

A Bayesian Approach to Use Emerging Patterns for Classification

Hongjian Fan

Kotagiri Ramamohanarao

Department of Computer Science and Software Engineering
The University of Melbourne,
Parkville, Victoria 3010, Australia,
Email: {hf an, rao}@cs.mu.oz.au

Abstract

Emerging Patterns (EPs) are itemsets (characteristics) whose supports change significantly from one data class to another. This work proposes a novel approach to use EPs as a basic means for classification. It is called Bayesian Classification based on Emerging Patterns (BCEP). As a hybrid of the EP-based classifier and Naive Bayes (NB) classifier, it provides several advantages. First, it is based on theoretically well-founded mathematical models to predict an unseen case given a training sample. Second, it extends NB by using essential emerging patterns to relax the strong attribute independence assumption. Lastly, it is easy to interpret, as many unnecessary EPs are pruned based on data class coverage. An empirical study carried out on 21 benchmark datasets from the UCI Machine Learning Repository shows that our method is superior to other state-of-the-art classification methods such as C5.0, NB, CAEP and LB in terms of overall predictive accuracy.

Keywords: emerging patterns, Bayesian learning, classification, approximation, complexity

1 Introduction

The task of supervised classification - i.e., learning to predict class memberships of test cases given labelled training cases - has been studied substantially in statistics, machine learning, neural networks and expert systems over decades (Duda & Hart 1973, Kononenko 1991, Quinlan 1993, Cheeseman & Stutz 1996, Kohavi 1996, Domingos & Pazzani 1996, Domingos & Pazzani 1997, Webb & Pazzani 1998, Zhang, Ling & Zhao 2000). The proliferation of database management systems (DBMS) has contributed to recent massive gathering of all sorts of information, producing large databases with high dimensionality. Since the widely used traditional statistical data analysis techniques are not sufficiently powerful for building accurate and efficient classifiers on large and high-dimensional datasets (Fayyad, Piatetsky-Shapiro & Smyth 1996), the problem of classification has been an important research topic in database and KDD (*Knowledge Discovery in Databases*, or data mining) communities recently (Chen, Han & Yu 1996).

Bayes' theorem tells us how to optimally predict the class of a previously unseen example, given a training sample. The chosen class should be the one which maximizes

$$P(C_i|T) = \frac{P(T, C_i)}{P(T)} = \frac{P(C_i)P(T|C_i)}{P(T)}$$

Copyright ©2003, Australian Computer Society, Inc. This paper appeared at Fourteenth Australasian Database Conference (ADC2003), Adelaide, Australia. Conferences in Research and Practice in Information Technology, Vol. 17. Xiaofang Zhou and Klaus-Dieter Schewe, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

where C_i is the class label, $T = \{a_1 a_2 \dots a_n\}$ is the test case, $P(Y|X)$ denotes the conditional probability of Y given X , and probabilities are estimated from the training sample. Since classification focuses on discriminate prediction of a single class, rather than assigning explicit probabilities to each class, the denominator $P(T)$, which does not affect the relative ordering of the classes, can be omitted. So the class is chosen with the highest probability $P(T, C_i) = P(C_i)P(T|C_i)$. Because in practice it is very hard to calculate the exact probability $P(T, C_i)$, one must use approximations under some certain assumptions.

The Naive Bayesian (NB) classifier (Duda & Hart 1973) makes a simple assumption that all attributes are conditionally independent given the class C_i , hence,

$$P(T, C_i) = P(a_1 a_2 \dots a_n | C_i) = P(C_i)P(a_1|C_i)P(a_2|C_i) \dots P(a_n|C_i) = \frac{\prod_{j=1}^n P(a_j, C_i)}{P(C_i)^{n-1}}$$

Essentially, this means that the classification solely depends on the values of $P(a_j, C_i)$ and $P(C_i)$. Despite its simplicity, NB is a surprisingly successful classification method that has demonstrated to outperform much more complicated methods such as decision tree, rule learning and instance-based learning algorithms in many application domains. However, the assumption that the attributes are independent with respect to the class variable is almost certainly false in many induction scenarios.

Large Bayes (LB) (Meretakakis & Wuthrich 1999) relaxes the strong independence assumptions implied by NB. It uses the supports of interesting long, frequent (large) itemsets to approximate probabilities. The probability $P(T, C_i)$ can be estimated using different product approximations, where each product approximation assumes different independence of the attributes (Lewis 1959). For instance, $P(a_1 a_2 a_3 | C_i)P(a_4 a_5 | a_1 C_i)$ and $P(a_1 a_2 a_3 | C_i)P(a_5 | a_2 C_i)P(a_4 | a_1 a_5 C_i)$ are both product approximations of $P(a_1 a_2 a_3 a_4 a_5 | C_i)$, where the first product approximation uses $\{a_1 a_2 a_3\}$ first and then $\{a_1 a_4 a_5\}$; the second uses $\{a_1 a_2 a_3\}$, $\{a_2 a_5\}$ and $\{a_1 a_4 a_5\}$ sequentially. LB uses only interesting large itemsets to make the approximation reliable. It selects a certain product approximation by selecting certain large itemsets. LB's accuracy was claimed to be consistently better than that of NB, and generally better than that of the widely known and used decision tree classifier C4.5 (Quinlan 1993) and TAN (a Bayesian network extension of NB).

Recently a new type of knowledge patterns, called emerging patterns (EPs) (Dong & Li 1999), were introduced for knowledge discovery from databases (KDD). EPs are defined as itemsets whose supports change *significantly* from one data class to another.

EPs capture multi-attribute contrasts between two classes of things, such as edible mushrooms vs poisonous mushrooms, normal tissues vs cancer tissues. The significance of the difference is measured by the magnitude of the frequency-change ratio (called growth rate) of the patterns from one class to another. Usually the large the growth rate, the stronger discriminating power the patterns have.

