Efficient Mining of Contrast Patterns and Their Applications to Classification

Kotagiri Ramamohanarao, James Bailey and Hongjian Fan

***Abstract*— Data mining is one of the most important areas in the 21 century with many wide ranging applications. These include medicine, finance, commerce and engineering. Pattern mining is amongst the most important and challenging techniques employed in data mining. Patterns are collections of items which satisfy certain properties. Emerging Patterns are those whose frequencies change significantly from one dataset to another. They represent strong contrast knowledge and have been shown to be very successful for constructing accurate and robust classifiers. In this paper, we examine various kinds of contrast patterns. We also investigate efficient pattern mining techniques and discuss how to exploit patterns to construct effective classifiers.**

I. Introduction

Data mining is one of the most important areas in the twenty-first century – the information age. Its applications are wide ranging, including medicine, finance, commerce and engineering, to name a few. Data mining aims to discover interesting or useful patterns and relationships in a large volume of data. Major tasks in data mining include concept description, association, classification, prediction, clustering, evolution analysis and outlier analysis. Among these, classification is one of the fundamental and most important task and is our focus in this paper. Classification has been studied extensively in statistics, machine learning, neural networks and expert systems for many decades [32], [48]. The input is a set of *training records* (*training instances*), where each record has several attributes. Attributes with discrete domains are referred to as *categorical*, while those with continuous domains are referred to as *numerical*. There is one distinguished attribute called the *class label*. In general, given a database of records, each with a class label, a classifier generates a concise and meaningful description (or model) for each class in terms of the attributes. The model is then used to predict class labels of unknown objects. Classification is also known as *supervised learning*, as the learning of the model is "supervised", that is, each training instance is labelled indicating its class. In contrast to supervised learning, there is *unsupervised learning* (sometimes called *clustering*), which seeks to identify homogeneous groups of objects based on the values of their attributes, where no class labels denoting an *a priori* partition of the objects are given. Classification has been successfully applied to a wide range of application areas, such as scientific experiments, medical diagnosis, weather prediction, credit approval, customer segmentation, target marketing, and fraud detection [13], [27].

Classification based on patterns is a relatively new methodology. "A *Pattern* is an expression in some language describing a subset of the data" [49]. Let an item refer to an attribute-value pair. A set of items (called an itemset) is a conjunction of attribute values. An itemset is also called a pattern[1]. The recently proposed Emerging Pattern (EP) is a new type of knowledge pattern that describes significant changes (differences or trends) between two classes of data [20]. Emerging Patterns are sets of items whose frequency changes significantly from one dataset to another. Like other patterns or rules composed of conjunctive combinations of elements, Emerging Patterns can be easily understood and used directly by people. EPs have been successfully used for predicting the likelihood of diseases such as acute lymphoblastic leukemia [40] and discovering knowledge in gene expression data [41], [45].

Example 1 Table I shows a small, hypothetical dataset taken from [45] containing gene expression data, which records expression levels of genes under specific experimental conditions. There are 6 tissues samples in total: 3 normal and 3 cancerous tissues. Each tissue sample is described by the 4 gene expressions (namely, gene_1, gene_2, gene_3 and gene_4).

TABLE I
A simple gene expression dataset

ID	Cell type	gene_1	gene_2	gene_3	gene_4
1	Normal	0.10	1.20	-0.70	3.25
2	Normal	0.20	1.10	-0.83	4.37
3	Normal	0.60	1.30	-0.75	5.21
4	Cancerous	0.40	1.40	-1.21	0.41
5	Cancerous	0.50	1.10	-0.78	0.75
6	Cancerous	0.30	1.00	-0.32	0.82

We call gene_j@$[l, r]$ an *item*, meaning the values of expression of gene_j is limited inclusively between l and r. Inspecting Table I, we find the following interesting patterns.

- The pattern {gene_1@[0.3, 0.5], gene_4@[0.41, 0.82]} has a frequency of 0% in the sub-dataset with normal cells but 100% with cancerous cells.
- The pattern {gene_2@[1.1, 1.3], gene_3@[-0.83, -0.7]} appears three times in the sub-dataset with normal cells but only once with cancerous cells.

These patterns represent a group of gene expressions that have certain ranges of expression levels *frequently* in one type of tissue but *less frequently* in another. Therefore, they

Kotagiri Ramamohanarao, James Bailey and Hongjian Fan are with the Department of Computer Science and Software Engineering, University of Melbourne, Victoria, AUSTRALIA 3010. E-mail: {rao, jbailey, hfan}@csse.unimelb.edu.au

[1]In this paper, we use "pattern" and "itemset" interchangeably.

are excellent discriminators to distinguish the normal and cancer cells. Emerging Patterns are not only useful for medical doctors to gain a deeper understanding of the problem, but also for making reliable decisions on whether a patient has cancer. For example, based on Emerging Patterns, work in [45] proposes a personalized therapy which tries to modify the expression of some specific genes to convert cancer cells into normal ones. ■

The remaining of this paper is organized as follows. We first in Section 2 present preliminaries of patterns. We then in Section 3 survey a number of algorithms for efficient pattern discovery. We next in Section 4 investigate how to use patterns to build highly accurate classifiers. We conclude by discussing related work.

