Constructing Classication Features Using Minimal Predictive Patterns

Iyad Batal
Department of Computer Science
University of Pittsburgh
iyad@cs.pitt.edu

Milos Hauskrecht
Department of Computer Science
University of Pittsburgh
milos@cs.pitt.edu

ABSTRACT

Choosing good features to represent objects can be crucial to the success of supervised machine learning methods. Recently, there has been a great interest in applying data mining techniques to construct new classification features. The rationale behind this approach is that patterns (feature-value combinations) could capture more underlying semantics than single features. Hence the inclusion of some patterns can improve the classification performance. Currently, most methods adopt a two-phases approach by generating all frequent patterns in the first phase and selecting the discriminative patterns in the second phase. However, this approach has limited success because it is usually very difficult to correctly identify important predictive patterns in a large set of highly correlated frequent patterns

In this paper, we introduce the minimal predictive patterns framework to directly mine a compact set of highly predictive patterns. The idea is to integrate pattern mining and feature selection in order to filter out non-informative and redundant patterns while being generated. We propose some pruning techniques to speed up the mining process. Our extensive experimental evaluation on many datasets demonstrates the advantage of our method by outperforming many well known classifiers.

Categories and Subject Descriptors

I.2.6 [**LEARNING**]: General

General Terms

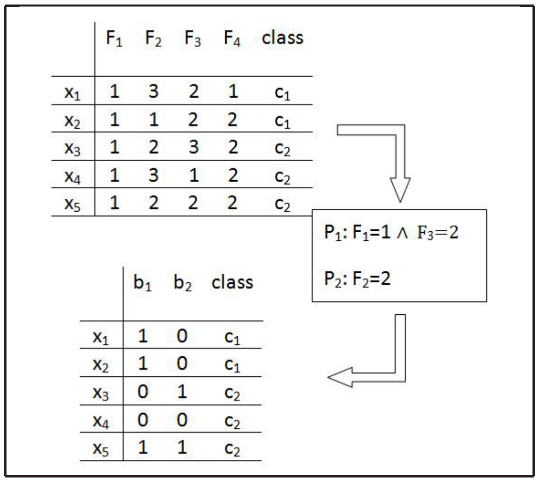
Algorithms, Experimentation, Performance

1. INTRODUCTION

Classification and pattern mining are two very important problems in data mining and machine learning. Recently, there has been an increasing interest in studying the combination of these problems. Earlier studies mainly focused on *associative classification*, where a rule based classifier is built from high-support and high-confidence association rules [20, 18, 10, 27, 26]. Recently, the focus was more on using discriminative frequent patterns to define new features in order to improve the quality of the learning algorithm [8, 12, 5].

Let us explain the rationale of using patterns in classification in terms of a linear classifier, like the famous SVM classifier [25]. Linear SVM tries to learn the *optimal* (maximum margin) hyperplane that separates the classes. While this classifier can achieve very high accuracies on some data, it may completely fail on other data due to the fact that the classes cannot be linearly separated in the original feature space. One approach to overcome this problem is to use the *kernel trick* [11], which implicitly maps the data into a higher dimensional space. Pattern based classification is another more data-driven approach to solve the problem by trying to construct new classification features using pattern mining techniques.

Consider the example in Figure 1. The two classes c_1 and c_2 cannot be separated by any linear classifier in the original 4 dimensional space. Now assume an oracle told us that this data contain two patterns: P_1: F_1=1 $\wedge$ F_3=2 and P_2: F_2=2 (a pattern is a conjunction of feature-value pairs). Using this information, we can map the data into a higher dimensional space by defining two additional binary features b_1 and b_2, where b_i simply indicates the presence or absence of pattern P_i in each data instance. After performing this dimensionality expansion, it becomes very easy for a linear model to separate the classes.



	F_1	F_2	F_3	F_4	class
x_1	1	3	2	1	c_1
x_2	1	1	2	2	c_1
x_3	1	2	3	2	c_2
x_4	1	3	1	2	c_2
x_5	1	2	2	2	c_2

P_1: F_1=1 $\wedge$ F_3=2

P_2: F_2=2

	b_1	b_2	class
x_1	1	0	c_1
x_2	1	0	c_1
x_3	0	1	c_2
x_4	0	0	c_2
x_5	1	1	c_2

Figure 1: An example illustrating the use of patterns in classification

Frequent pattern mining is a very popular data mining technique to extract patterns from the data. Since their introduction in [1], frequent pattern and association rule mining have received a great deal of attention and have been successfully applied to various domains [30, 5, 17]. The key strength of frequent pattern mining is that it searches the space of patterns thoroughly by examining all patterns that occur frequently in the data. However, its main disadvantage is that the number of frequent patterns is usually very large. Moreover, many of these patterns are redundant because they are only small variations of each other. Hence, it is not practical to use all frequent patterns in classification. The purpose of this study is to present a new way to intelligently select a small subset of frequent patterns that are useful for the classification task.

Most of the methods that use patterns in classification adopt a *two-phases* approach [8, 12, 5, 18, 20] by first mining all frequent patterns and then selecting the top k discriminative patterns to be used by the classifier. However, there are many inherent problems with this approach. First, it is not clear what is the optimal number of patterns (k) that should be used by the classifier. Second, the number of frequent patterns can be very large, making the application of an effective feature selection method computationally expensive. Third, the algorithm may falsely select many *spurious* patterns, where a spurious pattern is a pattern that seems interesting by itself, but is completely redundant with respect to other patterns.

Having discussed these problems, it is important to develop a method to *directly* mine a small set of patterns that are highly *predictive* and at the same time contain *low redundancy*. To achieve this goal, we first introduce the concept of *minimal predictive patterns* (MPP). The idea is to exploit the nested structure of the patterns in order to assure that every pattern in the result offers a significant predictive advantage over all of its generalizations (simplifications). After than, we propose an effective algorithm to directly mine these patterns. In contrast to the two-phases approach, our algorithm integrates pattern mining and feature selection and evaluates each pattern as soon as it is generated. Finally, we present a very efficient pruning technique to approximately mine the MPP set. Our experimental evaluations clearly demonstrate the benefits of our method to efficiently learn very accurate classifiers.