EP	Malignant-support	Benign-support	growth rate
e_1	0.41%	20.31%	49.54
e_2	0%	3.28%	∞

$e_1 = \{(\text{Bare-Nuclei}, 1), (\text{Bland-Chromatin}, 3), (\text{Normal-Nucleoli}, 1), (\text{Mitoses}, 1)\},$
 $e_2 = \{(\text{Marginal-Adhesion}, 1), (\text{Bare-Nuclei}, 1), (\text{Normal-Nucleoli}, 2)\},$
 where an $(\text{attribute}, \text{value})$ pair denotes an attribute-value.

Figure 1: Examples of Emerging Patterns

For example, Figure 1 shows two EPs from the Malignant class to the Benign class of the Wisconsin-breast-cancer dataset, from the UCI Machine Learning Repository (Blake & Merz 1998). The EP e_1 , with a growth rate of 49.54, is a four-attribute feature contrasting the benign instances against the malignant instances. It has very high predictive power: the odds that instances containing or satisfying e_1 are benign is 98%. The EP e_2 has even greater predictive power: the odds that instances containing e_2 are benign is 100%. More specifically, e_2 is a *jumping* emerging patterns (JEP¹) with support 0 in the malignant class and thus growth rate ∞ .

By aggregating the differentiating power of EPs/JEPs classification systems such as JEP-Classifier (Li, Dong & K. Ramamohanarao 2000) and CAEP (Classification by Aggregating Emerging Patterns) (Dong, Zhang, Wong & Li 1999) usually achieve higher accuracy than other state-of-the-art classifiers such as C5.0 (Rulequest 2000) and CBA (Classification Based on Associations).

In this paper we describe a hybrid approach that attempts to utilize the advantages of both EP-based classifiers (i.e., EP's sharp differentiating power) and Naive Bayes (evidence accumulation from multiple attributes). The proposed classifier is called BCEP, i.e., Bayesian Classification based on Emerging Patterns. Experiments carried out on a large and varied selection of benchmark datasets from the UCI Machine Learning Repository shows that BCEP is superior to other state-of-the-art classification methods such as C5.0, NB, CAEP and LB in terms of overall predictive accuracy, and it achieves the best or very close (within 1%) to the best accuracy on 15 out of 21 datasets and also performs very well on the remaining six.

An EP of a class C_i can be regarded as a distinguishing feature of C_i . Its support sup_i in C_i estimates the probability that the pattern occurs given the class label C_i . And its support sup'_i in non- C_i (the contrasting data class) estimates the probability that the pattern occurs given the contrasting class. EPs are itemsets which maximize the former probability and minimize the latter. EPs express the relationships between items(attributes), indicating that those items(attributes) contained in a single EP are not actually independent. Recall that if X and Y

¹A JEP is a special type of emerging pattern (also a special type of discriminant rule), defined as an itemset whose support increases abruptly from zero in one dataset, to non-zero in another dataset — the ratio of support-increase being infinite.

are independent given Z then $P(X|Z, Y) = P(X|Z)$. As an extreme case, an item a_1 appears in half of class C_1 and an item a_2 appears in the other half of class C_1 (therefore, a_1 and a_2 can not appear at the same time in C_1); while both a_1 and a_2 appear in the same half of class C_2 . Then $\{a_1, a_2\}$ is an EP with support 50% in C_2 and 0% in C_1 . Obviously, $P(a_1|C_1, a_2) = 0$, $P(a_1|C_1) = 0.5$, $P(a_1|C_2, a_2) = 0$, $P(a_1|C_2) = 0.5$. So $P(a_1|C_1, a_2) \neq P(a_1|C_1)$ and $P(a_1|C_2, a_2) \neq P(a_1|C_2)$. We can see that both a_1 and a_2 are not independent given either C_1 or C_2 . By using EPs in the product approximation, we can relax the strong independence assumption implied by NB, which assumes all attributes are conditionally independent given the class C_i .

There can be a very large number (e.g., 10^9) of general EPs E_i from class C_i in the dense and high-dimensional datasets of a typical classification problem. It has been shown that many of them are not so useful in classification (Fan & Ramamohanarao 2002). Here we are only interested in a special type of Emerging Pattern, called essential emerging patterns (eEP), which are believed to be the most useful patterns for classification. eEP are EPs with very large growth rates (typically more than 1000), enough (large) supports in the target class (usually a threshold 1%), and that are contained in the left bound of the border representing the EP collection. Intuitively, large growth rates ensure EP's sharp discriminating power, large supports, which means enough coverage on the training dataset, make EPs more resistant to noise, and the boundary requirement states that any proper subset of a boundary itemset is no longer an eEP. eEPs represent the essence of the discriminating knowledge. We utilize the tree-based algorithms (Fan & Ramamohanarao 2002) to efficiently mine the complete set of eEPs for each class.

Although the set of eEPs is much smaller than the set of general EPs, there still exist redundant eEPs in terms of data class coverage. Suppose an EP e_i covers² a set COV_{e_i} of instances in the training data class C_i , another EP f_i covers a set COV_{f_i} of instances in C_i . If COV_{f_i} is the subset of COV_{e_i} , we can see f_i does not provide more useful information, as both of their growth rates are very large and the support of e_i is larger than that of f_i . So f_i can be safely removed without losing much classification information. This pruning based on the concept of data class coverage is pursued as the last step in the training phase after the complete set of eEPs for each class are mined. By further pruning unnecessary EPs effectively, only a small set of high quality EPs F_i from class C_i is selected. Hence the resulting classifier's complexity is reduced and its efficiency is improved (shorter classification time). Moreover, in our experiments, better accuracy can be achieved because redundant and noisy information is deleted.

When a new case $T = \{a_1, a_2 \dots a_n\}$ arrives to be classified, BCEP combines the evidence provided by the subsets of T that are present in F_i to approximate $P(T, C_i)$, where F_i denotes the final high quality EPs from class C_i for classification and $P(T, C_i)$ determines the conditional probability $P(C_i|T)$, i.e., the probability that the case belongs to class C_i given the evidence. The evidence which is selected from F is denoted as B , where $F = F_1 + \dots + F_c = \sum_{i=1}^c F_i$, c is the number of classes in the training dataset. An EP of F_i with its high support in the target class C_i and low support in the contrasting class C_j ($j \neq i$) can be seen as a strong signal which implies the items contained in the EP are dependent either given C_i or C_j . And the strength of such a signal is expressed by its

²Here we say an EP "covers" an instance if the instance contains the EP.

growth rate and its support in the target class C_i . We select EPs from B in the strength-descending-order. Such order is important because different combinations of itemsets of B lead to different product approximation, and even when the same itemsets are used in different order, the product approximations are still different. The EPs of B are used sequentially to construct the product approximation of $P(T, C_i)$ and their class supports in both the target and the background data class are used to calculate the probabilities. The result is the class with the highest value of $P(T, C_i)$. Our experiments show that selecting and aggregating a small number of essential EPs which provide discriminating knowledge is an effective strategy to compute the class probabilities.

