ_I and Φ_Q produce larger sets at that threshold.

Regarding the comparison of Φ_I to Φ_Q, it should be noted that for the "low-support" orderings Φ_I shows a steeper descent for low thresholds which means that it is quicker in recovering the size of the partition.

The behavior for Φ_Q is not surprising, since it measures the ability of patterns to split existing blocks into sub-blocks. It is to be expected that raising the threshold in little steps will exclude only a few patterns compared to the setting before so that smooth curves are produced. In contrast, large steps will ratchet up the selectivity quite a bit, leading to larger drops in cardinality.

As explained before Φ_I would e.g. reject a pattern that splits only one (or few) block(s), especially if they are small since then the pattern could be reliable inferred on the majority of instances. On the other hand, patterns that would be rejected by Φ_Q as not adding enough blocks might be accepted by Φ_P if the "right" blocks are split. This means that especially for relatively low thresholds less patterns are removed from consideration, thus allowing to quicker find near-optimal patterns regarding partitioning the data.

Finally, the main operational difference between Φ_C and the other two measures lies (as mentioned in Section 3) in the fact that the effect of rolling back an "old" decision affects acceptance of a new pattern. This means that two types of patterns will be accepted: those that split many blocks (as it should be) and those that evaluate the same for similar pattern combinations.

Assuming a threshold of 0.01 and a pattern that splits a single block.

- Φ_Q will very likely accept
- Φ_I will accept if the pattern is not too *uniform* over the other blocks
- Φ_C will accept if the pattern is not too *diverse* over the other blocks

Uniformity over blocks is less likely than diversity. So Φ_I will potentially accept more patterns than Φ_C. Also, Φ_C will accept less patterns than Φ_Q (at least if the increase in number of blocks required for acceptance by Φ_Q is not large). This should explain why there is such a drop-off for small increases in the beginning. Later when large changes in the *Rand* are required, patterns are aided by the fact that roll-back of earlier decisions might help them in being accepted. Based on these observations we perform the following comparisons:

4.1. Support$^{\downarrow}$ *vs* Length$^{\uparrow}$

The similarities of these two orders are not that surprising, given that in pattern mining short (general) patterns usually match relatively many transactions, i.e. they have high support. This, however, does only hold to a certain degree[4].

We are interested in how similar the $\mathbb{S}^*$ induced by those two orderings are. This is evaluated for each method w.r.t. the orders in question. We use the *Rand* index as a similarity measure for any two pattern sets, which will decrease if there are a different number of blocks or instances are partitioned differently. Since a partition induced by a $\mathbb{S}^*$ constructed using one order will not necessarily have a corresponding partition of equal cardinality, we compare to the two closest partitions (one with higher, one with lower cardinality) and consider the larger *Rand* index.

The partitions induced using the two orders are very similar for all three methods, especially for permissive thresholds reaching *Rand* values in excess of 0.95. The similarity stays high for Φ_I and Φ_Q while Φ_C derives rather different partitions for tight thresholds.

4.2. Ascending$^{(\uparrow)}$ *vs* Descending$^{(\downarrow)}$

Obviously, the patterns selected from descending and ascending orderings will not be the same. It is however possible that patterns selected from one ordering can be inferred from the other pattern set. To evaluate this, we compare pattern sets resulting from descending vs. ascending orderings while keeping all other variables fixed. Similarly to the