2. PATTERN BASED CLASSIFICATION

2.1 Definitions

Assume a dataset with only categorical features (attributes): all numeric features should be first discretized. Each (*feature*,*value*) pair is mapped to a distinct *item* in $\Sigma = \{I_1, ..., I_l\}$. A *pattern* is a conjunction of items: $P = I_{q_1} \wedge ... \wedge I_{q_k}$ where $I_{q_j} \in \Sigma$. If a pattern contains k items, we call it a *k-pattern* (an item is a *1-pattern*). Assume an item $I = (fea, val)$, where fea is a feature and val is a value. Given a data instance $\boldsymbol{x}$, we say that $I \in \boldsymbol{x}$ if $fea(\boldsymbol{x}) = val$ and that $P \in \boldsymbol{x}$ if $\forall I_j \in P : I_j \in \boldsymbol{x}$.

Given a dataset $D = \{\boldsymbol{x_i}\}_{i=1}^{n}$, the instances that contain pattern P define a group $D_P = \{\boldsymbol{x_j} | P \in \boldsymbol{x_j}\}$. If P' is a subpattern of P ($P' \subset P$), then $D_{P'}$ is a supergroup of D_P ($D_{P'} \supseteq D_P$). Note that the empty pattern Φ defines the entire population. The support of P is defined as: $sup(P) = |D_P|/|D|$.

In our framework, we are interested in mining patterns that are predictive of the class variable. So for pattern P, we can define a rule R: $P \Rightarrow c$ with respect to class label c. The confidence of R is the posterior probability of c in group D_P. Note that confidence of $\Phi \Rightarrow c$ is the prior probability of c. We say that rule R': $P' \Rightarrow c'$ is a subrule of rule R: $P \Rightarrow c$ if $c' = c$ and $P' \subset P$.

Given a dataset $D = \{\boldsymbol{x_i}, y_i\}_{i=1}^{n}$ in d dimensional feature space and a set of patterns $\Omega = \{P_1, ..., P_m\}$, D is mapped into a higher dimensional space with $d + m$ dimensions as follows:
$\boldsymbol{x_i} \rightarrow \boldsymbol{x_i'} = \boldsymbol{x_i} \cup \{b_{i,1}, ..., b_{i,m}\}$ where $b_{i,j} = 1$ if $P_j \in \boldsymbol{x_i}$ and $b_{i,j} = 0$ if $P_j \notin \boldsymbol{x_i}$
The classification model is then learned in the new *expanded* feature space $D' = \{\boldsymbol{x_i'}, y_i\}_{i=1}^{n}$.

2.2 Why using Frequent Pattern Mining?

We say that pattern P is a *frequent pattern* if $sup(P) \geq min\_sup$, where *min_sup* is a defined threshold. Frequent patterns obey the *monotonicity* property: "if pattern P is not frequent, then all its superpatterns ($P'' \supset P$) are not frequent". This property implies that if P is frequent, then all its subpatterns ($P' \subset P$) are also frequent.

There are two important reasons for using frequent patterns in classification:

- **Rare patterns are not useful predictors:** [8] studied the relationship between the discriminative power of a pattern and its support. They demonstrated, both analytically and experimentally, that low support patterns have limited discriminative power and that including them in the classifier can harm the classification accuracy due to overfitting.

- **Mining frequent patterns is more tractable:** Obviously generating all possible patterns from data is computationally intractable due the combinatorial explosion in the number of patterns. However, imposing a support threshold makes the problem more tractable and we can use an off-the-shelf frequent mining algorithm [2, 16] to find those patterns.

2.3 Selecting the Classification Patterns

Even though some frequent patterns can be important predictors, using all frequent patterns in the classifier is not a good option for the following reasons:

- **Large number of patterns**: It is not uncommon to generate thousands of frequent patterns even from a small dataset. To understand this problem, remember that all subpatterns of a frequent pattern are also frequent. So if P is a frequent *l-pattern*, then all its $2^l - 1$ subpatterns are included in the result!

- **Many non-predictive patterns**: Since frequent patterns are generated solely based on their support, not based on their discriminative power, a large number of frequent patterns are not useful for classification.

- **Many spurious patterns**: A spurious pattern is a pattern that may seem important (predictive) by itself, but is completely redundant given some other patterns. A large number of these spurious patterns can be formed by adding irrelevant items to other predictive patterns. We will explain this problem in details using the following example.

Example 1: Consider a Bayesian belief network example in Figure 2, where we have a causal relation between feature F_1 and the class variable C. All other features are independent of C: $F_i \perp\!\!\!\perp C$: $i \in \{2..n\}$. Assume that pattern (item) F_1=1 is highly predictive of class c_1, so that the posterior Pr(C=c_1 | F_1=1) is significantly larger than the prior Pr(C=c_1). Let us denote this pattern as P_M. Clearly, P_M is the only predictive pattern in this data.

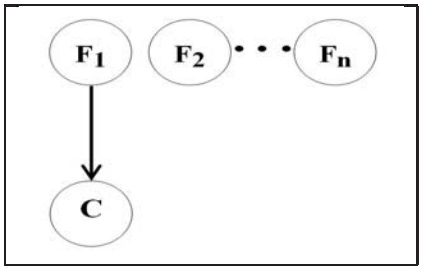


Figure 2: Illustrating the problem of spurious patterns in frequent pattern mining

Now consider a spurious pattern P_S: F_1=1 $\wedge$ F_{q_1}=v_{q_1} ... $\wedge$ F_{q_k}=v_{q_k}, where $F_{q_i} \in \{F_2, ..., F_n\}$ and v_{q_i} is *any* possible value of variable F_{q_i}. The network structure implies that Pr(C=c_1 | P_S) $\approx$ Pr(C=c_1 | P_M). The problem is that if we evaluate the patterns individually (without considering the nested structure of the patterns), we may falsely think that P_S is a predictive pattern because Pr(C=c_1 | P_S) is significantly larger than Pr(C=c_1). However, P_S is totally redundant given P_M. Clearly, the number of these spurious patterns can become huge. For example, if we assume that all variables are binary, there are $3^{n-1}-1$ spurious patterns in this network structure.

