Knowl Inf Syst (2009) 18:61–81
DOI 10.1007/s10115-008-0136-4

REGULAR PAPER

# One in a million: picking the right patterns

**Björn Bringmann · Albrecht Zimmermann**

Received: 29 October 2007 / Revised: 24 December 2007 / Accepted: 29 January 2008 /
Published online: 28 March 2008
© Springer-Verlag London Limited 2008

**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 leaning 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 two general heuristic algorithms—Bouncer and Picker—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 both techniques succeed 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 developed solutions are very well suited for the goals we aim at.

**Keywords** Data mining · Post processing · Pattern reduction

## 1 Introduction

Pattern search is one of the main topics in data mining. In the past few 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

B. Bringmann · A. Zimmermann (✉)
Departement Computerwetenschappen, Katholieke Universiteit Leuven,
Celestijnenlaan 200a, 3001 Heverlee, Belgium
e-mail: Albrecht.Zimmermann@cs.kuleuven.be

B. Bringmann
e-mail: Bjoern.Bringmann@cs.kuleuven.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 inspected by 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* [15] 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. 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. Insights into the effects and, the semantic, of the measures lead us to improve the heuristics and formulate an upper-bound technique that is independent from orderings. When evaluated on several data sets the techniques reduce the set of closed patterns severely.

The rest of the 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 Sect. 3, we describe several instantiations of the first algorithm, motivating the different selection strategies. Based on interpreting the measures and selection effects, we refine the heuristic in Sect. 4, and develop a second algorithm. The new formulations allow for the calculation of *upper bounds*, removing the need for pre-defined orderings. Their derivation is outlined in Sect. 5. In Sect. 6, 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 Sect. 7 we discuss related work and finally, we conclude in Sect. 8.

## 2 The general algorithm

As stated earlier, the main objective 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 transactional 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 objective is the following:

**Given** a *set* $\mathbb{S}$ of *patterns* $p_i$, the database $\mathcal{T}$, and a redundancy measure $\Phi$ **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 $\Phi$.
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 $p : \mathcal{T} \to \{true, false\}$. 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. Since it will be very handy lateron, we define the support measure for a pattern $p$ and a set of transactions $b \subseteq \mathcal{T}$ as usual

$$\sigma_p(b) = |\{t \in b : p(t) = true\}|$$

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. Taking into account the actual syntactical composition and support of the patterns in the selection measure would require measures that are specifically bound to certain types of pattern languages. Since this approach aims at a more general applicability, namely all patterns that can be defined as absent or present further details on syntacital composition or similar are *not* of interest here.

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 for some of the methods we will introduce. Furthermore, a domain expert can also incorporate his knowledge into the process by selecting a set of patterns and using the algorithm to saturate this set with patterns of value according to the method employed. 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.

To accomplish the goal, we define a measure $\Phi$ as

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

The measure is supposed to rate the value of adding a pattern $p$ to the currently selected subset $\mathbb{S}^*$. The higher the value of $\Phi$, the more valuable it would be adding the pattern $p$ 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 subgoals stated earlier 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 $k$ [8]. 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 first, rather general heuristic algorithm called BOUNCER to compute $\mathbb{S}^*$ given $\mathbb{S}$, $\mathcal{T}$, an order $<$, $\Phi$, and some threshold $t \in [0, 1] \subset \mathbb{R}$. It is shown as Algorithm 2.1.

**Algorithm 2.1** The BOUNCER algorithm

```
1: for i = 1 to |S| do
2:   if |T/ ~(S*∪p_i)| > |T/ ~S*| then
3:     if Φ(T, S*, p_i) ≥ t then
4:       S* = S* ∪ {p_i}
5: return S*
```

Similar to a doorman at a venue the algorithm considers the candidates in a sequential order and every candidate only once. On the basis of criteria expressed via the measure $\Phi$, a candidate is accepted or refused. More formally 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 compared to the partition induced by $\mathbb{S}^*$. If there is no change, the pattern is rejected, thus reducing redundancy (subgoal 2). If there is change, the measure $\Phi$ is evaluated against the threshold and if the threshold is passed, the pattern is chosen.

To give some intuition about the process, consider Fig. 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 $\mathbb{S}^*$ 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.

Obviously there are—apart from $\mathcal{T}$—two major points influencing the result $\mathbb{S}^*$. First, there is the measure $\Phi$ 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.