II. Patterns

A. Notation

To help formally define Emerging Patterns, we first give some preliminary definitions. Suppose that a dataset $\mathcal{D}$ is defined upon a set of attributes $\{A_1, A_2, \cdots, A_n\}$. For each attribute A_i, there is a set of permitted values, called the domain of that attribute, denoted as $domain(A_i)$. Attributes can be either categorical or continuous. For a continuous attribute, we assume that its value range is discretized[2] into intervals. For example, attribute *sex* is categorical and $domain(sex) = \{male, female\}$; attribute *age* is continuous, $domain(age) = [0, 150]$, and it can be discretized into intervals [0, 18], [18, 60] and [60, 150]. After discretization, $domain(age) = \{[0, 18], [18, 60], [60, 150]\}$. We call each (attribute, categorical-value) or (attribute, continuous-interval) pair an *item*. $(sex, male)$ and $(age, [18, 60])$ are two examples of items. By aggregating all the domain categorical-values and continuous-intervals across all attributes, we obtain the set of all items in $\mathcal{D}$, denoted as I, where $I = \{domain(A_1) \cup domain(A_2) \cup \cdots \cup domain(A_n)\}$. For convenience, we map all items from I including (attribute, categorical-value) and (attribute, continuous-interval) pairs to consecutive positive integers, i.e., we use 1 to represent the first item in I, 2 to the second item, and so on. By doing this, the original dataset can be treated as a transaction database.

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 $\mathcal{D}$, $supp_D(X)$, is $count_D(X)/|\mathcal{D}|$, where $count_D(X)$ is the number of instances in $\mathcal{D}$ containing X.

A pattern is *frequent* is its support is no less than a predefined *minimum support threshold* ξ. A *maximal frequent pattern* is a frequent pattern such that no proper super set is also frequent.

A pattern is *infrequent* is its support is less than a predefined *minimum support threshold* ξ. A *minimal infrequent pattern* is a frequent pattern such that no proper subset is also infrequent.

An association rule is an implication of the form $X \Rightarrow Y$, where X and Y are itemsets and $X \cap Y = \emptyset$. The left-hand side (LHS) X is called the antecedent of the rule, and the right-hand side (RHS) Y is called the consequent of the rule. The support of the rule in a transaction database TDB is $sup(X \cup Y)$; the confidence of the rule in TDB is $\frac{sup(X \cup Y)}{sup(X)}$.

[2]Our method of choice for discretization is the entropy technique from [36].

B. Emerging Patterns

Unlike frequent patterns, Emerging Patterns are concerned with two classes of data. We first define the growth rate of an itemset with respect to both classes.

Definition 1 Given two different classes of datasets $\mathcal{D}_1$ and $\mathcal{D}_2$, the **growth rate** of an itemset X from $\mathcal{D}_1$ to $\mathcal{D}_2$ is defined as $GrowthRate(X) = GR(X) =$

$$\begin{cases} 0 & \text{if } supp_1(X) = 0 \text{ and } supp_2(X) = 0 \\ \infty & \text{if } supp_1(X) = 0 \text{ and } supp_2(X) > 0 \\ \frac{supp_2(X)}{supp_1(X)} & \text{otherwise} \end{cases}$$

Emerging Patterns are those itemsets with large growth rates from $\mathcal{D}_1$ to $\mathcal{D}_2$.

Definition 2 Given a growth rate threshold $\rho > 1$, an itemset X is said to be a ρ-**Emerging Pattern** (ρ-**EP** or simply **EP**) from a background dataset D_1 to a target dataset D_2 if GrowthRate(X)$\geq \rho$.

When $\mathcal{D}_1$ is clear from context, an EP X from $\mathcal{D}_1$ to $\mathcal{D}_2$ is simply called an EP of $\mathcal{D}_2$, and $supp_{\mathcal{D}_2}(X)$ is called the support of the EP.

Specifically, a Jumping Emerging Pattern (JEP) is an EP with infinite growth rate (i.e., it is present in one class and absent in the other).

An EP with high support in its home class and low support in the contrasting class can be seen as a strong signal indicating the class of a test instance containing it. The strength of such a signal is expressed by its supports in both classes and its growth rate.

Definition 3 The **strength** of an EP X is defined as $strength(X) =$

$$\frac{GR(X)}{GR(X)+1} * supp_2(X) = \frac{supp_2(X) * supp_2(X)}{supp_2(X) + supp_1(X)}.$$

Note that when $GR(X) = \infty$, $strength(X)$ will be simply equal to $supp_2(X)$.

Generally, there can be a huge number of EPs. Therefore, a number of interestingness measures are defined to reduce the number of discovered EPs while not sacrificing their overall impact. A Chi Emerging Pattern is one such example.

Definition 4 An itemset X is called an *Chi Emerging Pattern* (Chi EP), if all the following conditions are true:

1) $supp(X) \geq \xi$, where ξ is a minimum support threshold;
2) $GR(X) \geq \rho$, where ρ is a minimum growth rate threshold;
3) $\neg\exists Y (Y \subset X) \wedge (supp(Y) \geq \xi) \wedge (GR(Y) \geq \rho) \wedge (strength(Y) \geq strength(X))$;

4) $|X| = 1 \vee |X| > 1 \wedge (\forall Y(Y \subset X \wedge |Y| = |X| - 1) \Rightarrow chi(X, Y) \geq \eta)$, where $\eta = 3.84$ is a minimum chi-value threshold and $chi(X, Y)$ is computed using chi-squared test [10].

An EP is *minimal* if no proper subset is also an EP; An EP is *maximal* if no proper super set is also an EP. Strong Jumping Emerging Patterns (SJEPs) [22], [26] are defined as minimal JEPs that satisfy a minimum support threshold. The support constraint of SJEP removes potentially less useful JEPs while retaining those with high discriminating power.

Other variety of EPs can also be defined, by incorporation of appropriate constraints, such as Constrained Emerging Patterns [6], Maximal Emerging Patterns [56], Generalized Noise-tolerant Emerging Patterns [26].

III. Mining Issues

The task of mining Emerging Patterns is computationally expensive for large, dense and high-dimensional datasets, because the number of patterns present can be exponential in the number of attributes in the worst case. What is worse, the Apriori anti-monotone property – every subset of a frequent pattern must be frequent, or in other words, any superset of an infrequent item set cannot be frequent – which would be very effective for pruning the search space, does not apply to mining Emerging Patterns. The reason is as follows. Suppose a pattern X with k items is not an EP. This means its growth rate – the support ratio between two data classes – does not satisfy the growth-rate threshold. Consider Y, a super-pattern of X with $(k+1)$ or more items. Y will usually have decreased support in both classes, but its growth rate (the support ratio) is free to be any real value between 0 and ∞. So a superset of a non-EP may or may not be an EP.