The two-phases approach: This approach [8, 12, 5, 18, 20] performs feature selection on the result of a frequent pattern mining algorithm. However, the above analysis shows that many spurious patterns can be considered predictive using most interestingness measures [15]. To circumvent this problem, [8] suggested using a redundancy score (e.g. the Jaccard score [15]) and selecting the features in an incremental way: a feature is added to the set of features if it is both *predictive* and has *low redundancy* to the features already selected. We can see that performing effective feature selection in the two-phases approach is:

- **Difficult**: It is not clear how to determine the optimal number of patterns that should be used by the classifier. In other words, it is very hard to correctly identify the real patterns from a large pool of spurious patterns.
- **Inefficient**: Applying feature selection using methods like the incremental feature selecting can be very slow because the number of frequent patterns is usually large.

Objective: Having discussed these problems, it is important to develop a principled way to *directly* mine the classification patterns. Basically, the method should return a *small* set of patterns that are highly *predictive* and at the same time contain *low redundancy*. In order to satisfy these requirements, we define the *minimal predictive patterns* concept.

3. MINIMAL PREDICTIVE PATTERNS

Definition We say that pattern P is a minimal predictive pattern (MPP) if there is a class label c such that P predicts c significantly better than all of its subpatterns (including the empty pattern ϕ).

We call these patterns *minimal* because removing any non-empty combination of items from the pattern would cause a significant drop in its predictive power. We can see that the MPP definition prefers simple patterns over more complex patterns (the Occam's Razor principal) because pattern P is not an MPP if its effect on the class distribution "can be explained" by a simpler pattern ($P' \subset P$) that covers a larger population.

3.1 The MPP Significance Test

Assume our task is to check whether pattern P is an MPP with respect to class c. Let us define rule R: $P \Rightarrow c$. Assume that group D_P contains N instances, out of which N_c instances belong to class c. Let P_c be the highest confidence achieved by any subrule of R: $P_c = \max_{P' \subset P} \Pr(c|D_{P'})$. The null hypothesis presumes that N_c is generated from N according to the binomial distribution with probability P_c. The alternative hypothesis presumes that the true underlying probability that generated N_c is significantly higher than P_c. Hence, we perform a *one sided* significance test (we are interested only in increases in the proportion of c) using the binomial distribution and calculate its p-value as follows:

$$p = Pr_{binomial}(x \geq N_c | N, P_c)$$

If this p-value is smaller than a significance level α (commonly α=0.05), we conclude that P significantly improves the predictability of c over all its subpatterns, hence P is an MPP.

3.2 Geometric View

Geometrically, we can see group D_P of pattern P as defining a distinct *region* in the feature space. Hence, when P is a composite pattern (P contains more than one item), its region is the intersection of the regions of all of its subpatterns: $D_P = \bigcap_{P'_i \subset P} D_{P'_i}$. Our objective is to identify those regions in the space that are important for classification. Since this can be seen similar to the objective of a decision tree, so let us start by illustrating the difference between these two approaches.

A decision tree partitions the space recursively by choosing at each time the best item to split the data. Therefore, the space is partitioned into *non-overlapping* regions. However, because the partitioning is performed greedily, it is very likely for a decision tree to miss important regions in the space. In comparison, frequent pattern mining performs a more complete search by exploring all possible frequent combinations of items. Therefore, it produces a large number of *highly overlapping* regions.

Now pattern P is an MPP if P predicts a class c significantly better than all of its subpatterns. This means that we cannot express the class distribution in region D_P by *any* convex combination of the distributions in the regions that contain D_P:

$$[C|D_P] \neq \sum_{D_{P'_i} \supset D_P} \alpha_i \times [C|D_{P'_i}] \text{ where } \sum \alpha_i = 1 \text{ and } \alpha_i \geq 0$$

Therefore, P describes a distinct discriminative region D_P in the space where the class distribution in D_P cannot be explained by any of its containing regions.

3.3 Removing Spurious Patterns

Let us go back to example 1 (Figure 2) to explain how the MPP framework tackles the problem of spurious patterns in frequent pattern mining. Pattern P_S: F_1=1 $\wedge$ F_{q_1}=v_{q_1} ... $\wedge$ F_{q_k}=v_{q_k} is not an MPP because P_S will not be significantly more predictive than its subpattern P_M: F_1=1.

More generally, for pattern P: F_{q_1}=v_{q_1} ... $\wedge$ F_{q_k}=v_{q_k} to be an MPP, there should exist a path from each variable F_{q_i} to the class C that is not blocked (d-separated) by the other variables in the pattern: F_{q_i} not $\perp\!\!\!\perp C$ | { F_{q_1}, ... , $F_{q_{i-1}}$, $F_{q_{i+1}}$, ... , F_{q_k} }. Because this is not the case in example 1, we expect these spurious patterns to be filtered out.

4. THE MINING ALGORITHM

In this section we explain our algorithm for mining the MPP set. The algorithm is outlined in Figure 3. Briefly, the algorithm explores the space by performing an Apriori-like level wise search. At each level (l), we first remove the candidate *l-patterns* that do not pass the minimum support threshold (line 6). Then we extract all MPPs from those frequent *l-patterns* (line 7). Finally, we generate the candidates for the next level (line 8). This process is repeated until no more candidates can be generated.

```
Algorithm 1: Mine all MPPs
Input: Dataset: $D$, minimum support: $min\_sup$
Output: minimal predictive patterns: $MPP$
//global variables
01: $MPP$=$\Phi$, $tbl\_max\_conf$=$hashtable()$
//the prior distribution of the class
02: $tbl\_max\_conf[h(\Phi)]$=$calculate\_class\_distribution(\Phi, D)$
03: $Cand$=$generate\_1\_patterns()$
04: $l = 1$
05: while ($Cand \neq \Phi$)
          //remove candidates that are not frequent
06:       $FP[l]$=$prune\_infrequent(Cand, D, min\_sup)$
          //find all MPPs at level $l$
07:       extract_MPP($FP[l]$, $D$)
          //generate candidate $(l+1)\_patterns$
08:       $Cand$=$generate\_candidates(FP[l])$
09:       $l = l + 1$
10: return $MPP$
```

Figure 3: The algorithm for mining the MPPs from a dataset

4.1 Extracting MPPs

The process of testing if frequent pattern P is an MPP is not trivial because the definition requires us to check the pattern against all its subpatterns. This requires checking all 2^l-1 subpatterns if P has length *l*. Our algorithm avoids this inefficiency by caching the statistics needed to perform the MPP test within the *(l-1)-subpatterns* from the previous level. This part of the algorithm is outlined in Figure 4.