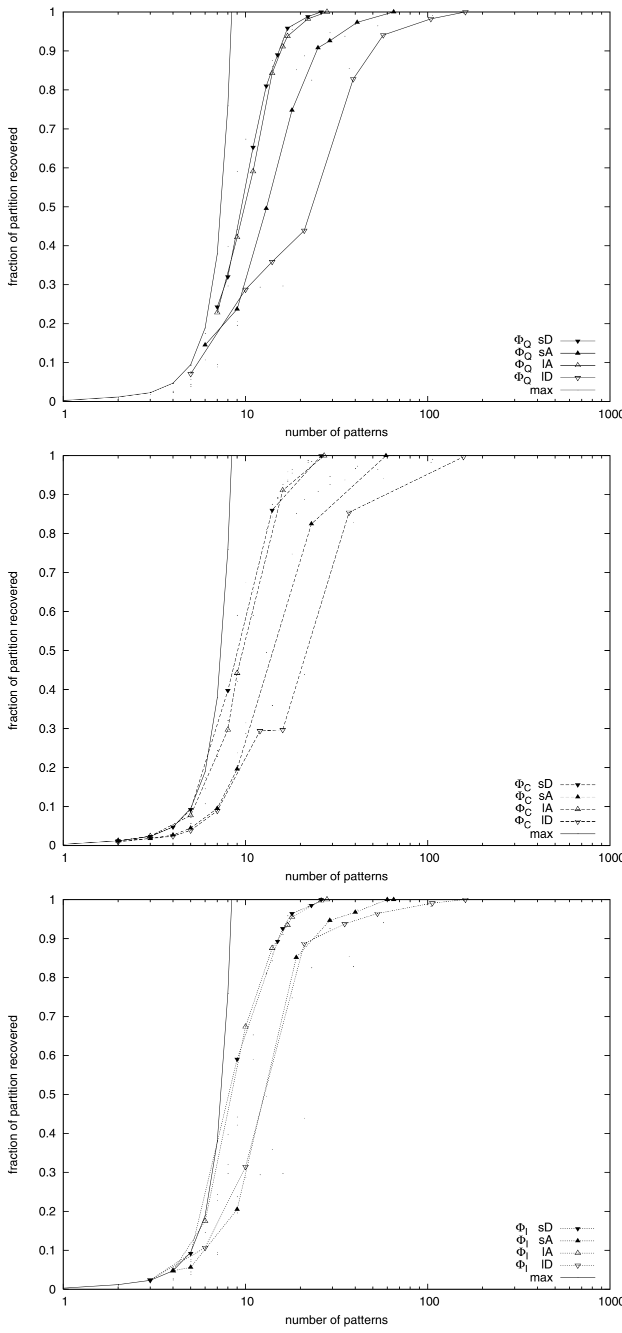

Figure 2. Plots for the three measures on Voting-record $\sigma = 25$ showing nicely the characteristics present in almost all data sets analyzed. The *max* in each plot indicates the maximum number of blocks that can be induced by the given number of patterns.

[4]Some patterns consisting of 2 items might have higher support than some single item patterns for instance.

inference selection step, we induce a model on $\mathcal{T}$ for each pattern of $\mathbb{S}_1^*$ using the patterns of $\mathbb{S}_2^*$ as features, and vice versa. The minimum accuracy for each pattern set is a quantifier of how well one pattern set can be inferred by another one.

Indeed, for $\mathbb{S}^*$ inducing maximal partitions on the data set we observe very high accuracies regarding inference. When the $\mathbb{S}^*$ induce partitions with fewer blocks, the inference accuracy decreases as well. However, $\mathbb{S}^*$ obtained using a $s^{\downarrow}$ ordering can usually infer larger sets obtained using a $s^{\uparrow}$ ordering. This observation suggests the following: to recover the complete partition with a compact (understandable) $\mathbb{S}^*$, choose e.g. $s^{\downarrow}$. To discover information not encoded in $s^{\downarrow}$-sets, tighten the thresholds and use $s^{\uparrow}$, since now $s^{\uparrow}$-sets become manageable.

Again, we also compared the partitions induced by the $\mathbb{S}^*$ using the *Rand* index. Surprisingly, Φ_C shows very similar behavior to the $s^{\downarrow}$-$l^{\uparrow}$ comparison. The difference between e.g. $s^{\uparrow}$- and $l^{\uparrow}$-induced partitions seems to be negligible at least when compared to an $s^{\downarrow}$-induced partition. Φ_I is particularly stable in this comparison, never having a *Rand* value of less than 0.85. This essentially mirrors the observation that these pattern sets can be inferred quite well from each other.