We highlight our main contributions in this paper as follows.

1. BCEP can be thought of as a new extension of NB. By using emerging patterns of arbitrary size when estimating $P(T, C_i)$, BCEP relaxes the strong attribute independence assumption implied by NB while retains NB's strength: superior accuracy in many cases over previously published classifiers.
2. BCEP is based on theoretically well-founded mathematical model to compute the probability that the case belongs to some class given the evidence. The scoring functions of previous EP-based classifiers are coarse approximations of such probability. Because of our method's mathematical soundness, we can use few EPs to build high accuracy classifiers.
3. BCEP solves the problem of interpretability, which is an important issue in data mining. Compared with previous EP-based classifiers using thousands of EPs, BCEP relies on much fewer (a few hundred) EPs to make a decision and outperforms those EP-based classifiers, because the idea of data class coverage is introduced to prune many unnecessary EPs. Although the idea itself is not new, our approach is the first application of using it to prune EPs for classification and has been shown successful.

Organization: An outline of the remainder of this paper is as follows. Section 2 reviews related work and motivates our work. Section 3 defines the essential emerging patterns (eEPs) In section 4 the idea of data class coverage is introduced to further prune a lot of unnecessary EPs. Section 5 details our Bayesian approach to use EPs for classification. Section 6 presents an extensive experimental evaluation of *BCEP* on popular benchmark datasets from the UCI Machine Learning Repository repository and compares its performance with C5.0, NB, CAEP and LB. Finally, in section 7 we provide a summary and discuss future research issues.

2 Related Work and Motivation

BCEP can be thought of as a hybrid system of the EP-based classifier family and the Bayesian classifier family.

The EP-based classifiers, such as CAEP and JEP-Classifier, aggregate each individual EP's sharp differentiating power to compute the aggregate scores for each class. The result is the class with the highest value of such scores. The major differences between EP-based classifiers and BCEP are as follows:

- The EP-based classifiers usually depend on a huge number of mined EPs. For example, CAEP

uses an average of 9940 EPs³ and JEP-Classifier uses an average of 6397 JEPs⁴. Such huge number of EPs make the resulting classifiers complex. BCEP utilizes the idea of data class coverage with respect to E_i (the EPs of the class C_i) to effectively prune unnecessary EPs. The result is a small set (an average of 398, which is calculated based on Table 2) of high quality EPs F_i from class C_i . Experimental results show that such pruning does not lose any useful patterns for classification and actually removes noise. We also notice fewer EPs lead to shorter classification time.

- The approaches to use EPs for classification are different. We take CAEP for example to explain how it works. (JEP-Classifier works in a similar way) The contribution of an EP e is the product of two terms. The first term is the conditional probability that an instance is in class C_i given that the instance contains e ; the second term is the fraction of the instances of class C_i that e applies. CAEP sums the contributions of the individual EPs to obtain a score S_i for class C_i and then "normalize" S_i by dividing it using a base score at certain percentage for the training instances of class C_i . CAEP's approach has two weaknesses.

1. Repeated contributions are counted. Suppose two EPs which cover almost the same percentage of the training sample, the contributions of both of them are added, which is undesirable. Our pruning technique minimizes such repeated contributions.
2. The normalization is somewhat intuitive. The normalization is introduced to resolve the problem caused by unbalanced distributions of EPs for different classes, i.e., a class which has many more EPs tends to get higher scores. Although it is successful, how to choose the base score remains the art of human.

BCEP is based on theoretically well-founded mathematical models for discriminating classification learning. BCEP uses EPs of arbitrary size when estimating $P(T, C_i)$. Under different independence assumptions about the attributes, the probability $P(T, C_i)$ can be obtained using different product approximations. EPs are combined using the chain rule of probability and all necessary independence assumptions are assumed true.

- JEP-Classifier uses exclusively *jumping* emerging patterns (JEPs) and only their supports are aggregated to compute the scores. JEP-Classifier performs well when there are many JEPs in each class of dataset. However, in real world classification problems, some data classes may contain few or even no JEPs whose supports meet a reasonable threshold (such as 1%). In such cases, EPs with large growth rate will play an important role.

Although partially influenced by LB, BCEP is different from LB in that BCEP uses EPs, which contain more useful information than frequent itemsets. EPs require an itemset frequent in one data class while very infrequent in another. The approaches used to choose itemsets or EPs and use them in approximation are also very different.

³The number is calculated based on Table 5.2 on page 97 from (Zhang 2001)

⁴The number is calculated based on Table 5.2 on page 84 from (Li 2000)

3 Essential Emerging Patterns and Classification

Suppose a data object $obj = (a_1, a_2 \cdots a_n)$ follows the schema $(A_1, A_2 \cdots A_n)$, where $A_1, A_2 \cdots A_n$ are called attributes. Attributes can be categorical or continuous. For a categorical attributes, we assume that all the possible values are mapped to a set of consecutive positive integers. For a continuous attributes, we assume that its value range is discretized into intervals, and the intervals are also mapped to consecutive positive integers. By doing so, a raw set of data objects is encoded into the binary transaction database where emerging patterns are defined upon. We call each (attribute, integer-value) pair an *item*.

Let $C = \{c_1, \cdots, c_m\}$ be a finite set of *class labels*. A *training dataset* is a set of data objects such that, for each object obj , there exists a class label $c_{obj} \in C$ associated with it. A classifier is a function from $(A_1, A_2 \cdots A_n)$ to C , which assigns a class label to an unseen example.