In the border-based approach [20], borders are used to represent candidates and subsets of EPs; the border differential operation is used to discover EPs. Given a minimum growth rate threshold $\rho > 1$, suppose we want to find EPs from $\mathcal{D}_1$ to $\mathcal{D}_2$ with growth rates more than ρ. We first fix the minimum support threshold ξ_1 for $\mathcal{D}_1$, and use a border-discovery algorithm such as Max-Miner [9] to find the border for $\mathcal{D}_1$. We then obtain the minimum support threshold ξ_2 for $\mathcal{D}_2$, where $\xi_2 = \xi_1 \times \rho$; we again use the border-discovery algorithm to find the border for $\mathcal{D}_2$, which essentially consists of the maximal frequent patterns with respect to ξ_2. After both borders for $\mathcal{D}_1$ and $\mathcal{D}_2$ are available, the border of Emerging Patterns can be efficiently derived by the border differential procedure.

ConsEPMiner [57] follows level-wise, candidate generation-and-test approach and mines EPs satisfying several constraints including the growth-rate improvement constraint.

A. Algorithms Based on the Contrast Pattern Tree Structure

Inspired by the FP-tree [33], the CP-tree data structure is used for EP mining for the first time [22]. A CP-tree registers the counts in both the positive and negative class. An example CP-tree is illustrated in Figure 1. Because every training instance is sorted by its support ratio (the order is denoted as $\prec$) between both classes when inserting into the CP-tree, items with high ratio, which are more likely to appear in an SJEP, are closer to the root. The map from a path in the CP-tree to an itemset is a one-to-one mapping. Using the predefined order $\prec$, we can produce the complete set of paths (itemsets) systematically through depth-first searches of the CP-tree. Unlike the FP-growth algorithm that performs frequent pattern mining from leaf to root and must create many conditional FP-trees during the process, the CP-tree based algorithm searches the CP-tree depth first from root and performs a powerful technique, node merge, along with the search. The CP-tree based algorithm can discover SJEPs of both $\mathcal{D}_1$ and $\mathcal{D}_2$ from the CP-tree at the same time - a "single-scan" algorithm. Previous EP mining methods such as the border-based algorithms [20] and consEPMiner [57] have to call the corresponding algorithm twice using $\mathcal{D}_1$ and $\mathcal{D}_2$ as target dataset separately. Unlike those two approaches, we do not need to construct one CP-tree for mining SJEPs of $\mathcal{D}_1$, and construct another CP-tree for mining SJEPs of $\mathcal{D}_2$.

In [5], a fast algorithm for mining Jumping Emerging Patterns (JEPs) is proposed, which is typically 5-10 times faster than the earlier border-based approach. The algorithm constructs tree structures to target the likely distribution of JEPs.

TABLE II
An example dataset with two classes

ID	Class Label	Instances (Itemsets)	itemsets ordered by $\prec$
1	$\mathcal{D}_1$	{a,c,d,e}	[e,a,c,d]
2	$\mathcal{D}_1$	{a}	[a]
3	$\mathcal{D}_1$	{b,e}	[e,b]
4	$\mathcal{D}_1$	{b,c,d,e}	[e,b,c,d]
5	$\mathcal{D}_2$	{a,b}	[a,b]
6	$\mathcal{D}_2$	{c,e}	[e,c]
7	$\mathcal{D}_2$	{a,b,c,d}	[a,b,c,d]
8	$\mathcal{D}_2$	{d,e}	[e,d]

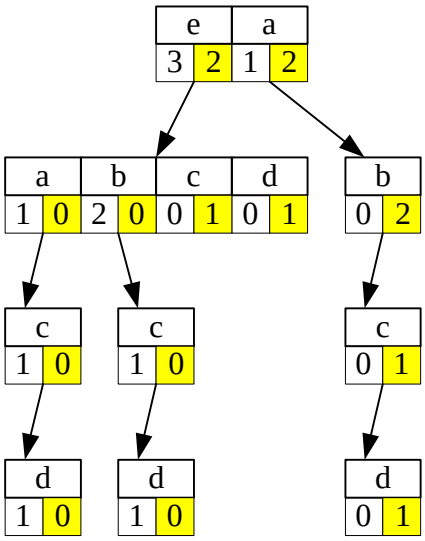

Fig. 1. The original CP-tree of the example dataset

B. Computing Hypergraph Transversals

Analysis of hypergraphs is a well established field of discrete mathematics which has many applications in computer science, ranging from minimal diagnosis and propositional circumscription, to learning boolean formulae and boolean switching theory.

Hypergraphs are defined by a set of vertices $V = \{v_1, v_2, \cdots, v_n\}$ and a set of edges E, where each edge is some subset of V. A *transveral* of a hypergraph is any set of vertices that contains at least one element of every edge. A *minimal transversal* is a transveral such that no proper subset is also a transversal.

The hypergraph minimal transversal problem is particularly significant from a data mining perspective. Indeed, the algorithmic complexity of mining maximal frequent pattern and minimal infrequent pattern is closely linked to the complexity of computing minimal hypergraph transversals [29]. The hypergraph minimal transversal problem has close connection to the mining of Emerging Patterns. This is further highlighted in [7].

A new algorithm for computing hypergraph transversals is developed in [7]. It is based on the idea of paritioning, which is fundamentally designed to prevent situations in which prohibitively large (with respect to cardinality) edges have their cross-product computed. The algorithm uses a divide and conquer approach to iterate through specific subsets of edges. Candidate subsets are identified such that cross-product expansion/minimization is more tightly controlled. The process of partitioning is recursive. Experiments on a number of large datasets show that the algorithm outperforms previous approaches (such as Berge's algorithm) by a factor of 9-29 times.