4.3. Comparing the prediction quality of selection methods

As we have seen, the size of reduced pattern sets is small enough that it would be possible for a human to inspect them. We have argued however that reducing the pattern set also helps machine learning algorithms that use the binary vector representation to learn a model. We are especially interested in whether the smallest sets (easiest to inspect for a human) give sensible accuracy results. To evaluate this we use C4.5 to induce models on the binary feature representation obtained by using the different $\mathbb{S}^*$ and estimate their classification accuracy, via ten-fold cross-validation. We also learn models on the binary vector set constructed using $\mathbb{S}$ and on the original attribute-value representation.

An example for accuracies attainable with different selection measures and orderings is shown in Figure 3. The orderings from left to right are $s^{\downarrow}$, $s^{\uparrow}$, $l^{\downarrow}$, $l^{\uparrow}$. It can be seen that the ”high support” orderings are performing better than their respective ”low support”-counterparts. While this does not hold of for *all* data sets it is a rather common trend.

We use a second figure (Figure 5) for comparing the accuracies of C4.5 models on four representations. These are the original attribute-value representation of the data, and binary vectors which are created using $\mathbb{S}$, the best-performing $\mathbb{S}^*$ selected by an instantiation of our approach, and *miki*s, respectively. At the top of the each bar representing the best $\mathbb{S}^*$, a number denotes the size of the corresponding pattern set. The most relevant result of this comparison is that usually neither the binary vector representation derived from the whole $\mathbb{S}$, nor the one based on the $\mathbb{S}^*$ that gives the maximal partition are best-suited for the machine learning algorithm. Instead, for all data sets, a reduced pattern set gives the best accuracy after cross-validation and pruning. This supports our assumption that too many features only lead to an over-fitting effect and don’t benefit the learner. The size of the corresponding pattern sets is so small that they should be easily interpretable by the user. An example for the *tic-tac-toe* data set is given in Figure 4.

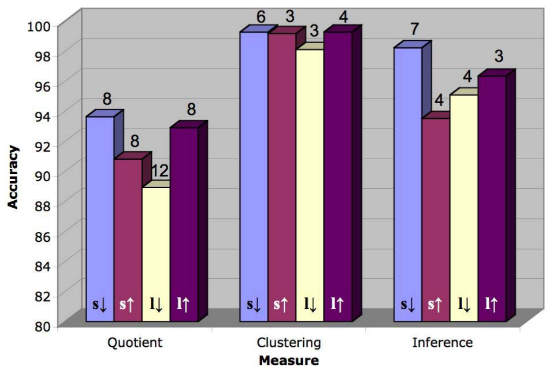

Figure 3. Best cross-validated C4.5 accuracies (all orderings for each measure) on TicTacToe ($\sigma = 10$%)

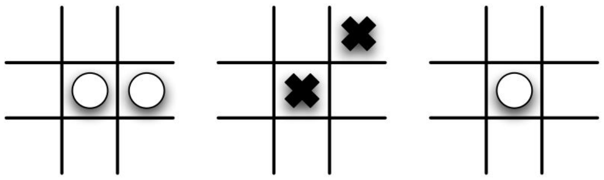

Figure 4. A visualisation of $\mathbb{S}^*$ for Tic-Tac-Toe $\sigma = 5$% using $s^{\downarrow}$, Φ_C, and a threshold of 0.4.

The attribute-value representation still proves to be more expressible than the pattern representations for most cases, a problem that could probably be alleviated with the use of a more lenient support threshold. Furthermore, no single ordering or selection method does distinguish itself and selection thresholds vary between 0.03 and 0.4. Keep in mind however that we didn’t aim to maximize predictive accuracy and our illustrative instantiations are not meant to exhaust the entire issue.