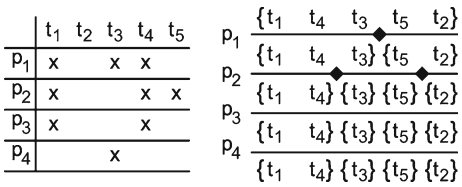


**Fig. 1** Partitions induced by patterns. The *left table* shows which patterns $p_i$ occur in the transactions $t_j$. The *right* shows how the patterns split up the data set. Four binary patterns can induce at most 16 blocks. This combination of patterns and instances yields only four blocks, however

## 3 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.

### 3.1 Partition size quotient $\Phi_Q$

While rejection of patterns that do not change the partition will effectively cut down on the number of patterns retained already, there is a 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 us 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

$$\Phi_Q(\mathcal{T}, \mathbb{S}^*, p) = 1 - \frac{|\mathcal{T}/\sim_{\mathbb{S}^*}|}{|\mathcal{T}/\sim_{(\mathbb{S}^* \cup p)}|}.$$

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 $\Phi_C$

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. Thus, a clustering algorithm can be denoted as a function $\mathcal{C}(\mathcal{T}, k)$ that returns a partition $\mathbb{P}$ of $\mathcal{T}$ consisting of $k$ blocks.

Let us 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, the sub-block in which the new pattern is present will have a distance of 2 to all blocks where it is absent (except to its sibling), and vice versa. 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 earlier, 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 can be defined—$c_{ij}$ 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

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

We define

$$\Phi_C(\mathcal{T}, \mathbb{S}^*, p) = 1 - \text{Rand}(\mathcal{T}/\sim_{\mathbb{S}^*}, \mathcal{C}(\mathcal{T}/\sim_{\mathbb{S}^* \cup p}, |\mathcal{T}/\sim_{\mathbb{S}^*}|))$$

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 $\frac{1}{2} \cdot n(n-1)$ 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.

Please note that $\Phi_C$ will always yield 0 if $\mathbb{S}^*$ is empty, since the clustering would then have to create a partition with *one* block. Thus its Rand-index is zero. To overcome this problem the measure will return 1 for any pattern that can create two blocks if $\mathbb{S}^*$ is empty.

### 3.3 Inference of patterns $\Phi_I$

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 candidate pattern $p$ 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 $c(t_i) = p(t_i)$. We use a learner[2] to induce a hypothesis $h : X \mapsto \{0, 1\}$ where $X = \{\overrightarrow{f_{\mathbb{S}^*}}(t) \mid t \in \mathcal{T}\}$ and define the measure

$$\Phi_I(\mathcal{T}, \mathbb{S}^*, p) = 1 - \frac{|\{t_i : h(\overrightarrow{f_{\mathbb{S}^*}}(t_i)) = c(t_i)\}|}{|\mathcal{T}|}.$$

Note that in this case *all* instances are represented as binary 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.

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

[2] The J48 implementation of WEKA.

### 3.4 Ordering relations

As we mentioned earlier, the second important issue in the instantiations of the BOUNCER algorithm 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 Refining the heuristic

In the BOUNCER algorithm a pattern that just exceeds the threshold will be chosen, neglecting the possibility of any other pattern later in the sequence reaching a much higher score, and thus introducing a more desireable split. To address this issue, in each selection step each pattern has to be evalulated in order to select the pattern with the highest score. However, except for $\Phi_Q$, evaluating the score of all pattens in every selection step introduces a much higher computational complexity.

Much of this complexity arises from the use of a classification/clustering algorithm in two of the selection steps. The classification algorithm used in $\Phi_I$ allows the rejection of patterns that only split off small blocks, and usage of a sophisticated algorithm (such as an SVM) or evaluation schemes such as cross-validation might correct over-fitting effects. The use of an unpruned classifier evaluated on training-data—the most efficient method—equates to simple counting.

Similarly, while we originally chose the clustering technique to introduce an element of chance to offset the ordering effect, it can be questioned how valuable this is. Furthermore, given the systematic dissimilarity of blocks, a maximally different clustering could be derived from analysis of the partition-tree.

Therefore, a possible solution to the effects of the sequential processing lies in foregoing the algorithmic components within our measures. Instead, we propose new formulations that allow the calculation of upper bounds for patterns, thus allowing for pruning, and ideally, enough efficiency-gain to compensate for the repeated evaluation of patterns.

**Algorithm 4.1** The PICKER algorithm

```
1: repeat
2:    p_c = p_1
3:    for i = 2 to |𝕊| do
4:       if |𝒯/∼_(𝕊*∪p_i)| > |𝒯/∼_𝕊*| then
5:          if Φ(𝒯, 𝕊*, p_i) > Φ(𝒯, 𝕊*, p_c) then
6:             p_c = p_i
7:    if Φ(𝒯, 𝕊*, p_c) ≥ t then
8:       𝕊* = 𝕊* ∪ {p_c}
9: until Φ(𝒯, 𝕊*, p_c) < t
10: return 𝕊*
```

The decision to employ a more exhaustive evaluation gives rise to the PICKER algorithm, shown as Algorithm 4.1. Differing from the BOUNCER algorithm, it is more thorough in its evaluation of patterns, greedily selecting the *best*, not the first pattern exceeding the threshold.