In general, given a training dataset, the task of **classification** is to build a classifier from the training dataset such that it can be used to predict class labels of unknown objects. **Classification** is also known as **supervised learning** as the learning of the model is “supervised“ in that it is told to which class each training example belongs. Emerging patterns can serve as a classification model because they represent knowledge which discriminates between different classes of datasets.

Let I denote the set of all items in the encoding dataset D . A set X of items is also called an itemset, which is defined as a subset of I . We say any instance S contains an itemset X , if $X \subseteq S$. The support of an itemset X in a dataset D , $supp_D(X)$, is $count_D(X)/|D|$, where $count_D(X)$ is the number of instances in D containing X .

Definition 1 Given two different classes of datasets D' and D'' , the growth rate of an itemset X from D' to D'' is defined as

$$GrowthRate(X) = GR(X) = \begin{cases} 0 & \text{if } supp'(X) = 0 \text{ and } supp''(X) = 0 \\ \infty & \text{if } supp'(X) = 0 \text{ and } supp''(X) > 0 \\ \frac{supp''(X)}{supp'(X)} & \text{otherwise} \end{cases}$$

Emerging Patterns are the itemsets with large growth rate from D' to D'' .

Definition 2 Given a growth rate threshold $\rho > 1$, an itemset X is said to be an ρ -*emerging pattern* (ρ EP or simply EP) from a background dataset D' to a target dataset D'' if $GrowthRate(X) \geq \rho$.

When D' is clear from context, an EP X from D' to D'' is simply called an EP of D'' . The support of X in D'' , $supp_{D''}(X)$, denoted as $supp(X)$, is called the support of the EP.

Emerging Patterns can be described by borders (Dong & Li 1999). A collection of sets represented by the border $\langle L, R \rangle$ is

$$[L, R] = \{Y \mid \exists X \in L, \exists Z \in R, X \subset Y \subset Z\}.$$

For instance the border $\langle \{\{1\}, \{2\}\}, \{1, 2, 3, 4\} \rangle$ represents those sets which are either supersets of $\{1\}$ and subsets of $\{1, 2, 3, 4\}$ or supersets of $\{2\}$ and subsets of $\{1, 2, 3, 4\}$. Clearly, borders are usually much smaller than the collections they represent. The collection of emerging patterns discovered from different classes of data, can be concisely represented by their border $\langle L, R \rangle$, where L is the sets of the minimal itemsets and R is the sets of the maximal itemsets.

After introducing general Emerging Patterns, we now define a special type of EP, called essential Emerging Patterns (eEP), which are the most useful patterns for classification. eEP are EPs satisfying the following conditions:

Condition 1 Their growth rates are very large (typically more than 100-1000, depending on the size of dataset);

Condition 2 They have enough supports in the target class (usually a threshold 1%);

Condition 3 They are contained in the left bound of the border representing the EP collection, i.e., they are minimal EPs.

We believe EPs satisfying the above three conditions are the most expressive patterns for classification for the following reasons.

1. Very large or even infinite growth rates ensure EPs' significant level of discrimination.
2. The support threshold makes an EP cover at least a certain number of examples in the target class of training dataset, hence reliable to be used in the product approximations. Itemsets with too low supports are regarded as noise.
3. EPs in the left bound of the border have no subsets within the collection of EPs. That is, any proper subset of such an EP is not an EP any more. Such EPs are the *shortest*. Consider that EPs are actually itemsets. A shorter EP means less items (attributes). If we can use less attributes to distinguish two data classes, adding more attributes will not contribute to classification, and even worse, bring noise. In our approach, as many EPs as possible should be used, i.e., the product approximations should contain as many factors as possible. So short EPs are also desirable.
4. Supersets of eEPs are not useful for classification because of the following reason. Let e_1 be an eEP satisfying all of the three conditions (**Condition 1, 2 and 3**), e_2 be an itemset satisfying only the first two conditions (**Condition 1 and 2**), and $e_2 \supset e_1$. Since both of them have very large (hence enough) growth rates, we are only concerned about their coverage on the training dataset. e_1 covers more (at least equal) instances of the target class than e_2 because $supp(e_1) \geq supp(e_2)$.

The learning phase of *BCEP* employs algorithms using a tree based data structure to efficiently mine essential EPs eE_i from class C_i , namely itemsets passing the user-specified support and growth rate thresholds, and the minimal requirement (**Condition 1, 2 and 3**).

4 Further Pruning Essential Emerging Patterns Based on Data Class Coverage

The number of the eEPs generated in the learning phase can be huge. To make the classification effective and efficient, we need to further prune unnecessary eEPs to delete redundant and noisy information. The pruning is based on a global order defined on the EPs.

Definition 3 Given two eEPs, e_i and e_j ($i \neq j$), e_i is said to have higher rank than e_j (also called e_i precedes e_j), denoted as $e_i \prec e_j$, if and only if

1. the support of e_i is greater than that of e_j , or

2. their supports are the same, but the length of e_i is smaller than that of e_j .

Note the above order does not consider the growth rates of eEPs. It is because eEPs are defined as EPs with very large growth rates, and it does not make much sense to further compare two large values.

Let eE_i be the set of eEPs for class C_i , and D_i the training data (belonging to class C_i). The basic idea is to choose a set of high precedence eEPs in eE_i to cover D_i . This method is related to the traditional covering method. However, the major difference is in the rules or patterns that they use. BCEP learns all patterns from the entire training data using exhaustive search, while each rule in the traditional covering method is learned using a heuristic method from the remaining data after the examples covered by previous rules are deleted (high quality rules may lose due to greedy heuristic).

Algorithm 1 (Prune EPs based on data class coverage)

Input: a set of EPs eE_i for class C_i ,
the training dataset D_i from C_i

Output: a final set of EPs F_i for classification

Method:

1. Sort eE_i by rank in descending order;
2. While both D_i and eE_i are not empty,
for each EP e in the rank descending order,
find all data objects containing e .
If e can correctly classify at least one object
then select e and remove those objects of D_i
containing e .