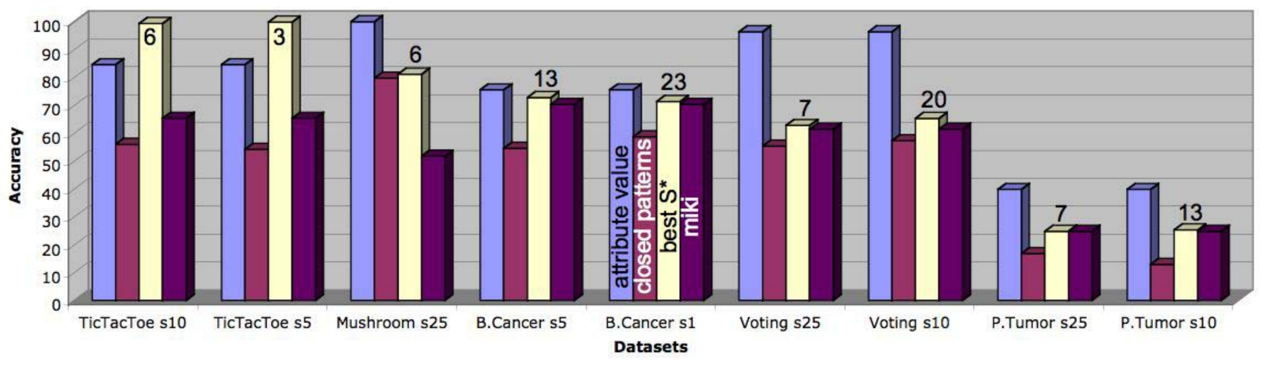

Figure 5. C4.5 cross-validated accuracies for attribute-value representation, closed-set binary representation, best $\mathbb{S}^*$, and *miki* (2 patterns)

4.4. Comparison to pattern teams

Finally, we also used the algorithms by Knobbe *et al.* to mine pattern teams on the sets of closed patterns. Due to the rather large running times of these techniques only pattern teams of size $k = 2$ have been selected. The comparisons in this case focus on the similarity of the pattern teams to our reduced pattern sets (*Rand*-index and inference) and on the effectiveness as features for classification purposes.

We compared the maximally informative k-itemsets (*miki*) obtained with algorithms by Knobbe *et al.* to reduced sets obtained by using Φ_C with a rather strict threshold. Interestingly the sets compared exhibit very similar behavior w.r.t. to all comparisons. First, all but one consist of two patterns. Their prediction qualities are the same, and they can both equally well infer the other set. High *Rand* values furthermore show that our approach induces very similar partitions. This is interesting insofar as *miki*s are mined using a complete method maximizing mutual entropy, a measure that rewards balanced partitions, while we employ a heuristic method.

5. Related Work

There are two fields of research that our work shows similarities to: the selection of "informative" subsets from mined patterns, and feature selection/feature construction for classification purposes.

5.1. Selecting informative subsets

Much work has been done on reducing the number of patterns that are returned to the user by a mining process. The approaches can roughly be differentiated into three different categories:

5.1.1 Selecting subsets with ad-hoc properties

The earliest and in a sense simplest approach to reducing sets of frequent patterns lies in selecting subsets of this set that allow for the reconstruction of the entire set. The earliest such technique reduces a set of frequent patterns to their (positive or negative) borders [10]. The information stored in the borders allows to quickly determine what the syntactic makeup of frequent pattern is. Support information would have to be found by rescanning the data again but borders are usually a lot smaller than the complete set. Subsets that allow for the reconstruction of both syntactic makeup and support information while at the same time decreasing the size of the set of patterns to be considered include the sets of *closed, free*, and *non-derivable* patterns [12, 2, 4].The main difference to our work lies in the fact that the underlying assumption of these solutions is that **all** found patterns are of interest to the user, an assumption that we do not make. Also, as can be seen in our experiments, even those property-restricted sets can be large and there is still a lot of redundant information present.