C. Incremental Mining Techniques

The space of JEPs is convex, so it is possible to incrementally modify and maintain the concise boundary descriptions of the JEP space when small changes occur to the data. An incremental algorithm is proposed in [42], [43] to handle four types of changes: insertion of new data, deletion of old data, addition of new attributions and deletion of old attributes. The algorithm computes the borders of the new EP space by using the boundary patterns from the old space, instead of recomputing from scratch. Experimental results show that it is much faster than the From-Scratch method. Its high efficiency is largely due to the border operations that manipulate only the boundary patterns but do not need to enumerate all contained EPs.

D. Contrasts for Sequence Data

Contrast patterns can also be developed for data formats that are more complex than relational tables. In [34], techniques for the discovery of minimal distinguishing subsequences are presented. A distinguishing subsequence is a subsequence that appears frequently in one class of sequences, yet infrequently in another. A distinguishing subsequence is minimal if none of its subsequences is distinguishing. A key property of a minimal distinguishing subsequence is that its items do not have to appear consecutively, there may be gaps between them. Such sequences are useful in the comparison of sequence data, such as DNA, proteins or text.

Mining minimal distinguishing subsequence patterns is a challenging task and is significantly different from mining contrasts between relational data. The order in which items occur is important and items may occur multiple times. Another challenge which arises is due to consideration of gap constraints. Items in a minimal distinguishing subsequence do not have to appear immediately next to each other in the original sequences. However, subsequences in which items are far away from each other are less likely to be meaningful than those whose items are close in the original sequence. It is necessary, therefore, to set a maximum gap constraint during mining. This restricts the distance between neighboring elements of the subsequence. Additional benefits are that the mining output is smaller and the mining process can be faster. Work in [34] describes the ConSGapMiner algorithm, which is able to mine all minimal distinguishing subsequences according to a maximum gap constraint. It employs highly efficient bitset and boolean operations for powerful gap based pruning within a prefix growth framework. A performance evaluation shows that it is able to mine patterns from high dimensional datasets at low supports.

IV. Classification

In this section we investigate how to use a set of Emerging Patterns for classification.

A. A General Framework of EP-based Classifiers

Algorithm 1: Training the EP-based Classifier

input : a set of training instances D (contaning n classes of data, i.e. $D = D_1 + D_2 + \cdots + D_n$)
output: the EP-based classifier
foreach $1 \leq i \leq n$ **do**
  mine the set E_i of EPs from $(\bigcup_{j=1}^{n} D_j - D_i)$ to D_i
end
post-process the discovered EPs;
output the classification model made up of EPs;

Algorithm 2: Classification by EP-based Classifier

input : the EP-based classifier and a testing instance T
output: the classification for T
foreach *class i* **do**
  compute a score $S(i,t)$ based on `Scoring-Function` using EPs' supports and growth rates
end
assign the class with the highest score to T;

Since an EP has high support in its home class and low support in the contrasting class, it can be seen as a strong signal indicating the class of a test instance containing it.

We can use EPs as an effective means for classification. Examples of EP-based classifiers include CAEP [21] and the JEP-Classifier [38].

The training stage of the EP-based classifier is shown in Algorithm 1. EP-based classifier can handle datasets containing more than two classes by partitioning the training datasets according to the class label. The training phase consists of discovery of EPs and post-processing of these EPs. EP-discovery algorithms include border-based approach [20] and recent tree-based methods [24]. A classification model is built from the training data, which is represented by n sets of EPs, one set per class.

For a testing instance T, we derive n scores for it, one score per class, by feeding the EPs of each class into a scoring function, as shown in Algorithm 2. Usually the scoring function is carefully designed to make good use of the characteristics of different kinds of EPs. The class with the highest score is assigned to T as its class label. Ties are broken in favor of the class with more instances in the training dataset. When the scores are exactly same or very close (i.e., the absolute difference is less than a given threshold), it will predict the majority class.

B. Aggregation of Support

It is natural to aggregate the discriminating power of those EPs that are contained in a test instance to determine the class label of the test.

CAEP (**C**lassification by **A**ggregating **E**merging **P**atterns) is the first application of EPs for classification. It uses the following scoring function. Usually, the EPs used in CAEP satisfy a minimum support of 1% and a minimum growth rate of 5.

Definition 5 Given a test instance T and a set $E(C_i)$ of EPs of data class C_i, the **aggregate score (or score)** of T for the class C_i is defined as $score(T, C_i) =$

$$\sum_{X \subseteq T, X \in E(C_i)} \frac{growth\_rate(X)}{growth\_rate(X)+1} * supp_{C_i}(X),$$

where $supp_{C_i}(X)$ is the support of X in class C_i, and $growth\_rate(X)$ is $supp_{C_i}(X)$ divided by the X's support in non-C_i class.

The JEP-Classifier uses only JEPs, that is, EPs with infinite growth rates. Therefore, $score(T, C_i)$ becomes $\sum_{X \subseteq T, X \in E(C_i)} supp_{C_i}(X)$.

C. BCEP: a Bayesian Approach to Use Emerging Patterns for Classification

It is relatively easy to implement the scoring function using simple aggregation of EPs. However, this method of score calculation does not have any solid statistical foundation. Work in [23] proposes a classifier called "**B**ayesian **C**lassification by **E**merging **P**atterns" (BCEP), whose scoring function is based on Bayes theorem. BCEP is a hybrid of the EP-based classifier and Naive Bayes (NB) classifier and it provides several advantages. First, it is based on theoretically well-founded mathematical models (Bayes theorem) 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. The detail of the BCEP classifier can be found in [23]. An empirical study carried out on benchmark datasets 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.