Figure 2: Prune EPs based on data class coverage

The algorithm to prune emerging patterns eE_i based on data class coverage with respect to class C_i is given in Figure 2. It removes one data object from the training dataset immediately after it is covered by some selected EPs. The pruning is based on the assumption that if each tuple in the training dataset is covered by a given set of high quality EPs, a new test should also be covered by the same set of EPs, because the training and testing dataset supposedly have the same distribution. We understand that such assumption is not always true in the real world classification problems, so we can choose more EPs by modifying the algorithm a little to let a data object stay there until it is covered by at least μ ($\mu \geq 2$) EPs.

There are two reasons why we need more EPs. First, if too few EPs are chosen after pruning, some effective EPs may be lost. Second, more EPs are desirable in our Bayesian approach, because generally the more itemsets we use in the product approximation, the more reliable the product approximation is for classifying new data objects. There are also reasons why we prefer fewer EPs (that is why we prune EPs), which have been discussed in the beginning of the section. This raises the following question, "How to determine μ ?" Fortunately, after conducting experiments on a few datasets from the UCI Machine Learning Repository, we find that classification accuracy improves when μ goes from 1 to 2 (more EPs are selected when $\mu = 2$ than $\mu = 1$), but it remains nearly stable when μ goes from 2 to 5. So as a trade-off, we let μ be 2.

As a byproduct of the pruning process, the percentage of D_i covered by one or more EPs of eE_i is available. If we find the percentage is high (typically more than 90%), we are much confident that we have mined enough high quality EPs for classification. However, in rare cases when such percentage is low (typically below 80%), where there are few or even no EPs with very large (more than 1000) growth rates, it means the thresholds for eEPs are too aggressive, hence we lose some high quality EPs. We have to adjust the growth rate threshold (reduce it to 10-100) and repeat the EP mining process to obtain a larger set of EPs. This can be done *automatically*: Repeat mining eEPs eE_i until eE_i covers enough percent of D_i .

To obtain the final set of high quality EPs F for classification, the pruning is done on all classes in the training dataset D . Let the number of classes in D is c ,

$$F = F_1 \cup \dots \cup F_c = \bigcup_{i=1}^c F_i.$$

5 The Bayesian Approach to Use Emerging Patterns for Classification

5.1 Basic ideas

Upon arrival of a new case $T = \{a_1, a_2, \dots, a_n\}$ to be classified, BCEP combines the evidence provided by the subsets of T that are present in F , where F denotes the final set of high quality EPs for classification. The evidence is denoted as B ,

$$B = \{s \in F | s \subset T\}.$$

BCEP uses the EPs of B to derive product approximations of $P(T, C_i)$ for all classes. The product approximation of the probability of an n -itemset T for a class C_i contains a sequence of at most n subsets of T such that each itemset contains at least one item not covered⁵ in the previous itemsets. Recall the general chain rule is

$$P(X_1, X_2, \dots, X_n) =$$

$$P(X_1)P(X_2|X_1) \dots P(X_n|X_1, \dots, X_{n-1}).$$

To obtain the product approximation of $P(T, C_i)$, the itemsets are combined using the chain rule of probability while assuming that all necessary attribute independence assumptions are true.

Suppose a test instance $T = \{a_1, a_2, a_3, a_4, a_5\}$ arrives. After consulting F , we find its corresponding $B = \{\{a_2, a_5\}, \{a_3, a_4\}, \{a_1, a_2, a_3\}, \{a_1, a_4, a_5\}\}$. We can use some itemsets of B to make several different product approximations of $P(T, C_i)$ as follows:

- I $\{a_1, a_2, a_3\}, \{a_1, a_4, a_5\} \implies P(C_i)P(a_1a_2a_3|C_i)P(a_4, a_5|a_1C_i)$
- II $\{a_1, a_4, a_5\}, \{a_2, a_5\}, \{a_3, a_4\} \implies P(C_i)P(a_1a_4a_5|C_i)P(a_2|a_5C_i)P(a_3|a_4C_i)$
- III $\{a_2, a_5\}, \{a_3, a_4\}, \{a_1, a_4, a_5\} \implies P(C_i)P(a_2a_5|C_i)P(a_3a_4|C_i)P(a_1|a_4a_5C_i)$
- IV $\{a_1, a_2, a_3\}, \{a_2, a_5\}, \{a_3, a_4\} \implies P(C_i)P(a_1a_2a_3|C_i)P(a_5|a_2C_i)P(a_4|a_3C_i)$

Note that $\{a_1, a_2, a_3\}, \{a_1, a_4, a_5\}$ and $\{a_2, a_5\}$ is not a product approximation since all items of $\{a_2, a_5\}$ are already covered by the first two itemsets. We also

⁵An item which is already included in the product approximation is said to be covered.

point out that although II and III use the same itemsets $\{a_2, a_5\}$, $\{a_3, a_4\}$ and $\{a_1, a_4, a_5\}$, the final product approximation is different because these itemsets are used in different order.

Clearly, different combinations of itemsets of B lead to different product approximation, and the product approximations are different even when the same itemsets are used in different order. The product approximation of $P(T, C_i)$ is created incrementally adding one EP at a time until no more EPs can be added (either all the items of the remaining EPs from B are already covered or no more EPs are available in B).

An EP with its high support in the target class and low support in the contrasting class can be seen as a strong signal indicating the class of a test instance containing it. And the strength of such a signal is expressed by its support in the target class.

Definition 4 The strength of an EP e_i of class C_i is defined as

$$strength(e_i) = \frac{GR(e_i)}{GR(e_i) + 1} * supp_i(e_i)$$

The more strength an EP has, the earlier it should be used in the product approximation as long as it provides items which have not been covered. We are also concerned about EP's length. When two EPs have the same strength, the shorter EPs should be used first in order to use as many as possible EPs in the product approximation,

EPs of B are first sorted in the strength-descending-order, and the final list is denoted as O . EPs are extracted from the list O from the beginning to incrementally construct the the product approximation. The set of covered items is denoted as cov . An EP p inserted in the product approximation should satisfy the following rules:

Rule 1: $|p - cov| \geq 1$;

Rule 1 means p contains new items which have not been covered. It guarantees that the product solution satisfies the chain rule and hence it is a valid product approximation.