5.1.2 Summarizing frequent patterns

On the other hand, to give the user a good idea of the information encoded in the patterns without focusing too much on the exact composition and support properties, it is possible to summarize frequent patterns. To this end, frequent patterns (and association rules) have been clustered [9], usually by defining a similarity measure involving the syntactic makeup and regions of the data covered. To the user then representatives of the found frequent pattern clusters are returned with the assumption that analysis of those representatives will allow the user to gain information. The first step of the general technique we introduce does something similar in that for all patterns matching the exact same instances only one pattern will be returned. A difference here is that no patterns that encode the same information as a *combination* of patterns will be returned, which would probably form their own cluster(s) in the existing approaches. Yet again, the working assumption of summarizing frequent patterns is that the user is interested in the information encoded by **all** patterns.

5.1.3 Selecting informative patterns

Finally, approaches that are most similar to our work are relatively new and attempt to select subsets of *informative* patterns with informativeness being defined w.r.t. some task at hand. This could for instance be lossless compression of the database using an MDL criterion [11], or removal of redundant information according to the joint entropy criterion [6]. The work of Siebes *et al.* (developed independently and a bit before our technique) follows very similar ideas and intuitions about the meaning and use of patterns, and the naïve variant of their algorithms, given the right modifications, could be formulated as an instantiation of our general idea. A main difference is that we cannot expect to find patterns that allow a lossless compression of the database, which on the other hand means that the reduced pattern sets of Siebes *et al.* are larger than ours. The approach of Knobbe *et al.* is different in that their focus is very strongly on low-redundancy patterns. The upside of this that their pattern teams show a lot less redundancy than the sets we derive for loose threshold settings. The downside is that searching the space of possible pattern teams is far larger which is why their approach needs an additional parameter, namely the size of the pattern teams (which is significantly below 10). As we have shown in the experimental session, we do, for strict settings of our thresholds arrive at pattern sets very similar to Knobbe's pattern teams

5.2. Feature selection/construction for classification

5.2.1 Feature selection

Approaches to feature selection involve for instance relevancy constraints in subgroup discovery [8]. We mention this approach in particular since the underlying idea is similar to ours, with the restriction to class-labeled data. Many other feature selection approaches also select relevant subsets of all features, with many techniques using heuristic methods. In general, measures such as an improvement of accuracy on some testing data could be used in our general algorithm. However, our approach is not limited to class-labeled data, but can easily be used *unsupervised* and the focus is not necessarily better classification accuracy but the removal of redundancy.

Another class of feature selection methods that is related to our technique are so-called "wrapper" approaches. In these usually features are selected, handed to an evaluation algorithm and then on their inclusion decided. An example that seems very close to our work is [5] in that the wrapper uses unsupervised learning instead of an induction algorithm. A main difference in such approaches lies in the fact that usually all features for inclusion are considered, making the search space again rather large. By using a heuristic approach, we can handle far more patterns than such an approach could.

5.2.2 Feature construction

A somewhat similar approach to this work is embodied by the KFOIL[7] system, as well as by TREE^2[3]. The main principle of these approaches is that patterns are constructed while a classifier is built. This means that obviously every pattern used in the final classifier is dependent on the other ones and the information contained in them. Since those approaches proceed in such a way that they greedily construct the next best feature and the focus is again classification accuracy, there are also quite a few differences to our technique.

6 Discussion and Conclusions

The focus of this work is the reduction of result sets of pattern mining operations to sets of what we called "valuable" patterns - pattern sets with little redundancy among their members, capturing most of the characteristics of the underlying data, and being of a cardinality still accessible to humans. To achieve this task we introduced a general heuristic algorithm and showed several instantiations with selection criteria and orderings w.r.t. which the patterns are processed.

In the experimental evaluation on several UCI data sets we showed that we achieve our goal of significantly reducing the sets of closed patterns mined on them. Given a well-chosen ordering (support descending or size ascending), cardinality of reduced pattern sets is in most cases below or around 30, a size that should still be interpretable by humans.