D. Using EPs for Rare-class Classification

The classification of rarely occurring cases is a challenging problem in many real life applications, such as the identification of responders in large surveys or marketing campaigns and the identification of intruders in security networking systems. It implies processing an imbalanced data set to distinguish rarely-occurring cases from other overwhelming cases. In the case of network security, the number of intrusion sessions (e.g., by guessing passwords) can be very low (5% – 10%) compared to the overall number of legal sessions. The scarcity of the rare cases makes it difficult for traditional classifiers to classify them correctly. For example, although popular and powerful, C4.5 decision tree fails to perform well in rare-class classification because the scarcity of the rare class biases most of the decisions in the tree toward the major class.

Work in [2] proposes to use EPs in rare-class classification for the first time. EPs are improved through three stages: (1) generating new undiscovered EPs for the rare class; (2) pruning bad EPs; and (3) increasing the supports of the rare-class EPs. In the first stage, the new EPs are generated by replacing different attribute values (in the original rare-class EPs) with the highest-growth-rate values. In the second stage, bad EPs are pruned for both the major and rare classes, where bad EPs refer to those whose average growth rates are less than the given threshold. At the last step, we increase the support of rare-class EPs by a given percentage, in order to compensate the effect of the large number of major-class EPs (i.e., overwhelming major-class EPs make many rare-class instances classified as major class). The new approach outperforms many other classifiers including PNrule [35], which is reported to be the best rare-class classifier.

Later work in [3] combines EPs and decision trees in rare-class classification. It employs the power of EPs to improve the quality of rare-class classification in two ways: (1) generating new non-existing rare-class instances; and (2) over-sampling the most important rare-class instances. In the first approach, the new rare-class instances are created by combining EPs. If a value for an attribute is missing, it is substituted by the value that has the highest growth rate for the same attribute (which is the best single-attribute itemset that represents the rare class). The newly generated rare-class instances are very similar to the original rare-class instances and can support the classification of rare cases effectively. In the second approach, we over-sample the most important rare-class instances only, rather than over-sampling randomly, since it may affect the

performance negatively due to the possibility of noise amplification. The most important instances are those that contain rare-class EPs – they have the most important information that can aid the classification process and do not introduce noise in the space of rare-class instances. The experimental evaluation shows this method improves the previous one further and outperforms other classifiers such as PNrule, C4.5, Metacost and over-sampling.

E. DeEPs: an Instance-based Lazy Discovery and Classification System

Lazy learning [1] is sometimes called instance-based learning. In contrast to lazy learning is eager learning, where a model (a group of rules or a set of patterns) is first induced from the training data, and then it is repeatedly applied to new instances for classification. The EP-based classifiers discussed so far are all eager. In lazy learning, we use the new instance as a constraint to extract knowledge useful only for the classification of this instance. A typical lazy classifier is k-nearest neighbor (k-NN) [18]. The intuition behind it is that the class of a test instance is most likely to be the majority class among the k nearest training instances to the test in a certain distance measure.

DeEPs is an instance-based lazy discovery and classification system [37], [39], [44]. Its fundamental idea is an effective data reduction technique. When classifying a new instance, DeEPs uses that instance as a filter to remove irrelevant training values, making the original training data sparse in terms of both dimensionality (number of attributes) and volume (number of instances). The reduced training instances are further compressed along the volume direction by selecting only those maximal ones. DeEPs can handle continuous valued attributes very well. It targets discretization towards the test instance using the technique of neighborhood-based intersection. Unlike k-NN which only locates "raw" training instances without extraction of high level patterns, DeEPs can provide comparative knowledge patterns and rules for people to understand a new instance. DeEPs is accurate and reliable: its performance is comparable to and often better than other instance-based classifiers such as k-NN. Due to the lazy-learning style, DeEPs is very useful for practical applications where the data is frequently updated.

F. A Weighting Scheme Based on Emerging Patterns for Weighted Support Vector Machines

Support Vector Machines (SVMs) were introduced in the early 1990s and the topic on the entire family of kernel-based learning methods has developed into a very active field of Machine Learning research [15], [55]. SVMs are based on statistical learning theory developed by Vapnik and their formulations embody the structural risk minimization (SRM) principle. Due to their good generalization ability for a wide range of applications, SVMs have been powerful tools for solving classification problems, delivering state of the art performance in applications from analysis of DNA microarray data to text categorization, from handwritten digits recognition to protein homology detection.

SVMs combine two key ideas. The first is the concept of an *optimum margin classifier*. An optimum margin classifier is a linear classifier; it constructs a separating hyperplane which maximizes the distance to the training points. The important point is maximization of the margin, which turns the under-specified learning problem into an optimization problem (without local optima) and has been shown to give very good generalization performance. In general, the optimum margin hyperplane will be a linear combination of the input vectors; *support vectors* are those training instances which obtain a non-zero coefficient, i.e., the ones that lie closest to the separating hyperplane. The second key concept underlying SVMs is a *kernel*. In its simplest form, a kernel is a function which calculates the dot product of two training vectors. Intuitively, this dot product expresses the similarity of the two training instances in terms of the given attributes. If we use feature transformation techniques to reformulate the input vectors in terms of new features and find a way to calculate dot products in this feature space, we can leave the linear classifier unaffected. Generally, the kernel can be thought of as a non-linear similarity measure and kernel functions are inner products in some feature space (potentially very complex).

Because the optimal hyperplane obtained by a SVM depends on only a small part of the data points (support vectors), it may become sensitive to noise or outliers in the training set. To address this problem, Weighted Support Vector Machines (weighted SVMs) are proposed in [46], which associate a weight with each data point such that different points can have different impacts on the learning of the optimal separating hyperplane. Weighted SVMs try to maximize the margin like the classical SVMs, but use weights to prevent some points (i.e., noise or outliers) from making narrower margin. If the data points are already associated with the weights, it is straightforward to use this information to train weighted SVMs. If a noise distribution model of the data is given, we can set the weight as the probability of the point that is not a noise, or as a function of it. However, almost all real world applications lack the weight information and it is very hard to know the true noise distribution. It is an open issue how to generate a good and reliable weighting model from data without any domain knowledge.