To explain the method, it is useful to envision the progress of the algorithm as gradually building a lattice structure level by level, starting from the empty pattern Φ. An example lattice is shown in Figure 5. Every frequent *l-pattern P* is a node in the lattice with l children: one child for each of its *(l-1)-subpatterns*.

The key idea of our algorithm is to store in each node P the maximum confidence score for every class that can be obtained by the patterns in the sublattice with top P (including P itself): $max\_conf_P[c]$=max(Pr($c \mid D_{P'}$)): $\forall\ P' \subseteq P$. These *max_conf* values are computed from the bottom up as algorithm progresses.

Initially, max_conf_Φ for the empty pattern is set to be the prior distribution of the class variable. In order to compute max_conf_P for pattern P, we first compute $conf_P$ (line 2), the distribution of the class variable in group D_P: $conf_P[c]$=Pr($c \mid D_P$). Then we use the *max_conf* values of P's direct children to compute $max\_conf\_children_P$ (line 3) so that: $max\_conf\_children_P[c]$=max(Pr($c \mid D_{P'}$)): $\forall\ P' \subset P$. Finally, we compute max_conf_P by taking the element-wise maximum of two arrays: $conf_P$ and $max\_conf\_children_P$ (line 4).

Now we want to check if P is an MPP. So for each class label c, we perform the MPP significance test to check if P predicts c

```
Algorithm 2: extract_MPP ($FP[l]$, $D$)
//add pattern $P \in FP[l]$ to $MPP$ if $P$ is significantly more predictive
than all its subpatterns
1: for each $P \in FP[l]$
        //the distribution of the class in group $D_P$
2:      $conf$=$calculate\_class\_distribution(P, D)$
3:      $max\_conf\_children$=max $\{tbl\_max\_conf[h(S_{l-1})]\} : S_{l-1} \subset P$
4:      $max\_conf$=max$\{conf, max\_conf\_children\}$
5:      $tbl\_max\_conf[h(P)]$=$max\_conf$
6:      if ( $is\_MPP(P, max\_conf\_children, D)$ )
7:              $MPP = MPP \cup P$
8:      $lossless\_pruning(P, max\_conf, D, FP[l])$

Function is_MPP($P$, $max\_conf\_children$, $D$)
//return true if $P$ predicts any class significantly better than
all its subpatterns
$N$=$count(P, D)$
for each class label $c$
    $N_c$=$count(P \cup c, D)$
    $p\_value$=$Pr_{binomial}(x \geq N_c \mid N, max\_conf\_children[c])$
    if($p\_value < \alpha$)
        return true
return false

Function lossless_pruning($P$, $max\_conf$, $D$, $FP[l]$)
//prune $P$ if it cannot produce any $MPP$ for any class
for each class label $c$
    $N_c$=$count(P \cup c, D)$
    $p\_value$=$Pr_{binomial}(x \geq N_c \mid N_c, max\_conf[c])$
    //exit if $P$ can potentially produce an $MPP$ for class $c$
    if($p\_value < \alpha$)
        return ;
$remove(P, FP[l])$
```

Figure 4: The algorithm for extracting the MPPs from the frequent patterns at level l

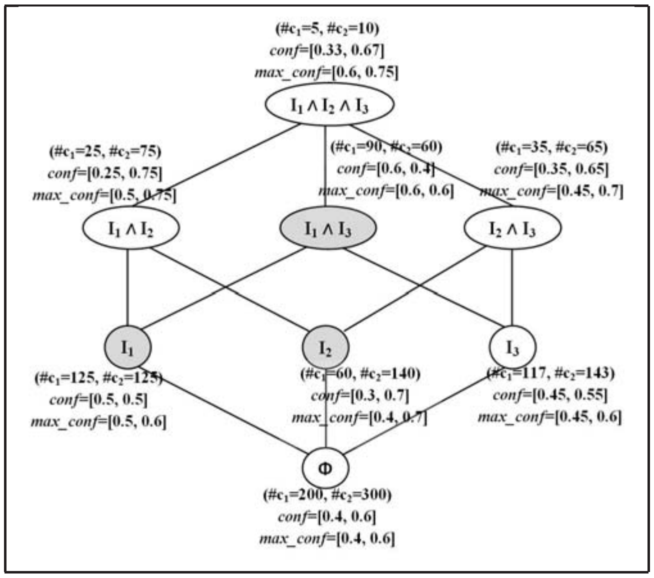


Figure 5: An illustrative example showing the frequent pattern lattice associated with $I_1 \wedge I_2 \wedge I_3$. The MPPs are shaded

significantly better than $max\_conf\_children_P[c]$. If the test is positive, we add P to the MPP set (line 7).

Please note that in our implementation, we do not explicitly build the frequent pattern lattice. Instead, we use a hash table *tbl_max_conf* to provide direct access to the *max_conf* values. So that *tbl_max_conf*$[h(P)]$ = max_conf_P, where h is a hash function. Also note that none of the functions (*calculate_class_distribution*, *is_MPP* and *lossless_pruning*) requires another scan on the data because we can collect the class specific counts during the same scan that counts the pattern.