Selection among alternatives is done as follows. p is selected instead of another EP q if the following rules are satisfied in the given order of importance:

Rule 2: $strength(p) > strength(q)$;

Rule 3: $length(p) < length(q)$;

Rule 4: $|p - cov| \leq |q - cov|$.

Rule 2 ensures EPs with larger strength (stronger signals) be used first if they do not break Rule 1. Rule 3 assures shorter EPs be considered first if there are alternatives with the same strength. This is equivalent to maximizing the number of EPs used in the product approximation. Rule 4 gives priority to those EPs among the remaining alternatives which contain the smallest number of not covered items. This is last attempt to make as many EPs as possible stay in the sequence.

5.2 The Algorithm to Compute the Product Approximation

After the final set of high quality EPs F is available, a new unlabelled test $T = \{a_1, a_2, \dots, a_n\}$ is classified by the Algorithm 2 in Figure 3. The supports in both classes and the strength of each EP can be calculated in the training phrase.

The algorithm incrementally builds the product approximation of $P(T, C_i)$ by adding one itemset

(EP) at a time until no more can be added. It first finds the evidence B provided by the subsets of T that are present in F . $covered$ is the subset of T already covered; $numerator$ and $denominator$ are the sets of itemsets in numerator and denominator, respectively. Procedure $Next(covered, B)$ (Figure 4) then repeatedly pickup next itemsets from B . The algorithm stops once all items in T have been covered. In the for loop, the numerator and denominator itemsets, B_i and $B_i \cap covered$ respectively, are stored in two separate set $numerator$ and $denominator$. Finally, they are used to derive the value of the product approximation of $P(T, C_i)$ for each class C_i . The most likely class is then returned.

Algorithm 2 (Bayesian Classification based on Emerging Patterns)

Input: the final set of EPs F for classification and a test instance T

Output: the classification c_i of T

Method:

1. $B = \{s \in F | s \subset T\}$
2. $covered = \emptyset$
3. $numerator = \emptyset$
4. $denominator = \emptyset$
5. for $(i = 1; covered \subset T; i++)$
 - {
 - $B_i = Next(covered, B)$
 - $numerator = numerator \cup B_i$
 - $denominator = denominator \cup \{B_i \cap covered\}$
 - $covered = covered \cup B_i$
 - }
6. for each class, compute

$$P(T, C_i) = P(C_i) \frac{\prod_{u \in numerator} P(u, C_i)}{\prod_{v \in denominator} P(v, C_i)}$$
7. output the class c_i with maximal $P(T, C_i)$

Figure 3: Bayesian Classification based on Emerging Patterns

Procedure $Next(covered, B)$ selects from B the next itemset to be used in the product approximation. This is uniquely determined by the Rules 1-4.

5.3 Zero Counts and Smoothing

The support (or observed frequency) of an itemset can be unreliable substitution for its probability, especially when the dataset is small or when the training dataset contains noise. A typical example is a jumping EP with a growth rate of $\infty (\frac{\geq 0}{0})$, where the zero-support in the background class is very unreliable, and it is very undesirable to use zero in the product approximation.

A standard statistical technique is to incorporate a small-sample correction into the observed probabilities. This is called smoothing and helps eliminate unreliable estimates and zero-counts. In our experiment, we use M-estimate with $m=2$ to estimate $P(X|C_i)$ and Laplace-estimate to estimate $P(C_i)$.

```

Next(covered, B)
  Z = {s ∈ B ∧ |s - covered| ≥ 1};
  return an itemset Bi ∈ Z
  such that for all other itemsets Bj ∈ Z:
1. strength(Bi) > strength(Bj);
2. strength(Bi) = strength(Bj) and length(Bi) <
   length(Bj);
3. strength(Bi) = strength(Bj) and length(Bi) =
   length(Bj) and |Bi - covered| ≤ |Bj - covered|.

```

Figure 4: Procedure $Next(covered, B)$

M-estimate:

$$\frac{\#(X, C_i) + n_0 \frac{\#(X)}{|D|}}{|D_i| + n_0}$$

where $\#(X, C_i)$ denotes the number of training examples belonging to class C_i and containing itemset X ; $\#(X)$ denotes the number of training examples containing itemset X ; n_0 is a small number which is set to 5.

Laplace-estimate:

$$\frac{|D_i| + k}{|D| + c * k}$$

where $|D_i|$ is the number of training objects belonging to C_i ; $|D|$ is the total number of training objects; c is the number of classes; and k is normally 1.

Let $P(X, Y, C_i)$ and $P(Y, C_i)$ be the observed frequencies in D of $\{X, Y, C_i\}$ and $\{Y, C_i\}$ respectively (X and Y are itemsets). Instead of $P(X|Y, C_i) = P(X, Y, C_i)/P(Y, C_i)$, we use the smoothed conditional probability:

$$P(X|Y, C_i) = \frac{|D| * P(X, Y, C_i) + 5 * P(X)}{|D| * P(Y, C_i) + 5}$$

6 Experimental Evaluation

In order to investigate BCEP's performance compared to that of other classifiers, we carry experiments on 21 datasets from the UCI Machine Learning Repository. We compare BCEP with Naive Bayes and state-of-the-art classifiers: the widely known decision tree induction C5.0; an EP-based classifier CAEP; and LB, a recently proposed classifier extending NB using long itemsets.

All the experiments were performed on a 500Mhz Pentium III PC with 512Mb of memory. The accuracy was obtained by using the methodology of *stratified* ten-fold cross-validation (CV-10). We use the Entropy method in (Fayyad and Irani 1993) taken from the MLC++ machine learning library (Kohavi, John, Long, Manley & Pfleger 1994) to discretize datasets containing continuous attributes.

Table 1 summarizes the accuracy results. Columns 1,2,3 and 4 describe the datasets: the name of each dataset and the number of instances, attributes and classes, respectively. Columns 5 to 9 give the predictive accuracy of the classifiers.