Work in [25] uses EPs to construct a reliable and robust weighting model to reflect the true noise distribution in the training data, i.e., noise and outliers should have low weights. The intuition behind the idea is that a representative instance of a class should contain strong EPs of the same class, while noise and outliers should contain no EPs or EPs of contradicting classes. The overall procedure of the EP-based weighting model is shown in Figure 2. The experimental results show that the EP-based weighting scheme often improves the performance of weighted Support Vector Machines and is consistently better than SVM. This demonstrates that our EP-based weighting model has the ability to reduce the effects of noise and outliers. Experiments also show that the EP-based scheme is often superior to the previous distance based weighting model. This supports the hypothesis that the EP model can deal with high dimensional data (where distance does not work) and cope with arbitrary class shape

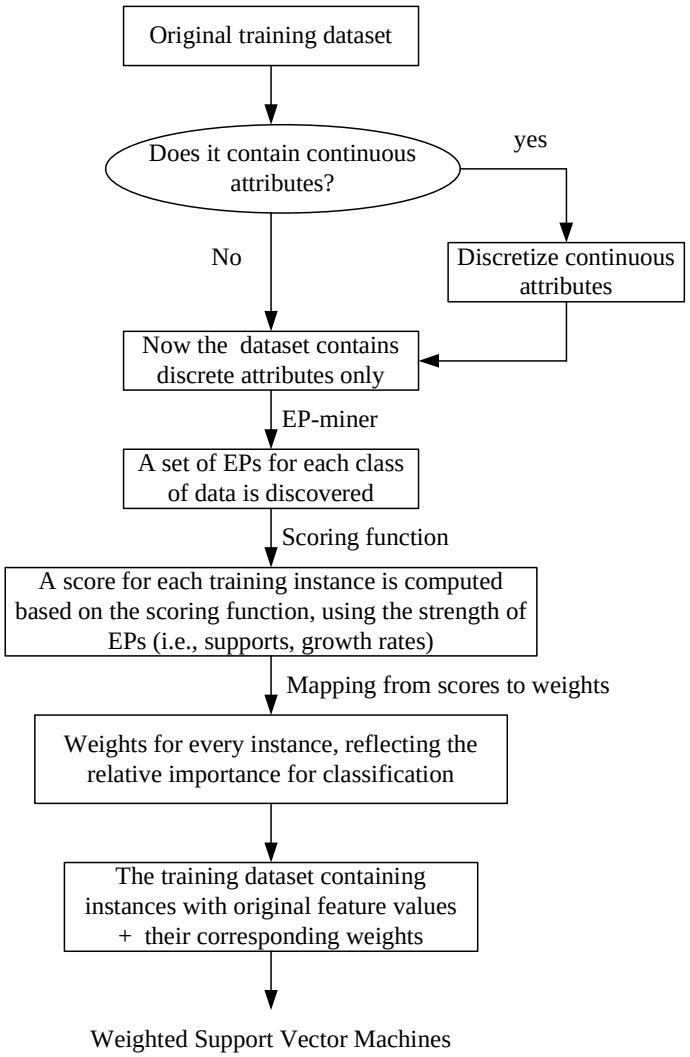

Fig. 2. Overall procedure of the EP-based weighting model

(no assumption about class centers).

G. Expanding the Training Data Space Using Emerging Patterns and Genetic Methods

The performance of traditional classifiers is proportional to the knowledge obtained from the training data. As a result, they cannot perform very well when the training data space is very limited (i.e., only a proportion of all the data points is available). Work in [4] combines the power of EPs (strong discriminating power) and genetic methods (great reproduction ability) to expand the training data space. Genetic methods incorporate evolutionary and computational techniques inspired by biology. The expansion process is performed by generating more training instances using four techniques: (1) generating by superimposing EPs; (2) generation by crossover; (3) generation by mutation; (4) generation by mutation and EPs. Technique (1) generates new instances by assembling attribute values from several EPs. Technique (2) switches the values of two instances before a randomly chosen breaking point. Technique (3) mutates an instance with the highest-growth-rate values. Techniques (4) mutates an instance with an EP, where all values in the instance are replaced with their matched values in the EP. A number of constraints are used to generate a reasonable number of high quality training instances. (1) Instead of using the whole set of EPs, a certain percentage of the best quality EPs is used; (2) Instead of using the whole set of training data, a certain percentage of the best quality instances is used; (3) We keep the best-quality generated instances while discarding those with bad quality; (4) We stop the generation process after a certain number of instances are generated. Note that the quality of EPs is measured by their growth rates, and the quality of a data instance is measured by the average support of the attribute values in the instance.

The generated instances are proved to contain necessary information that supports classification. The experimental evaluation shows that the method has a great impact on improving the results of other classification method such as C4.5, C4.5 Boosting and SVM.

H. Experimental Evaluation

TABLE III
ACCURACY COMPARISON. "CLEAN" MEANS NO NOISE ADDED. "ATTRIBUTE", "LABEL" AND "MIX" REFER TO 40% ATTRIBUTE NOISE, 40% LABEL NOISE AND 40% MIX (ATTRIBUTE + LABEL) NOISE, RESPECTIVELY

Noise type	NB	C4.5	SVM	BCEP
Clean	80.18	84.68	85.99	87.30
Attribute	76.94	78.42	76.71	82.29
Label	76.78	77.18	81.66	82.27
Mix	74.86	70.88	73.02	76.62
Average	77.19	77.79	79.35	82.12

We now provide a very small set of experimental results to demonstrate the superior power of EP-based classifiers. In table III, we compare the BCEP classifier against well-known classifiers such as Naive Bayes (NB), decision tree C4.5 and Support Vector Machines (SVM). Experiments are performed on a number of benchmark datasets from the UCI Machine Learning Repository [12]. We can see that the accuracy of BCEP is extremely competitive and it is also very robust in terms of noise. Detailed comparisons can be found in work [6], [21]–[23], [38], [39], [56].