Figure 5 illustrates the algorithm using a small lattice on a dataset that contains 200 instances from class c_1 and 300 instances from class c_2. For each pattern P (node), we show the number of instances from each class in group D_P, the distribution of the classes in D_P (*conf*) and the maximum confidence from P's sublattice (*max_conf*). Let us look for example at pattern $I_1 \wedge I_2$. This pattern is predictive of class c_2: *conf*$(I_1 \wedge I_2 \Rightarrow c_2) = 0.75$. However, it is not an MPP because it does not significantly improve the predictability of c_2 over subrule $I_2 \Rightarrow c_2$: $Pr_{binomial}(x \geq 75 \mid 100, 0.7)$ = 0.16 (not significant at significance level $\alpha = 0.05$). The MPPs from this example are: I_1, I_2 and $I_1 \wedge I_3$.

4.2 Pruning the search space

We say that pattern P is *pruned* if we do not examine any of P's superpatterns (D_P's subgroups). This can be seen as excluding the entire sublattice with bottom P from the pattern lattice.

The Apriori algorithm relies only on the support of the patterns and prunes the infrequent patterns according to the monotonicity property. However, frequent pattern mining becomes computationally very expensive when:

- The data contain many items.
- The *min_sup* level is low.
- The data features are highly correlated.

All of these reasons cause the frequent pattern lattice to become extremely huge. One simple way to speed up the algorithm is to set *min_sup* very high. However, this solution is not very effective because the algorithm may miss many important patterns. In fact, [8] argued that the discriminative power of very high support patterns is bounded by a small value[1]. Therefore, we need a method that can utilize the discriminative ability of the patterns in order to further restrict the search space. We present two pruning techniques: the first one is *lossless* (does not miss any MPP), while the second is *lossy*.

4.2.1 Lossless Pruning

The MPP significance test can help us to prune the search space. This pruning is implemented by the *lossless_pruning* function in Figure 4. The idea is to prune pattern P if we guarantee that P cannot produce any MPP. However, since our algorithm is applied in a level-wise fashion, we do not know what subgroups P will generate further in the lattice. To overcome this, we define the *optimal subgroup* $D_{P^*_{c_i}}$ in group D_P with respect to class c_i to be the subgroup that contains all the instances from c_i and none of the instances from any other classes. Clearly, P cannot generate any subgroup better than $D_{P^*_{c_i}}$ for predicting class c_i. Now, we *safely* prune P if for every class c_i, $P^*_{c_i}$ does not significantly improve the predictability of c_i with respect to the current best prediction (*max_conf*$[c_i]$). Please note that this pruning technique does not miss any MPP because the *lossless_pruning* test is anti-monotonic.

As an example, consider pattern $P = I_1 \wedge I_2 \wedge I_3$ in Figure 5. This pattern contains 15 examples, 5 from class c_1 and 10 from class c_2. Both $P^*_{c_1}$ and $P^*_{c_2}$ are not significant at α=0.05 with respect to the current best predictions 0.6 and 0.75 (for c_1 and c_2 respectively). Therefore, we can safely prune P.

4.2.2 Lossy Pruning

This technique performs *lossy* pruning, which means that it speeds up the mining, but at the risk of missing some MPPs. We refer to the set of patterns mined using the lossy pruning method as A-MPP (Approximated MPP). The idea is to prune pattern P if P does not show any sign of being more predictive than its subpatterns. To do this, we simply perform the MPP significance test, but at a higher significance level α_2 than the significance level used in the original MPP test: $\alpha_2 \in [\alpha, 1]$. If P does not satisfy the MPP test with respect to α_2, we prune P.

We can see that α_2 is a parameter that controls the tradeoff between efficiency and completeness. So if we set α_2 = 1, then we do not do any lossy pruning. On the end of the spectrum, if we set α_2 = α, then we prune every non-MPP pattern, i.e., we require every subpattern of every MPP to be MPP as well.

4.3 Illustrating Examples

4.3.1 Highly Correlated Items

In this example, we discuss the effect of item *correlation* on the efficiency of pattern mining. High correlation between items causes many long patterns to become frequent, and therefore severely slows down the mining algorithm. For instance, assume items $I_1, ..., I_n$ occur often together in the data. Let $P = I_1 \wedge ... \wedge I_n$. if P is frequent, then the algorithm has to generate all patterns in the lattice with top P. However, because of the high correlation, all of these patterns are going to cover very similar groups (regions) in the data. Therefore, the class distribution will be very similar: $[C|D_{P'}] \approx [C|D_{P''}] : \forall\, P', P'' \subseteq P$. Applying the lossy pruning can save us a lot of computation because P's lattice will not be explored beyond the second level. The reason that any 2-pattern $P' = I_i \wedge I_j \subset P$ is likely to fail the MPP test with $\alpha_2 < 1$ because P' will not show any potential of being different from its parents: $[C|D_{I_i \wedge I_j}] \approx [C|D_{I_i}] \approx [C|D_{I_j}]$.

4.3.2 The XOR Example

Consider the XOR example, where we have two binary features F_1 and F_2 and a class variable defined as: $C = F_1$ XOR F_2. Learning any linear classifier on this data is hopeless because no hyperplane can separate the two classes. Besides, trying to partition the space using a decision tree is also unlikely to succeed. Clearly, the data contains the 4 MPPs: $\{F_1$=0 $\wedge$ F_2=0, F_1=0 $\wedge$ F_2=1, F_1=1 $\wedge$ F_2=0, F_1=1 $\wedge$ F_2=1$\}$. By adding these MPPs as features, it becomes extremely easy for a linear classifier to perfectly classify the data[2].

However, the bad news is that the lossy pruning method will miss all of these MPPs. The reason is that the distribution of the class variable in the group of any item is the same as the prior distribution: $[C|D_{I_j}]$=$[C]$: $I_j \in \{F_1$=0, F_1=1, F_2=0, F_2=1$\}$. Therefore,

[1]This is analogous to the stop words in document retrieval or text categorization.

[2]A kernel-based classifier is also likely to succeed on this toy example.