In the following sections, we will introduce the reformulated measures before explaining the upper bound mechanisms possible.

### 4.1 Reformulating the partition size quotient $\Phi_Q$

The first measure, $\Phi_Q$, in a sense quantifies how fully a new pattern fulfills its potential to make the existing partition more fine-grained. It is formulated as

$$\Phi_Q(\mathcal{T}, \mathbb{S}^*, p) = 1 - \frac{|\mathcal{T}/\sim_{\mathbb{S}^*}|}{|\mathcal{T}/\sim_{(\mathbb{S}^* \cup p)}|}$$

In this formulation, it is measured how large the amount of unchanged blocks is. A reformulation based on the same idea would be

$$\Phi_{Q^+}(\mathcal{T}, \mathbb{S}^*, p) = \frac{|\mathcal{T}/\sim_{(\mathbb{S}^* \cup p)}| - |\mathcal{T}/\sim_{\mathbb{S}^*}|}{2 \cdot |\mathcal{T}/\sim_{\mathbb{S}^*}|}.$$

This $\Phi_{Q^+}$ returns the fraction of newly introduced blocks by the candidate pattern over the maximal amount of blocks that could result from introducing a new pattern. Thus, if the candidate pattern does not introduce a new split, $\Phi_{Q^+}$ equals 0. The maximum amount of splits a pattern can introduce equals the number of blocks, then the measure would return $\frac{1}{2}$.

### 4.2 Removing the element of chance from $\Phi_C$

The way $\Phi_C$ is formulated the blocks formed after applying the candidate pattern are recombined using an agglomerative clusterer and then compared to the old partition using the Rand-index. Since a trivial approach to this would simply recover the old partition, care is taken that non-siblings are combined if possible, thus assuring the introduction of change. The same could be effected by comparing the old and the new partition via the Rand-index. This would remove the element of chance, and reduce the computational complexity—resulting in a lower execution time. Analogous to the measures introduced earlier we define this Rand-based measure as

$$\Phi_{C^+}(\mathcal{T}, \mathbb{S}^*, p) = \frac{2}{|\mathcal{T}| \cdot (|\mathcal{T}| - 1)} \cdot \sum_{b \in \mathcal{T}/\sim_{\mathbb{S}^*}} \sigma_p(b) \cdot (|b| - \sigma_p(b))$$

This new measure is based on the observation that blocks can be considered separately. For each block $b$ the number of pairs that will be split is just the product of the number of instances that are covered by the candidate pattern and those which are not. Since splits are never reversed during the selection process, instances that are in different blocks can never be merged into one block and hence not contribute to the Rand-index. The summation of these products over all blocks is normalized by the maximum number of pairs that can split which is $\frac{|\mathcal{T}| \cdot (|\mathcal{T}|-1)}{2}$, as mentioned earlier.

Please note that the Rand-index usually counts the number of pairs that are *not* split, whereas here we count the number of pairs that will be split, thus *inverting* the Rand-index.

### 4.3 Formulating $\Phi_I$ in closed form

Currently, a classifier is employed to find the score a pattern achieves, in the form of

$$\Phi_I(\mathcal{T}, \mathbb{S}^*, p) = 1 - \frac{|\{t_i \in \mathcal{T} : h(\overrightarrow{f_{\mathbb{S}^*}}(t_i)) = c(t_i)\}|}{|\mathcal{T}|}$$

which is equivalent to $1 -$ accuracy of the hypothesis induced by the learning algorithm. The learner we employed for $\Phi_I$ uses a decision tree mechanism, which means that it learns a set of mutually exclusive conjunctive rules. In addition, we used the tree unpruned and evaluated it on the training data such that actually the (observed) probability $\mathcal{P}(p \mid p_1, \ldots, p_k)$ is measured. This can as well be found by counting the number of occurrences of $p$ on each block.

More formally

$$\Phi_{I^+}(\mathcal{T}, \mathbb{S}^*, p) = \sum_{b \in \mathcal{T}/\sim_{\mathbb{S}^*}} \frac{|b|}{|\mathcal{T}|} \cdot \min(\sigma_p(b),\ |b| - \sigma_p(b))$$

This definition equals the *minimum error* a classifier can achieve on the training set $\mathcal{T}$ using all features contained in $\mathbb{S}^*$. Thus $\Phi_{I^+} = \Phi_I$ if the learning algorithm employed in $\Phi_I$ builds an optimal model for the supplied dataset $\mathcal{T}$. This reformulation strongly reduces the complexity and thus can be used in the PICKER algorithm.

## 5 Using bounds to reduce complexity

As argued earlier, computational complexity could be reduced further if only those patterns would be evaluated in a step that have a chance of exceeding the score of the best pattern found so far in the current step. For each of the three measures $\Phi_{Q^+}$, $\Phi_{I^+}$, and $\Phi_{C^+}$ we can define such an upper bound $\Phi^* \geq \Phi$ that is easy to evaluate. We reformulate the PICKER algorithm as PICKER$^*$ (Algorithm 5.1) using this bound.