V. RELATED WORK

Concepts related to Emerging Patterns include version spaces [47], disjunctive version spaces [54], discriminant rules [30], [31] and contrast sets [8].

Many classification models have been proposed in the literature: Neural networks [11], [52], genetic algorithms [28], Bayesian methods [16], log-linear models and other statistical methods [17], instance-based learning algorithms [1] (such as nearest neighbor classifiers described in [19]) and decision trees or classification trees [14], [50], [51], [53]. EP-based classifiers are very different from the above models. EP classifier models use a set of multi-attribute tests (EPs) for decision making, while decision trees consider only one test on one attribute at a time and Bayesian models uses probabilities.

Compared to a neural network and a SVM classifier that act as a black box, EP-based classifiers are relatively easy to understand, similar to decision trees.

VI. Conclusion

In this paper we have discussed several kinds of patterns (Emerging Patterns in particular), efficient pattern mining techniques and effective classifiers based on patterns.

Mining of interesting patterns is an important field of knowledge discovery and data mining. As we have seen in the last few years, classifiers built using Emerging Patterns are extremely reliable, accurate and noise tolerant. However, there are still interesting questions, such as finding more generalized patterns (e.g., contrast graph patterns and contrast sequence patterns) and defining richer language expressions for patterns (e.g., in addition to conjunctions of attribute values, disjunction and negation are used). Efficient mining of these complex patterns is also an open question. Already researcher are applying Emerging Patterns in the biological domain with substantial successes [40], [45]. We believe that Emerging Patterns can be applied in much wider domain of problems such as e-commerce, network intrusion detection, anti-spam Email, image data, semi-structured texts (e.g., XML documents) and unstructured texts (e.g., World Wide Web documents).

References

[1] D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] Hamad Alhammady and Kotagiri Ramamohanarao. The application of emerging patterns for improving the quality of rare-class classification. In *Proc. 8th Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD2004)*, pages 207–211, Sydney, Australia, 2004.

[3] Hamad Alhammady and Kotagiri Ramamohanarao. Using emerging patterns and decision trees in rare-class classification. In *Proc. 4th IEEE Int'l Conf. on Data Mining (ICDM 2004)*, pages 315–318, Brighton, UK, 2004. IEEE Computer Society.

[4] Hamad Alhammady and Kotagiri Ramamohanarao. Expanding the training data space using emerging patterns and genetic methods. In *Proc. 2005 SIAM International Data Mining Conference (SDM2005)*, 2005.

[5] James Bailey, Thomas Manoukian, and Kotagiri Ramamohanarao. Fast algorithms for mining emerging patterns. In *Proc. 6th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'02)*, Helsinki, Finland, 2002.

[6] James Bailey, Thomas Manoukian, and Kotagiri Ramamohanarao. Classification using constrained emerging patterns. In *Proc. 4th Int'l. Conf. on Web-Age Information Management (WAIM2003)*, pages 226–237, Chengdu, China, Aug 2003.

[7] James Bailey, Thomas Manoukian, and Kotagiri Ramamohanarao. A fast algorithm for computing hypergraph transversals and its application in mining emerging patterns. In *Proc. 3rd IEEE Int'l Conf. on Data Mining (ICDM2003)*, pages 485–488, Melbourne, Florida, USA, 2003.

[8] Stephen D. Bay and Michael J. Pazzani. Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery*, 5(3):213–246, 2001.

[9] Robert J. Bayardo Jr. Efficiently mining long patterns from databases. In *Proc. 1998 ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD'98)*, pages 85–93, Seattle, WA, USA, June 1998. ACM Press.

[10] Robert M. Bethea, Benjamin S. Duran, and Thomas L. Boullion. *Statistical methods for engineers and scientists*. New York : M. Dekker, 1995.

[11] Christopher M. Bishop and Chris Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[12] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998.

[13] Ronald J. Brachman, Tom Khabaza, Willi Kloesgen, Gregory Piatetsky-Shapiro, and Evangelos Simoudis. Mining business databases. *Communications of the ACM*, 39(11):42–48, 1996.

[14] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Chapman & Hall, New York, 1984.

[15] C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Knowledge Discovery and Data Mining*, 2(2):121–167, 1998.

[16] Peter Cheeseman and John Stutz. Bayesian classification (autoclass): Theory and results. In *Proc. 2nd Int'l Conf. on Knowledge Discovery and Data Mining (KDD-96)*, pages 153–180. AAAI/MIT Press, 1996.

[17] Ronald Christensen. *Log-Linear Models and Logistic Regression*. Springer, 1997.

[18] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. Information Theory*, 13:21–27, 1967.

[19] Belur V. Dasarathy. *Nearest neighbor norms: NN pattern classification techniques*. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[20] Guozhu Dong and Jinyan Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proc. 5th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD'99)*, pages 43–52, San Diego, CA, USA, Aug 1999.

[21] Guozhu Dong, Xiuzhen Zhang, Limsoon Wong, and Jinyan Li. Caep: Classification by aggregating emerging patterns. In *Proc. 2nd Int'l Conf. on Discovery Science (DS'99)*, pages 30–42, Tokyo, Japan, Dec 1999.

[22] Hongjian Fan and Kotagiri Ramamohanarao. An efficient single-scan algorithm for mining essential jumping emerging patterns for classification. In *Proc. 6th Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD2002)*, pages 456–462, Taipei, Taiwan, China, May 2002.

[23] Hongjian Fan and Kotagiri Ramamohanarao. A bayesian approach to use emerging patterns for classification. In *Proc. 14th Australasian Database Conference (ADC2003)*, pages 39–48, Adelaide, Australia, Feb 2003.

[24] Hongjian Fan and Kotagiri Ramamohanarao. Efficiently mining interesting emerging patterns. In *Proc. 4th Int'l. Conf. on Web-Age Information Management (WAIM2003)*, pages 189–201, Chengdu, China, Aug 2003.