Dataset	*#fea*	*#items*	*#rec*	*#cls*	*SVM_linear*	*SVM_RBF*	*DT*	*DS*	*DT_fea*	*A-MPP*
lymphography	18	57	142	2	83.76	82.38	76.67	63.43	81.69	**88.51**
tic-tac-toe	9	27	958	2	65.34	86.73	86.85	69.51	86.85	**98.33**
breast cancer	9	41	286	2	71.65	73.39	68.15	64.43	68.15	**74.85**
nursery	8	27	12960	5	91.03	**99.91**	98.9	79.63	97.92	96.88
hepatitis	19	39	155	2	76.04	79.37	73.54	76.83	74.17	**84.66**
glass	10	22	214	6	99.57	98.16	97.64	94.41	99.11	**100**
mushrooms	22	116	8124	2	99.63	**100**	**100**	98.56	**100**	**100**
WDBC	30	94	569	2	95.08	92.81	92.45	**97.89**	93.68	96.32
wine	13	39	178	3	**95.51**	68.6	91.13	95.61	94.40	94.44
pen digits	16	147	10992	10	97.94	98.24	95.74	95.02	98.10	**98.51**
heart disease	13	33	303	2	81.53	64.63	78.23	73.44	78.23	**82.51**
Pima diabetes	8	19	768	2	76.44	73.31	70.84	73.43	70.97	**76.56**
mammographic	5	13	961	2	82.31	80.34	80.44	81.7	80.55	**83.24**
acute	6	12	120	2	**100**	**100**	**100**	89.17	**100**	**100**
car	6	21	1728	4	87.04	**98.96**	95.2	68.58	95.25	95.48
zoo	16	36	101	7	95.36	92.36	90.05	75.06	95.25	**96.27**
E.coli	7	19	336	8	81.34	**87.81**	82.49	25.6	83.36	83.52
image-seg	19	54	210	7	**90**	76.67	83.81	55.71	89.52	89.52
credit	15	69	690	2	85.5	72.44	85.79	82.88	85.79	**87.68**
parkinson	22	51	195	2	86.93	82.99	86.04	87.21	86.04	**93.78**
average					87.1	85.46	86.69	77.41	87.95	**91.05**

Table 1: Comparing the classification accuracy of A-MPP against several well known classifiers

none of these items will show any predictability signal and the algorithm will falsely prune all of them for any $\alpha_2 < 1$.

However, in real-world data, it is not very common for the class to follow an XOR-like distribution and our experiments will demonstrate that the lossy pruning can achieve orders of magnitude speed up while recovering most MPPs.

5. EXPERIMENTAL EVALUATION

In this section we present our experimental evaluation on 20 different UCI classification datasets [3]. Columns 1 to 4 in Table 1 show the main characteristics of the datasets (number of features, number of items, number of records and number of classes).

5.1 Compared Methods

The experiments compare the performance of the following methods:

- *SVM_linear:* The linear support vector machine (SVM) [25] classifier.
- *SVM_RBF*: SVM with the Radial Basis Function (RBF) kernel [11]. We optimize the kernel width (the gamma parameter) using internal cross-validation.
- *DT*: The CART (Classification And Regression Tree) [6] decision tree algorithm.
- *DS*: Decision Stumps (single-level decision trees) boosted using the Adaboost algorithm [14]. We use 100 iterations of boosting to build the classifier.
- *DT_fea*: This method uses patterns from a decision tree (corresponding to the leaves of the tree) to define new classification features. The purpose of including this method is to show that MPPs and A-MPPs are better classification features than patterns (regions) found heuristically by a decision tree.
- *two-phases*: This method generates the frequent patterns in the first phase and selects the discriminative ones in the second phase. As recommended by [8], we mine the *closed* frequent patterns [4] (lossless compression) instead of all frequent patterns in the first phase in order to reduce redundancy. We then select the top 50 closed patterns according to information gain (IG) to be used in classification.
- *MBST*: The recently proposed Model Based Search Tree (MBST) algorithm [13], which uses frequent pattern mining to induce a decision tree. The basic idea is to partition the data in a top down manner and construct a tree as follows: At each node of the tree, invoke a frequent pattern mining algorithm, select the most discriminative pattern (according to IG), and then divide the data into two subsets, one containing this pattern and the other not. After that, repeat the process recursively on the two subsets.
- *MPP*: This method uses the *complete* MPP set in classification. We set the significance level for the MPP statistical test to the conventional $\alpha = 0.05$ level. The mining algorithm applies the lossless pruning technique described in section 4.2.1 in addition to the support pruning.
- *A-MPP*: This method applies the lossy pruning technique described in section 4.2.2 and uses the *approximated* MPP set in classification. We set the pruning parameter $\alpha_2 = 0.25$.

We used the SVM implementation provided by the LIBSVM [7] package. For methods that use patterns in classification (*two-phases*, *DT_fea*, *MPP*, *A-MPP*), we learn a linear SVM classifier in the space of the original features plus the induced patterns.

For *two-phases*, *MPP* and *A-MPP*, we set the minimum support parameter (*min_sup*) to 10% the number of instances in the dataset. For *MBST*, we found that the performance of the algorithm (execution time and classification accuracy) is very sensitive to the invocation *min_sup* parameter. Unfortunately, the authors did not provide any guidance on how to set this parameter. We tried many values

Dataset	*two-phases*		*MBST*		*MPP*		*A-MPP*		Number of patterns		
	time (s)	acc	time (s)	acc	time (s)	acc	time (s)	acc	closed	MPP	A-MPP
lymphography	23,473	84.48	766	88.10	6,293	**89.33**	4	88.51	5,840	33	31
tic-tac-toe	17.1	91.24	7.6	74.42	11.7	**98.33**	8.8	**98.33**	191	44	44
breast cancer	16.7	72.02	9.3	**76.72**	11.9	74.85	0.9	74.85	379	10	10
nursery	240	94.82	71	77.44	162	**94.92**	116	94.88	121	46	41
hepatitis	>54K	NA	12,675	84.42	>54K	NA	5	**84.66**	NA	NA	27
glass	310	**100**	143	69.85	246	**100**	3	**100**	1,141	26	22
mushrooms	>54K	NA	>54K	NA	>54K	NA	398	**100**	NA	NA	439
WDBC	>54K	NA	>54K	NA	>54K	NA	56	**96.32**	NA	NA	266
wine	50.4	**96.66**	17.7	91.54	18.3	94.44	9.6	94.44	944	116	116
pen digits	350	98.28	631	63.94	348	**98.51**	349	**98.51**	92	92	92
heart disease	787	81.84	70	79.14	395	**82.51**	9	**82.51**	5,075	79	79
Pima diabetes	35.6	75.52	17.5	74.89	22.4	**76.56**	6.2	**76.56**	448	36	36
mammographic	7.5	82.21	7.8	**85.83**	4.4	83.24	1.9	83.24	95	16	16
acute	5.2	**100**	2.0	**100**	1.9	**100**	0.8	**100**	44	26	26
car	10.4	94.27	7.6	81.35	7.6	**95.48**	6.8	**95.48**	48	23	23
zoo	>54K	NA	5,797	93.94	9,206	**97.18**	9	96.27	NA	126	93
E.coli	26.5	80.78	15.7	79.28	18.2	**84.37**	4.9	83.52	150	38	35
image-seg	>54K	NA	>54K	NA	>54K	NA	47	**89.52**	NA	NA	299
credit	2,154	84.94	164	84.06	1,738	**87.72**	9	87.68	3,271	56	54
parkinson	>54K	NA	>54K	NA	>54K	NA	11	**93.78**	NA	NA	84