**Algorithm 5.1** The PICKER$^*$ algorithm

```
1: repeat
2:    p_c = p_χ(1)
3:    for i = 2 to |𝕊| do
4:       if Φ*(𝒯, 𝕊*, p_χ(i)) < Φ(𝒯, 𝕊*, p_c) then
5:          updatebound p_χ(i)
6:       else
7:          if |𝒯/ ∼_(𝕊*∪p_i)| > |𝒯/ ∼_𝕊*| then
8:             if Φ(𝒯, 𝕊*, p_χ(i)) > Φ(𝒯, 𝕊*, p_c) then
9:                p_c = p_χ(i)
10:               calculate bound p_χ(i)
11:   if Φ(𝒯, 𝕊*, p_c) ≥ t then
12:      𝕊* = 𝕊* ∪ {p_c}
13: until Φ(𝒯, 𝕊*, p_c) < t
14: return 𝕊*
```

In this extended version of the algorithm the candidate patterns are sorted according to their bounds. This is represented in the pseudo-algorithm by the permutation $\chi$. Thus, the patterns with the best chance to reach a high score are evaluated first. As soon as $\Phi(p_c) > p_{\chi(i)}$, all following patterns according to the permutation cannot exceed the score of $p_c$. However, the current bound may need to be updated since the newly introduced split can change the effect of later patterns. This update is performed in constant time.

### 5.1 A bound for $\Phi_{Q^+}$

Denoting the number of splits a candidate pattern would introduce in step $k$ as $\text{splits}_k(p)$ the bound for the reformulated quotient can be expressed as

$$\Phi^*_{Q^+}(\mathcal{T}, \mathbb{S}^*, p) = \frac{2^{k-|\mathbb{S}^*|} \cdot \text{splits}_k(p)}{\text{blocks}}$$

where $\text{blocks} = splits_k(p) + |\mathcal{T}/\sim_{\mathbb{S}^*}|$ for $k = |\mathbb{S}^*| + 1$. When the bound is calculated (not updated), $p_c$ is not known yet. Nevertheless, we know that $p_c$ has to introduce at least as many splits as $p$ since otherwise $p$ would be chosen. However, if the bound is updated later on, we know the exact number of splits since the splitting pattern $p_c$ is always chosen before bounds are updated. Thus, $\text{blocks} = |\mathcal{T}/\sim_{\mathbb{S}^*}|$ if $k > |\mathbb{S}^*| + 1$.

The maximum number of splits a pattern $p$ can introduce is $2^n$ as many as it could have introduced $n$ selection steps earlier. The exponent $k - |\mathbb{S}^*|$ denotes the number of steps since the pattern was last evaluated and therefore, a new estimate on its splitting power derived.

### 5.2 A bound for $\Phi_{C^+}$

Given that a pattern $p$ would introduce a split with a Rand value of $x$, we know that it cannot introduce a split with a Rand higher than $x$ on any subpartition. The reason is the following: Since we never join blocks, the only thing that affects the Rand calculation is the splitting of blocks. However, a pattern can never split *more* than $n$ pairs on a partition derived from a partition where it could split *exactly* $n$. Given that we know this, the calculated Rand serves as an upper bound.

$$\Phi^*_{C^+}(\mathcal{T}, \mathbb{S}^*, p) = \Phi_{C^+}(\mathcal{T}, \mathbb{S}^*, p)$$

Even though this looks surprisingly simple, it is a stronger upper bound than the upper bound for the quotient, which increases in each step whereas this upper bound does not increase and hence needs no update. It is also superior in that it will allow actual pruning of patterns whose Rand index does not clear the threshold anymore.

### 5.3 A bound for $\Phi_{I^+}$

The bound for the inference measure $\Phi_{I^+}$ turns out to be similar to the upper bound for the Rand-based measure. They are however based on different concepts. Obviously, there is a trivial upper bound, in that the prediction accuracy of a pattern's occurrence can never be worse in a single block than on the entire data set. To give a concrete example: a pattern that occurs in 10% of the data will always have a prediction accuracy of at least 0.9 (for absence, in that case). Prediction can of course be better, if for instance those ten percent are contained mainly in one single block.