Keeping in mind that no classification method can outperform all others in all possible domains, we can draw some interesting points from Table 1 as follows:

- Our BCEP performed perfectly (98% to 100% accuracy) on some datasets (chess, mushroom, shuttle, shuttle-small, tic-tac).
- In comparison with NB, BCEP consistently achieves higher accuracies. For the 21 datasets, BCEP wins on 20 while NB wins only on the splice dataset (please note BCEP's accuracy is very close to that of NB). This confirms our belief that BCEP really relaxes the strong attribute independence assumption implied by NB.
- Compared with C5.0, BCEP achieves higher accuracies. For the 21 datasets, BCEP wins on 15; C5.0 wins 6; they achieve the same accuracy on the mushroom and shuttle-small dataset.
- In comparison with CAEP, BCEP also achieves higher accuracies. For the 14 datasets where results of CAEP are available, BCEP wins on 9; CAEP wins 4; they achieve the same accuracy on the German dataset. This shows our Bayesian approach to approximate the class probabilities is better than the scoring functions of CAEP.
- Finally we compare BCEP with LB, a NB extension using large itemsets. BCEP produces better results. For the 17 datasets where results of LB are available, BCEP wins on 11; LB wins on 6. We believe it is because EPs contain more useful information than large itemsets when used in the product approximations.

Table 2 shows the effect of applying data class coverage to prune EPs. It compares the accuracy, the number of EPs used in classification, and the classification time (average one fold time over the 10 CV-folds) when pruning with those when not pruning (w/o pruning). It can be seen that the number of EPs has been dramatically reduced after the pruning process, which leads to much shorter classification time, while in almost all cases (except the German and waveform-21 datasets) there is an increase in the accuracy. For example, before pruning the splice dataset contains 10289 EPs and it takes BCEP 107 seconds to finish one classification fold with a accuracy of 91.14%. However, the pruning reduces the number of EPs down to 360 (around 3.5% of 10289 before reduction) and BCEP only needs 5 seconds (about 20 times faster) finishing one fold with a higher accuracy of 94.1%. We highlight some interesting points from Table 2.

- Many EPs are unnecessary or redundant for classification, because by pruning 50%-95% EPs, there is no loss in predictive accuracy, and often there is an increase in accuracy. Furthermore, some removed EPs are noisy, since using those EPs in classification leads to a decrease in accuracy, although the decrease is slight.
- Classification based on as few EPs as possible is desirable, as long as the small set of EPs provide sufficient information for prediction, because it is ten or even a hundred times faster to use fewer EPs to classify a test instance.

Table 1: Description of datasets and summary of results

Dataset	Dataset Properties			Accuracy				
	#Instances	#Attributes	#Classes	NB	C5.0	CAEP	LB	BCEP
Adult	45,225	14	2	84.12	85.54	83.09	85.11	85
Australian	690	14	2	85.65	84.93	85.51	85.65	86.4
Chess	3,169	36	2	87.15	99.31	85.45	90.24	98.56
Cleve	303	13	2	82.78	77.16	-	82.19	82.41
Diabete	768	8	2	75.13	73.03	-	76.69	76.8
German	1,000	20	2	74.1	71.90	74.5	74.8	74.5
Heart	270	13	2	82.22	76.30	82.22	82.22	81.85
Iono	351	34	2	89.45	91.45	89.76	-	93.24
Letter	20,000	16	26	74.94	88.06	-	76.4	84.28
Lymph	148	18	4	81.86	78.29	74.38	84.57	83.13
Mushroom	8124	22	2	99.68	100	93.91	-	100
Pima	768	8	2	75.9	75.39	77.6	75.77	75.66
Satimage	6,435	36	6	81.8	86.74	-	83.9	87.42
Segment	2,310	19	7	91.82	96.88	85.76	94.16	95.15
Sonar	208	60	2	75.4	76	78.33	-	78.4
Shuttle-small	5,800	9	7	98.7	99.65	-	99.38	99.65
Splice	3,190	59	3	94.64	94.2	-	94.64	94.1
Tic-tac	958	9	2	70.15	85.91	85.91	-	99.37
Vehicle	846	18	4	61.12	73.68	68.8	55.92	68.05
Waveform-21	5,000	21	3	78.51	75.62	83.92	79.43	82.25
Yeast	1,484	8	10	58.05	56.14	-	58.16	58.22
Average				81.1	83.15	82.08	81.13	84.97

Table 2: Effect of EP pruning based on data class coverage

Dataset	BCEP Accuracy		#EPs		classification time(sec's)	
	w/o prune	prune	w/o prune	prune	w/o prune	prune
Adult	83.94	85	3316	1115	774.123	340.613
Australian	85.59	86.4	1009	126	0.343	0.052
Chess	96.98	98.56	1078	56	9.341	1.378
Cleve	80.69	82.41	461	72	0.055	0.011
Diabete	75.4	76.8	62	19	0.011	0.010
German	74.8	74.5	807	77	0.316	0.024
Heart	81.48	81.85	103	36	0.012	0.006
Iono	90	93.24	2732	48	0.775	0.026
Letter	82.47	84.28	71714	3628	1145.97	118.083
Lymph	81.25	83.13	102	26	0.005	0.002
Mushroom	100	100	1958	36	240.846	13.634
Pima	74.2	75.66	54	21	0.008	0.006
Satimage	86.23	87.42	77684	1247	1560.57	57.819
Segment	93.68	94.76	11885	211	11.399	0.803
Sonar	76.5	78.4	699	41	0.227	0.016
Shuttle-small	99.65	99.65	378	45	11.094	3.253
Splice	91.14	94.1	10289	360	107.218	5.185
Tic-tac	86.63	99.37	704	27	0.177	0.008
Vehicle	66.22	68.05	1377	81	0.609	0.034
Waveform-21	82.71	82.25	5500	1069	29.35	6.575
Yeast	58.08	58.22	55	16	0.016	0.015

7 Conclusions and Future Research

In this paper, we have proposed a novel classification method, BCEP, i.e., Bayesian Classification based on Emerging Patterns. BCEP is based on Bayesian theory to predict an unseen case given a training sample, which greatly improves the scoring function of the EP-based classifiers. It relaxes the strong attribute independence assumption implied by NB by using essential emerging patterns to compute probabilities. The method also utilizes the data class coverage to prune a lot of unnecessary EPs. BCEP uses only a small set of high quality EPs for classification, which is much less complex and much more understandable than previous EP-based classifiers. Our extensive experiments on 21 benchmark datasets from the UCI Machine Learning Repository show that BCEP has good overall predictive accuracy, and in many cases it is also superior to other state-of-the-art classification methods such as C5.0, NB(Naive Bayes), CAEP(Classification by Aggregating Emerging Patterns) and LB(Large Bayes, a recently proposed extension of NB using long itemsets).