Table 2: Comparing the performance of several methods that use frequent pattern mining in classification

and found that setting the invocation *min_sup* to 30% is a reasonable choice for our datasets. Please note that setting *min_sup* = 10% for *MBST* is impractical because the algorithm becomes extremely inefficient even on simple datasets.

All classification results are reported using a 10 folds cross-validation scheme, where we use the same train/test splits for all of the competing methods.

5.2 Comparing against Classical Classifiers

Table 1 compares the classification accuracy of *A-MPP* against several famous classifiers. We can see that *A-MPP* is the best performing method on most datasets and achieves on average the highest classification accuracy. It is interesting to notice that the unconventional method of using decision tree patterns with SVM is able to produce good classifiers on several datasets. However, due to its restricted heuristic-based search, *DT_fea* is often outperformed by *A-MPP*. Also note that the performance of *SVM_RBF* is not consistent. It is the best method on some datasets (nursery, car and E.coli) and the worst method on other datasets (wine, heart-disease, credit and parkinson). This suggests that pattern based SVM can be considered an attractive alternative to kernel based SVM.

Let us discuss the behavior of the different classifiers on the tic-tac-toe dataset. We choose this data because the high interaction between the features greatly illustrates the usefulness of patterns in classification. This dataset encodes the different board configurations at the end of the tic-tac-toe game. Each data instance (board configuration) has 9 features, one for each of the 9 squares. Each feature takes one of the 3 values: 1=player x has taken, 2=player o has taken and 3=blank. The goal is to learn the target concept "win for player x". Although this task may seem easy for a human, it is not straightforward for a machine learning method to automatically learn the concept from data.

Table 1 shows that *SVM_linear* is the worst performing method on tic-tac-toe because the classes (win vs loose) cannot be linearly separated in the original 9 dimensional space. *DS* is the second worst method because even by applying the boosting technique, decision stumps (1-patterns) are still too simple to explain the relation between the features and the class. *SVM_RBF* improves the accuracy over *SVM_linear* because of its ability to form complex decision boundaries. However, it is still not able to clearly explain the model. Decision trees (*DT* and *fea_DT*) have acceptable performance because they are able to capture some predictive patterns in this data. However, they fail to extract all important patterns. Finally, *A-MPP* is able to identify most of the important patterns (achieving an accuracy of 98.33%) and it outperforms all other methods by a wide margin.

5.3 Comparing against Pattern based Classifiers

Table 2 shows the classification accuracy and the execution time (in seconds) for *two-phases*, *MBST*, *MPP* and *A-MPP*. The reported execution time is measured on a Dell Precision T7500 machine with an Intel Xeon 3GHz CPU and 16GB of RAM. All methods are implemented using matlab. The entries in the table with "NA" indicate that the method run for 15 hours without finishing! The last three columns in table 2 show the total number of closed patterns (the first phase of *two-phases*), the number of MPPs and the number of approximated MPPs.

Let us first discuss the classification accuracy of the different methods. We can see that *MPP* and *A-MPP* are the best performing methods on most datasets. Notice that the performance of *MBST* is not consistent. It performs very good on some datasets (e.g. breast cancer and mammographic), while it performs much worse than the other methods on other datasets (e.g. glass, pen-digits, tic-tac-toe and car).

Now Let us now focus on the efficiency aspect of the different methods. The results in table 2 show clearly that efficiency is a big concern for the *two-phases*, *MBST* and *MPP* methods. For instance, all of these methods fail to complete the mining on the mushroom, WDBC, image-seg and parkinson datasets using the 15 hours time budget. On the other hand, we can see that *A-MPP* is extremely fast (the longest execution time for *A-MPP* is on the mushrooms dataset, which took around 7 minutes).

Let us consider the lymphography dataset. Mining all closed frequent patterns took 23,473 seconds (around 6.5 hours). Applying the lossless pruning in *MPP* helps reducing the execution time to 6,293 seconds. Finally, applying the lossy pruning made the mining almost instant (4 seconds). It is interesting to see that this huge computational savings was at the cost of missing only 2 out of 33 MPPs. In fact, for many datasets, *A-MPP* did not even miss any MPP!

These results show that *A-MPP* is a very accurate method and achieves orders of magnitude speed up over the state-of-the-art pattern based classification methods.

6. RELATED RESEARCH

Frequent pattern mining is a very important research area in data mining. The original framework [1] has been extended to mine several types of data, including temporal [30], time series [5], text [21] and graph datasets [17, 12].

A lot of research has been conducted to reduce the output size of frequent patterns. [4] proposed the concept of closed frequent patterns to perform lossless compression of frequent patterns. More powerful compression usually rely on lossy compression using techniques like maximal patterns [19], clustering [28], pattern profiles [29], MDL compression [23] or maximum entropy patterns [24]. However, most of these methods are employed in an unsupervised setting to summarize the large collection of frequent patterns using fewer patterns. The objective of our work is different because we are mainly interested in selecting patterns that are important for the classification task.