The upper bound for $\Phi_{I^+}$ is based on pure blocks. A block $b$ is called pure w.r.t. a pattern $p$ if $p$ either occurs in all instances in $b$ or in none. Considering a whole partition we can identify three types of blocks: pure blocks $b_+$ that contain $p$, pure blocks $b_-$ that do not contain $p$, and blocks $b_?$ that are mixed. The latter ones are the only ones where a split can still influence the predictability (and hence value) of $p$. Simplifying matters, all $b_?$ can be considered as only *one* mixed block. Introducing a new split can never make the prediction worse. Splitting a pure block changes nothing while chopping of (pure) parts from the unpure block never increases the predictive error. Hence, the current error *is* an upper bound for the

**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 of patterns for the maximal partition as well as its cardinality

| Dataset | TicTacToe | | Mushroom | Breast cancer | | Vote | | Primary tumor | |
|---|---|---|---|---|---|---|---|---|---|
| Size | 958 | | 8,124 | | 286 | | 435 | | 339 |
| min support | 10 | 5 | 25 | 5 | 1 | 25 | 10 | 25 | 10 |
| c. patterns | 191 | 951 | 694 | 1,018 | 6,358 | 2,063 | 15,433 | 4,873 | 17,384 |
| $\Phi_Q$ | 5 | 6 | 3 | 3 | 3 | 5 | 2 | 3 | 3 |
| $\Phi_I$ | 3 | 3 | 3 | 4 | 4 | 3 | 3 | 2 | 6 |
| $\Phi_C$ | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| $\Phi_{Q^+}$ | 5 | 4 | 3 | 2 | 2 | 3 | 2 | 3 | 3 |
| $\Phi_{I^+}$ | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 4 |
| $\Phi_{C^+}$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| max blocks | 958 | | 1180 | | 266 | 337 | 342 | 258 | 275 |
| min patterns | 17 | | 21 | 23 | | 26 | | 19 | |
| max patterns | 160 | 78 | 71 | 133 | 231 | 161 | 229 | 88 | 141 |

predictive error. This simply reads as

$$\Phi^*_{I^+}(\mathcal{T}, \mathbb{S}^*, p) = \Phi_{I^+}(\mathcal{T}, \mathbb{S}^*, p)$$

Similar to the upper bound for the Rand-based measure, this upper bound does not require any update.

## 6 Experimental evaluation

To evaluate the presented approach we used pattern sets from five UCI itemset mining tasks, harvested using an Apriori implementation [2] 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 (or by the measures for which we used upper bound pruning), the large number by one of the other two.

For each of the data sets we evaluated two minimum support thresholds $\sigma$. Due to a less than efficient implementation, the subset selection for the 10% setting on the mushroom data set, which produced 16, 000+ closed patterns, did not finish in time and is not reported here. As Table 1 shows, even setting $\sigma$ 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 to the patterns needed to recreate the partition gives a large reduction for the "right" orders.

For experiments, we on the one hand used the three selection measures and four orders described in Sect. 3. The measure $\Phi_C$ using *clustering* turned out to be the most expensive in terms of computing power. The *quotient* measure $\Phi_Q$ was rather fast, leaving the measure $\Phi_I$ 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

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

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 the three measures $\Phi_Q$, $\Phi_I$, and $\Phi_C$, 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 does not 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 $\Phi_C$, and $\Phi_I$ and $\Phi_Q$. On the one hand, in the area where the thresholds are rather loose, leading to relatively large $\mathbb{S}^*$, $\Phi_I$ and $\Phi_Q$ show a smooth curve that corresponds to the small increases in the threshold setting while $\Phi_C$ 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 $\Phi_C$ than on the other two measures, resulting in a smooth curve for small sets for $\Phi_C$ and more abrupt changes for $\Phi_I$ and $\Phi_Q$. Ultimately, the threshold of 0.4 leads to a reduction of $|\mathbb{S}^*|$ to 2, essentially the lowest reasonable cardinality, while $\Phi_I$ and $\Phi_Q$ produce larger sets at that threshold.

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

The behavior for $\Phi_Q$ 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 earlier, $\Phi_I$ 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 $\Phi_Q$ as not adding enough blocks might be accepted by $\Phi_P$ 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 $\Phi_C$ and the other two measures lies (as mentioned in Sect. 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.

- $\Phi_Q$ will very likely accept
- $\Phi_I$ will accept if the pattern is not too *uniform* over the other blocks
- $\Phi_C$ will accept if the pattern is not too *diverse* over the other blocks.

Uniformity over blocks is less likely than diversity. So $\Phi_I$ will potentially accept more patterns than $\Phi_C$. Also, $\Phi_C$ will accept less patterns than $\Phi_Q$ (at least if the increase in number of blocks required for acceptance by $\Phi_Q$ 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.

With respect to the re-formulations of measures, very similar behavior can be observed over all data set. The only notable difference, also reported in Table 1, lies in the fact that $\Phi_{C^+}$ selects only a single pattern for all data sets for the strictest threshold. The chance effect of the agglomerative approach therefore does change the outcome of the mining operation.