Additionally, while tightening the thresholds has the effect that the cardinality of reduced sets decreases, the decrease in number of blocks is less or equal to this. This means that the number of patterns for consideration can be reduced without losing too much information. Furthermore, reducing the number of patterns does not necessarily lead to a decrease in classification accuracy when compared to larger sets or the original representation. Quite contrary, on several occasions larger pattern sets do "confuse" the machine learner, leading to overfitting.

To summarize, we manage to achieve our three subgoals with different instantiations of our general algorithm. All three selection measures have their merits, with Φ_C being computationally most expensive but also showing very good reduction effects while the Φ_I and Φ_Q recover more of the original partitioning at the expense of less reduction in set cardinality.

There is of course ample opportunity for future work. The flexibility of our approach allows for the definition of more exotic orderings - for instance could a human expert set a pattern (or a number of patterns) as seed(s) and define a dynamic ordering that changes given the current pattern set (e.g. based on city block distance to the current blocks). Additionally, other measures are also possible such as the MDL criterion used in Siebes *et al.*'s work or accuracy based selection measures that mirror "classical" feature selection.

So far our technique remains a post-processing step. And while the generality of the algorithm and the different measures it allows make it hard to incorporate the entire selection in a mining operation itself, at least the partition cardinality might be turned into a constraint that can be pushed into the mining step itself.

Acknowledgments We thank Arno Knobbe for interesting discussions and providing the algorithm used for finding *miki*s as well as Luc De Raedt and Arno Siebes for helpful comments and discussion on the topic. We are also grateful for the help of Siegfried Nijssen and Ulrich Rückert and the anonymous reviews which helped us improve this work. The authors were supported by the IQ project (European Union Project IST-FET FP6-516169) and by a doctoral research grant by the KU Leuven research fund.

References

[1] C. Borgelt. Recursion pruning for the apriori algorithm. In R. J. B. Jr., B. Goethals, and M. J. Zaki, editors, *FIMI*, 2004.

[2] J.-F. Boulicaut and B. Jeudy. Mining free itemsets under constraints. In M. E. Adiba, C. Collet, and B. C. Desai, editors, *IDEAS*, pages 322–329, 2001.

[3] B. Bringmann and A. Zimmermann. Tree^2 - decision trees for tree structured data. In A. Jorge, L. Torgo, P. Brazdil, R. Camacho, and J. Gama, editors, *PKDD*, pages 46–58. Springer, 2005.

[4] T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. In T. Elomaa, H. Mannila, and H. Toivonen, editors, *PKDD*, pages 74–85. Springer, 2002.

[5] J. G. Dy and C. E. Brodley. Feature selection for unsupervised learning. *Journal of Machine Learning Research*, 5:845–889, 2004.

[6] A. J. Knobbe and E. K. Y. Ho. Pattern teams. In J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *PKDD*, pages 577–584. Springer, 2006.

[7] N. Landwehr, A. Passerini, L. D. Raedt, and P. Frasconi. kfoil: Learning simple relational kernels. In *AAAI*. AAAI Press, 2006.

[8] N. Lavrac and D. Gamberger. Relevancy in constraint-based subgroup discovery. In J.-F. Boulicaut, L. D. Raedt, and H. Mannila, editors, *Constraint-Based Mining and Inductive Databases*, pages 243–266. Springer, 2004.

[9] B. Lent, A. N. Swami, and J. Widom. Clustering association rules. In W. A. Gray and P.-Å. Larson, editors, *ICDE*, pages 220–231, 1997.

[10] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Min. Knowl. Discov.*, 1(3):241–258, 1997.

[11] A. Siebes, J. Vreeken, and M. van Leeuwen. Item sets that compress. In J. Ghosh, D. Lambert, D. B. Skillicorn, and J. Srivastava, editors, *SDM*. SIAM, 2006.

[12] R. Taouil, N. Pasquier, Y. Bastide, and L. Lakhal. Mining bases for association rules using closed sets. In *ICDE*, page 307, 2000.