[25] Hongjian Fan and Kotagiri Ramamohanarao. A weighting scheme based on emerging patterns for weighted support vector machines. In *Proc. IEEE Int'l Conf. on Granular Computing (GrC 2005)*, 2005.

[26] Hongjian Fan and Kotagiri Ramamohanarao. Fast discovery and the generalization of strong jumping emerging patterns for building compact and accurate classifiers. *IEEE Transactions on Knowledge and Data Engineering*, submitted 2005.

[27] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17:37–54, 1996.

[28] A. A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Spinger-Verlag, Berlin, 2002.

[29] Dimitrios Gunopulos, Heikki Mannila, Roni Khardon, and Hannu Toivonen. Data mining, hypergraph transversals, and machine learning. In *Proc. 16th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems (PODS'97)*, pages 209–216, 1997.

[30] Jiawei Han, Yandong Cai, and Nick Cercone. Knowledge discovery in databases: An attribute-oriented approach. In Li-Yan Yuan, editor, *Proc. 18th Int'l Conf. on Very Large Databases*, pages 547–559, San Francisco, U.S.A., 1992. Morgan Kaufmann Publishers.

[31] Jiawei Han, Yandong Cai, and Nick Cercone. Data-driven discovery of quantitative rules in relational databases. *IEEE Trans. Knowledge and Data Engineering*, 5(1):29–40, 1993.

[32] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2000.

[33] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proc. 2000 ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD'00)*, pages 1–12, Dallas, TX, USA, May 2000.

[34] Xiaonan Ji, James Bailey, and Guozhu Dong. Mining minimal distinguishing subsequence patterns with gap constraints. In *To appear in Proc. the 5th IEEE International Conference on Data Mining (ICDM)*, Houston, Texas, USA, 2005.

[35] Mahesh V. Joshi, Ramesh C. Agarwal, and Vipin Kumar. Mining needle in a haystack: classifying rare classes via two-phase rule induction. In *Proc. ACM Conf. on Management of Data (SIGMOD2001)*, pages 91–102, 2001.

[36] Ron Kohavi, George John, Richard Long, David Manley, and Karl Pfleger. MLC++: a machine learning library in C++. *Tools with artificial intelligence*, pages 740–743, 1994.

[37] Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. Instance-based classification by emerging patterns. In *Proc. 4th European ConferencePrinciples of Data Mining and Knowledge Discovery (PKDD 2000)*, pages 191–200, Lyon, France, 2000.

[38] Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. Making use of the most expressive jumping emerging patterns for classification. *Knowl. Inf. Syst.*, 3(2):131–145, 2001.

[39] Jinyan Li, Guozhu Dong, Kotagiri Ramamohanarao, and Limsoon Wong. Deeps: A new instance-based lazy discovery and classification system. *Machine Learning*, 54(2):99–124, 2004.

[40] Jinyan Li, Huiqing Liu, James R. Downing, Allen Eng-Juh Yeoh, and Limsoon Wong. Simple rules underlying gene expression profiles of more than six subtypes of acute lymphoblastic leukemia (all) patients. *Bioinformatics*, 19(1):71–78, 2003.

[41] Jinyan Li, Huiqing Liu, See-Kiong Ng, and Limsoon Wong. Discovery of significant rules for classifying cancer diagnosis data. *Bioinformatics*, 19(Suppl. 2):ii93–ii102, 2003.

[42] Jinyan Li, Thomas Manoukian, Guozhu Dong, and Kotagiri Ramamohanarao. Incremental maintenance on the border of the space of emerging patterns. *Data Min. Knowl. Discov.*, 9(1):89–116, 2004.

[43] Jinyan Li, Kotagiri Ramamohanarao, and Guozhu Dong. The space of jumping emerging patterns and its incremental maintenance algorithms. In *Proc. Seventeenth Int'l Conf. on Machine Learning (ICML 2000)*, pages 551–558, Stanford University, Standord, CA, USA, 2000. Morgan Kaufmann.

[44] Jinyan Li, Kotagiri Ramamohanarao, and Guozhu Dong. Combining the strength of pattern frequency and distance for classification. In *Proc. 5th Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'01)*, pages 455–466, Hong Kong, China, 2001.

[45] Jinyan Li and Limsoon Wong. Identifying good diagnostic gene groups from gene expression profiles using the concept of emerging patterns. *Bioinformatics*, 18(5):725–734, 2002.

[46] Chun-Fu Lin and Sheng-De Wang. Fuzzy support vector machines. *IEEE Transactions on Neural Networks*, 13(2):464 – 471, March 2002.

[47] Tom Mitchell. Generalization as search. *Artificial Intelligence*, 18(2), 1982.

[48] Tom Mitchell. *Machine Learning*. McGraw Hill, 1997.

[49] Gregory Piatetsky-Shapiro and William J. Frawley. *Knowledge Discovery in Databases*. AAAI/MIT Press, Cambridge, MA, 1991.

[50] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[51] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[52] B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[53] RuleQuest. See5/c5.0, 2000. RULEQUEST RESEARCH data mining tools [http://www.rulequest.com/].

[54] M. Sebag. Delaying the choice of bias: A disjunctive version space approach. In *Proc. 13th Int'l Conf. on Machine Learning*, pages 444–452. Morgan Kaufmann, 1996.

[55] Vladimir Vapnik. *Statistical Learning Theory*. John Wiley, NY, 1998.

[56] Zhou Wang, Hongjian Fan, and Kotagiri Ramamohanarao. Exploiting maximal emerging patterns for classification. In *Proc. 17th Australian Joint Conf. on Artificial Intelligence*, pages 1062–1068, Cairns, Queensland, Australia, Dec 2004.

[57] Xiuzhen Zhang, Guozhu Dong, and Kotagiri Ramamohanarao. Exploring constraints to efficiently mine emerging patterns from large high-dimensional datasets. In *Proc. 6th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD'00)*, pages 310–314, Boston, USA, Aug 2000.