On the basis of these observations we perform the following comparisons:

### 6.1 Support$^{\downarrow}$ versus Length$^{\uparrow}$

The similarities of these two orderings 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 $\Phi_I$ and $\Phi_Q$ while $\Phi_C$ derives rather different partitions for tight thresholds.

When the results of the re-formulated measures are compared against each other, this effect is even more pronounced, with Rand values usually above or around 0.95, only dropping for the strictest threshold, 0.4.

### 6.2 Ascending$^{(\uparrow)}$ versus 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 versus ascending orderings while keeping all other variables fixed. Similarly to the *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.

We also compared partitions against each other, using the Rand index. While for $\Phi_I$, and $\Phi_C$, and their reformulations usually strong similarities are shown, the situation changes for

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

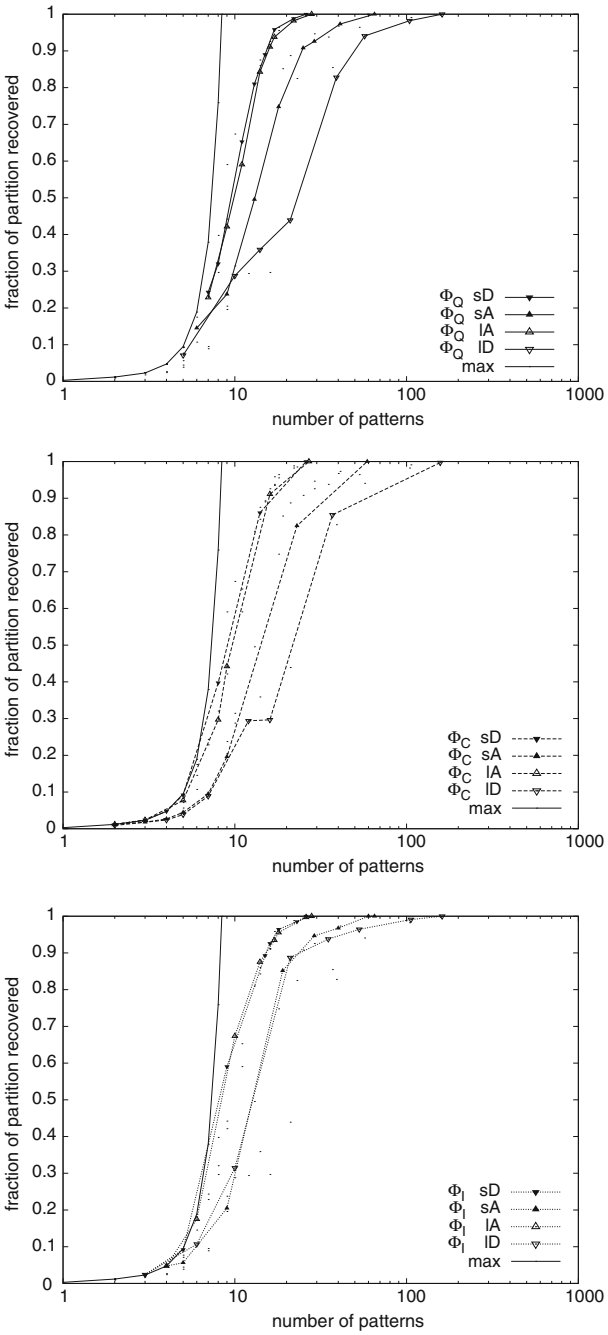


**Fig. 2** Plots for the three measures on Voting-record 25 showing nicely the characteristics present in almost all data sets analyzed. The max in each plot indicates the maximum number of blocks that can be induced by the given number of patterns

$\Phi_Q$. In fact, for data sets where one can truly speak about low-support patterns, such as Breast-Cancer and TicTacToe, Rand values drop as low as 0.5, showing that very different partitions are induced. While especially in the first steps, the other measures also choose

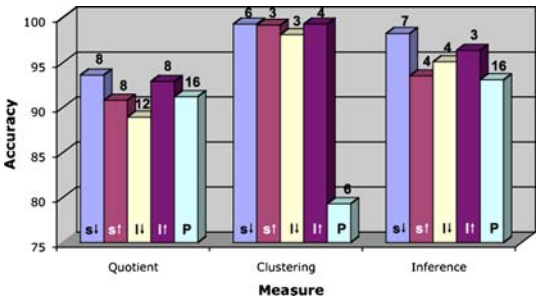


**Fig. 3** Best cross-validated C4.5 accuracies (all orderings for each measure, and the upper-bound approach) on TicTacToe (10%)

patterns with low coverage that only split off small blocks, this is corrected later. $\Phi_Q$ does not include the size of blocks in its score and therefore does not recover.