The usage of frequent patterns in classification has been explored by many recent studies. Earlier approaches mainly focused on *associative classification* [20, 18, 10, 27, 26], which mines predictive association rules and builds a rule-based classifier. The results in [20, 18] showed that associative classification can be more accurate than heuristic C4.5 [22]. Top-k rule mining [10] discovers top-k covering rules from high dimensional gene expression profiles. Harmony [27] uses an instance-centric approach and assures that the highest confidence rule for each training is included in the classifier. [26] applies a lazy classification philosophy and mines the rules on demand by focusing only on the test instance.

Recently, the focus was more on using discriminative frequent patterns to map the data into a new feature space in order to improve the quality of the classifier. Most of the methods [8, 12, 5] first mine all frequent patterns and then select the discriminative ones. [13, 9] are two very recent studies that do not apply the two-phases approach. The Model Based Search Tree (MBST) method [13] builds a decision tree using discriminative frequent patterns. Our experiments showed that we are able to outperform MBST in terms of both classification accuracy and efficiency. The Direct Discriminative Pattern Mining (DDPMine) method [9] is similar to [13] in that it mines the most discriminative patterns on progressively shrinking subsets of the data. The main difference is that DDPMine [9] uses the sequential covering paradigm instead of decision tree and performs some pruning to speed up the mining.

7. CONCLUSION

In this paper, we present a novel feature construction method using minimal predictive patterns. Our framework applies a statistical test to ensure that every pattern in the result offers a significant predictive advantage over all of its subpatterns. We show that this can effectively filter out many spurious patterns and produce a compact set of predictive patterns. Geometrically, each pattern P in the result represents a distinct region D_P in the feature space where the class distribution is surprisingly skewed and cannot be explained by the regions that contain D_P. Our classification approach is to identify these predictive regions, model them explicitly as new features and then learn a linear classifier in the new expanded space.

We present an algorithm to directly mine the MPPs. Our algorithm works in a level-wise fashion by examining the general patterns first and gradually testing and adding more specific patterns to the result. In contrast to the traditional two-phases approach, we integrate pattern mining and feature selection by evaluating each pattern with respect to its subpatterns as soon as it is generated. Finally, we present a very efficient version of the algorithm to approximately mine the MPP set.

Experimental results clearly demonstrate the benefits of our approach by outperforming many well known classifiers. Besides, we show that applying the lossy pruning technique makes the mining process very efficient without sacrificing the classification performance.

8. ACKNOWLEDGMENT

This research work was supported by grants 1R21LM009102-01A1, 1R01LM010019-01A1, and 1R01GM088224-01 from the NIH. Its content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH.

9. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of SIGMOD*, 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of VLDB*, 1994.

[3] A. Asuncion and D. Newman. UCI machine learning repository, 2007.

[4] Y. Bastide, N. Pasquier, R. Taouil, G. Stumme, and L. Lakhal. Mining minimal non-redundant association rules using frequent closed itemsets. In *Proceedings of the International Conference on Computational Logic*, 2000.

[5] I. Batal, L. Sacchi, R. Bellazzi, and M. Hauskrecht. Multivariate time series classification with temporal abstractions. In *FLAIRS*, 2009.

[6] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. The Wadsworth Statistics/Probability Series, 1984.

[7] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001.

[8] H. Cheng, X. Yan, J. Han, and C. wei Hsu. Discriminative frequent pattern analysis for effective classification. In *Proceedings of ICDE*, 2007.

[9] H. Cheng, X. Yan, J. Han, and P. S. Yu. Direct discriminative pattern mining for effective classification. In *Proceedings of ICDE*, 2008.

[10] G. Cong. Mining top-k covering rule groups for gene expression data. In *Proceedings of SIGMOD*, 2005.

[11] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 1995.

[12] M. Deshpande, M. Kuramochi, N. Wale, and G. Karypis. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Transactions on Knowledge and Data Engineering*, 17:1036–1050, 2005.

[13] W. Fan, K. Zhang, H. Cheng, J. Gao, X. Yan, J. Han, P. Yu, and O. Verscheure. Direct mining of discriminative and essential frequent patterns via model-based search tree. In *Proceedings of SIGKDD*, 2008.

[14] Y. Freund and R. Schapire. A short introduction to boosting. *J. Japan. Soc. for Artif. Intel.*, 14(5):771–780, 1999.

[15] L. Geng and H. J. Hamilton. Interestingness measures for data mining: A survey. *ACM Comput. Surv.*, 38(3), 2006.

[16] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of the ACM SIGMOD international conference on Management of data*, 2000.

[17] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *Proceedings of ICDM*, 2001.

[18] W. Li, J. Han, and J. Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In *Proceedings of ICDM*, 2001.

[19] D.-I. Lin and Z. M. Kedem. Pincer-search: A new algorithm for discovering the maximum frequent set. In *Proceedings of the international conference on Extending Database Technology*, pages 105–119, 1997.

[20] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Knowledge Discovery and Data Mining*, pages 80–86, 1998.

[21] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, C. Watkins, and B. Scholkopf. Text classification using string kernels. *Journal of Machine Learning Research*, 2:563–569, 2002.

[22] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.

[23] L. M. Siebes A, Vreeken J. Item sets that compress. In *Proceeding of SIAM international conference on data mining*, 2006.

[24] N. Tatti. Maximum entropy based significance of itemsets. *Knowledge Information System*, 17(1):57–77, 2008.

[25] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, NY, 1995.

[26] A. Veloso, W. Meira Jr., and M. J. Zaki. Lazy associative classification. In *Proceedings of ICDM*, 2006.

[27] J. Wang and G. Karypis. Harmony: Efficiently mining the best rules for classification. In *Proceedings of SDM*, 2005.

[28] D. Xin, J. Han, X. Yan, and H. Cheng. Mining compressed frequent-pattern sets. In *Proceedings of VLDB*, 2005.

[29] X. Yan, H. Cheng, J. Han, and D. Xin. Summarizing itemset patterns: a profile-based approach. In *Proceedings of SIGKDD*, 2005.

[30] M. J. Zaki. Spade: an efficient algorithm for mining frequent sequences. In *Machine Learning Journal*, pages 31–60, 2001.