An important issue is how to handle multi-class decision problems. As BCEP is only able to discriminate between two classes (recall EPs are defined upon the background and target class), it needs to turn multi-class problems into a set of binary problems. For a c -class problem, BCEP transforms it into c two-class problems, which are constructed by regarding class i as the background class and class j ($j = 1 \dots c, j \neq i$) as the target class. Recently, round robin classification (Furnkranz 2002) has been shown as a general ensemble technique to improve classification accuracy. The round robin binarization transforms a c -class problem into $c(c-1)/2$ two-class problems, one for each pair of classes. It is also of interest to investigate the performance of the round robin BCEP in future research.

References

- Bailey, J., Manoukian, T. & Ramamohanarao, K. (2002), Fast algorithms for mining emerging patterns, in 'Proc. 6th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'02)', Helsinki, Finland.
- Blake, C. & Merz, C. (1998), 'UCI repository of machine learning databases'.
- Cheeseman, P. & Stutz, J. (1996), Bayesian classification (autoclass): Theory and results, in 'Advances in Knowledge Discovery and Data Mining', AAAI/MIT Press, pp. 153–180.
- Chen, M., Han, J. & Yu, P. (1996), 'Data mining: an overview from a database perspective', *IEEE Trans. Knowledge and Data Engineering* **8**, 866–883.
- Domingos, P. & Pazzani, M. (1996), Beyond independence: Conditions for the optimality of the simple bayesian classifier, in 'Proc. of the Intl. Conf. on Machine Learning'.
- Domingos, P. & Pazzani, M. J. (1997), 'On the optimality of the simple bayesian classifier under zero-one loss', *Machine Learning* **29**(2-3), 103–130.
- Dong, G. & Li, J. (1999), Efficient mining of emerging patterns: Discovering trends and differences, in 'Proc. 1999 Intl. Conf. Knowledge Discovery and Data Mining (KDD'99)', San Diego, CA, USA, pp. 43–52.
- Dong, G., Zhang, X., Wong, L. & Li, J. (1999), Classification by aggregating emerging patterns, in 'Proc. the 2nd Intl. Conf. on Discovery Science (DS'99)', Tokyo, Japan, pp. 30–42.
- Duda, R. & Hart, P. (1973), *Pattern classification and scene analysis*, John Wiley & Sons, New York.
- Fan, H. & Ramamohanarao, K. (2002), An efficient single-scan algorithm for mining essential jumping emerging patterns for classification, in 'Proc. 2002 Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD'02)', Taipei, Taiwan.
- Fayyad, U. M. & Irani, K. B. (1993), Multi-interval discretization of continuous-valued attributes for classification learning, in 'Proc. the 13 International Joint Conference on Artificial Intelligence (IJCAI)', San Francisco, CA, pp. 1022–1029.
- Fayyad, U., Piatetsky-Shapiro, G. & Smyth, P. (1996), 'From data mining to knowledge discovery in databases', *AI Magazine* **17**, 37–54.
- Furnkranz, J. (2002), 'Round robin classification', *Journal of Machine Learning Research* **2**, 721–747.
- Kohavi, R. (1996), Scaling up the accuracy of Naive-Bayes classifiers: A decision-tree hybrid, in 'Proc. the 2nd Intl. Conf. on Knowledge Discovery and Data Mining (KDD-96)', Portland, Oregon, USA, pp. 202–207.
- Kohavi, R., John, G., Long, R., Manley, D. & Pflieger, K. (1994), 'MLC++: a machine learning library in C++', *Tools with artificial intelligence* pp. 740–743.
- Kohavi, R., Sommerfield, G. & Dougherty, J. (1997), 'Data mining using MLC++, a machine learning library in C++', *International Journal on Artificial Intelligence Tools* **6**(4), 537–566.
- Kononenko, I. (1991), Semi-naive bayesian classifier, in 'Proc of 6th European Working Session on Learning', Springer-Verlag, pp. 206–219.
- Lewis, P. (1959), 'Approximating probability distributions to reduce storage requirements', *Information and Control* **2**, 214–225.
- Li, J. (2000), Mining emerging patterns to construct accurate and efficient classifiers, PhD thesis, Melbourne University.
- Li, J., Dong, G. & K.Ramamohanarao (2000), Making use of the most expressive jumping emerging patterns for classification, in 'Proc. 2000 Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD'00)', pp. 220–232.
- Meretakos, D. & Wuthrich, B. (1999), Extending naive bayes classifiers using long itemsets, in 'Proc. the 5th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining, (KDD-99)', pp. 165–174.
- Quinlan, J. (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA.
- Rulequest (2000), 'See5/c5.0'. RULEQUEST RESEARCH [<http://www.rulequest.com/>].
- Tham, S., Doucet, A. & Ramamohanarao, K. (2002), Sparse bayesian learning for regression and classification using markov chain monte carlo, in 'Proc. 19th Intl. Conf. on Machine Learning', Sydney, Australia.

- Webb, G. & Pazzani, M. (1998), Adjusted probability naive bayesian induction, *in* 'Proc. 10th Australian Joint Conf. on Artificial Intelligence', Brisbane, Australia, pp. 285–295.
- Zhang, H., Ling, C. X. & Zhao, Z. (2000), The learnability of naive bayes, *in* 'Proc. of Canadian Artificial Intelligence Conference', pp. 432–441.
- Zhang, X. (2001), Emerging Patterns: Efficient Constraint-based Mining and the Aggregating Approach for Classification, PhD thesis, Melbourne University.