### 6.3 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 Fig. 3. The orderings from left to right are $s^{\downarrow}, s^{\uparrow}, l^{\downarrow}, l^{\uparrow}$. The column furthest to the right shows the performance of the pattern set selected by the bounded approach, since those are ordering-independent. It can be seen that the "high support" orderings are performing better than their respective "low support" counterparts. While this does not hold for *all* data sets it is a rather common trend. It is interesting to see that using a selection method with upper-bound pruning is not an effective strategy for TicTacToe. Selecting the highest-scoring pattern usually means that rather balanced blocks are formed, which is detrimental to classification performance on a data set where small subgroups are of relevance. The upper-bound selection performs closest to the greedier approach for $\Phi_Q$, the one measure where size of blocks (and thus balance) is not considered.

We use a second figure (Fig. 5) for comparing the accuracies of C4.5 models on five 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 ordering-dependent approach, $\mathbb{S}^*$ returned by the bounded selection step, and *miki*s, respectively. At the top of the 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 do not 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 Fig. 4. What can also be observed is that the weak performance of the upper-bound selection on

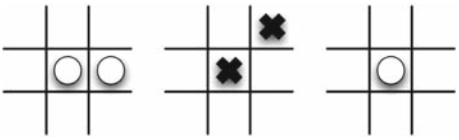

**Fig. 4** A visualization of $\mathbb{S}^*$ for Tic-Tac-Toe 5% using $s^{\downarrow}$, $\Phi_C$, and a threshold of 0.4

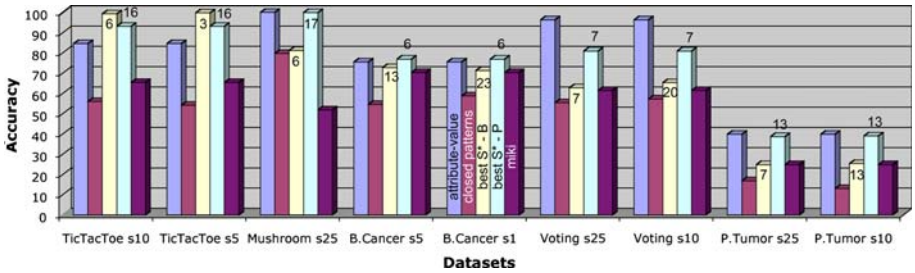


**Fig. 5** C4.5 cross-validated accuracies for attribute-value representation, closed-set binary representation, best $\mathbb{S}^*$, best $\mathbb{S}^*$using upper-bound pruning, and *miki* (2 patterns)

TicTacToe was indeed an exception. On the other data sets, it achieves the best accuracy of all the pattern sets, while at the same time still having relatively low cardinality.

The attribute-value representation still proves to be more expressive than the pattern representations for most cases, a problem that could probably be alleviated with the use of a more lenient support threshold. Furthermore, even though the reduced pattern sets always perform well, no single ordering or selection method does distinguish itself and selection thresholds vary between 0.03 and 0.4. This is particularly pronounced in the case of TicTacToe. Keep in mind however that we did not aim to maximize predictive accuracy and our illustrative instantiations are not meant to exhaust the entire issue.

In a sense, the selection technique using upper-bound pruning can be interpreted as using an adaptive ordering. Thus, while the other selection methods are constrained into a certain processing order, the upper-bound calculation allows to repeatedly select the *best* pattern in a situation, giving rise to the improved accuracy.

### 6.4 Comparison to pattern teams

Finally, we also used the algorithms by Knobbe et al. [8] 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 $\Phi_C$ with a rather strict threshold. Interestingly the sets compared exhibit very similar behavior w.r.t. 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 mikis are mined using a complete method maximizing mutual entropy, a measure that rewards balanced partitions, while we employ a heuristic method.

## 7 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.

### 7.1 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:

#### *7.1.1 Selecting subsets with ad hoc properties*

The earliest and in a sense simplest approach to reducing sets of frequent patterns lies in selecting subsets 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 [12]. 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 [1,3,5,15]. 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.

#### *7.1.2 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 [11], 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 that 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.

A technique that somewhat bridges the gap between the approach described in the preceding section and the one above was developed in [13]. While the bases mined in according to this work will be larger than borders, they carry approximate support information as well, therefore supplying more information to the user than a pure border would. At the same time, syntactic information can be recovered more reliable than from cluster representatives.

#### *7.1.3 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 [14], or removal of redundant information according to the joint entropy criterion [8]. The work of Siebes et al. [14] (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. [14] are larger than ours.

While stating their goal in slightly different words—namely, as summarizing categorical data in such a way as to achieve maximal compression at minimal information loss—the work described in [6] follows very similar lines of attack. In fact, the stated motivation is *very* similar to ours, just from a different point of view—that of data summarization instead of pattern reduction. This difference is also mirrored in the algorithmic solution they find, in which the loss w.r.t. transaction representation is made explicit as well as the gain in compression. Given the right definition of a selection measure, this is another technique that can be modeled by our approach.

The approach of Knobbe et al. [14] 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

### 7.2 Feature selection/construction for classification

#### *7.2.1 Feature selection*

Approaches to feature selection involve for instance relevancy constraints in subgroup discovery [10]. 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 the so-called wrapper approaches. In these usually features are selected, handed to an evaluation algorithm and then a decision on their inclusion is made. An example that seems very close to our work is [7] in that the wrapper uses unsupervised learning instead of an induction algorithm. A main difference to such approaches lies in the fact that usually all features for inclusion are considered, making the search space again rather large. Using a heuristic approach, we can handle far more patterns than such an approach could.

### 7.2.2 Feature construction

A somewhat similar approach to this work is embodied by the KFOIL[9] system, as well as by TREE$^2$[4]. 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.

## 8 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 two general heuristic algorithms 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, size ascending, or adaptive), 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 with 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 $\Phi_C$ being computationally most expensive but also showing very good reduction effects while the $\Phi_I$ and $\Phi_Q$ recover more of the original partitioning at the expense of less reduction in set cardinality.

The question which of the proposed measures is best-suited for the task of pattern set reduction, cannot be clearly answered. Both the indirect (via outside algorithms) and direct methods showed strengths and weaknesses, with the direct methods being better suited for explorative reduction since they can be evaluated faster. Using upper bounds to dynamically re-order the patterns while processing them gives rise to more balanced partitions that often are more useful for classification.

There is of course ample opportunity for future work. The flexibility of our approach allows for the definition of more exotic orderings—for instance a human expert could 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 mikis 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. Bonchi F, Lucchese C (2006) On condensed representations of constrained frequent patterns. Knowl Inf Syst 9:180–201
2. Borgelt C (2004) Recursion pruning for the apriori algorithm. In: Bayardo RJ Jr, Goethals B, Zaki MJ (eds) FIMI'04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Brighton, UK, November 1, 2004. CEUR Worshop Proceedings, CEUR-WS.org, vol 126
3. Boulicaut J-F, Jeudy B (2001) Mining free itemsets under constraints. In: Adiba ME, Collet C, Desai BC (eds) IDEAS, pp 322–329
4. Bringmann B, Zimmermann A (2005) Tree$^2$—decision trees for tree structured data. In: Jorge A, Torgo L, Brazdil P, Camacho R, Gama J (eds) PKDD. Springer, Berlin, pp 46–58
5. Calders T, Goethals B (2002) Mining all non-derivable frequent itemsets. In: Elomaa T, Mannila H, Toivonen H (eds) PKDD. Springer, Berlin, pp 74–85
6. Chandola V, Kumar V (2007) Summarization—compressing data into an informative representation. Knowl Inf Syst 12:355–378
7. Dy JG, Brodley CE (2004) Feature selection for unsupervised learning. J Mach Learn Res 5:845–889
8. Knobbe AJ, Ho EKY (2006) Pattern teams. In: Fürnkranz J, Scheffer T, Spiliopoulou M (eds) PKDD. Springer, Berlin, pp 577–584
9. Landwehr N, Passerini A, Raedt LD, Frasconi P (2006) kfoil: learning simple relational kernels. In: AAAI. AAAI Press, Menlo Park
10. Lavrac N, Gamberger D (2004) Relevancy in constraint-based subgroup discovery. In: Boulicaut J-F, Raedt LD, Mannila H (eds) Constraint-based mining and inductive databases. Springer, Berlin, pp 243–266
11. Lent B, Swami AN, Widom J (1997) Clustering association rules. In: Gray WA, Larson P-Å (eds) ICDE, pp 220–231
12. Mannila H, Toivonen H (1997) Levelwise search and borders of theories in knowledge discovery. Data Min Knowl Discov 1(3):241–258
13. Pei J, Dong G, Zou W, Han J (2004) Mining condensed frequent-pattern bases. Knowl Inf Syst 6:570–594
14. Siebes A, Vreeken J, van Leeuwen M (2006) Item sets that compress. In: Ghosh J, Lambert D, Skillicorn DB, Srivastava J (eds) SDM. SIAM, Philadelphia
15. Taouil R, Pasquier N, Bastide Y, Lakhal L (2000) Mining bases for association rules using closed sets. In: ICDE, p 307

## Author Biographies

**Björn Bringmann** received his MSc in Computer Science from the Albert-Ludwigs-University Freiburg in Germany in 2004. He is currently a PhD student in Computer Science Department at the Katolieke Universiteit Leuven. His research interests include AI, specifically data mining from structured data.

**Albrecht Zimmermann** received an MSc in Computer Science from the Albert-Ludiwgs University, Freiburg, Germany. He is currently a doctoral student of Professor Luc De Raedt's at the Katholieke Universiteit Leuven, Belgium. His research interests include data mining with a main focus of mining sets of local patterns and correlated pattern mining.