Discovery of Emerging Patterns and Their Use in Classification

Kotagiri Ramamohanarao and James Bailey

Department of Computer Science and Software Engineering
University of Melbourne, Australia
{rao,jbailey}@cs.mu.oz.au

Abstract. Emerging patterns are sets of items whose frequency changes significantly from one dataset to another. They are useful as a means of discovering distinctions inherently present amongst a collection datasets and have been shown to be a powerful method for constructing accurate classifiers. In this paper, we present different varieties of emerging patterns, discuss efficient techniques for their discovery and explain how they can be used in classification.

1 Introduction

Discovery of powerful distinguishing features between datasets is an important objective in data mining. An important class of patterns that can represent strong contrasts is known as *emerging patterns*. An emerging pattern is a set of items whose frequency changes significantly from one dataset to another. Emerging patterns have been successfully used in constructing highly accurate classifiers [19, 10, 20]. In particular, for predicting the likelihood of diseases such as leukaemia [21] and discovering patterns in gene expression data [22].

As an example, consider the following database defined on six attributes: (with all possible attribute values given in parentheses) Attr1 (a, b, c), Attr2 (d, e, f), Attr3 (g, h, i), Attr4 (j, k, l), Attr5 (m, n, o) and Attr6 (p, q, r).

The minimal emerging patterns in this dataset are sets of items which appear in one class and not in the other (by minimal, we mean that no proper subset of the items in the pattern should also be an emerging pattern). Inspecting the table, we see that patterns appearing in Class 1 include $\{a, i\}$, $\{o, p\}$, and $\{b, g\}$. Patterns appearing in Class 2 include $\{f\}$ and $\{k\}$. These patterns represent very strong changes (zero frequency in one dataset and non-zero frequency in

Table 1. Sample Database

Class 1 Instances	Class 2 Instances
$\{a, d, i, l, o, p\}$	$\{c, d, i, l, o, r\}$
$\{b, d, g, l, o, r\}$	$\{a, f, g, k, o, q\}$
$\{c, d, h, l, o, r\}$	$\{b, d, h, j, m, p\}$

T.D. Gedeon and L.C.C. Fung (Eds.): AI 2003, LNAI 2903, pp. 1–12, 2003.

the other) and they are given a special name to indicate this property - *jumping emerging patterns*. Other useful, but less restrictive emerging patterns can also be defined. e.g. *Constrained emerging patterns* are minimal sets of items which occur $\geq \alpha$ times in one class and $\leq \beta$ times in the other. The set of items $\{d, o\}$ is such a pattern appearing in Class 1, for $\alpha = 3$ and $\beta = 1$.

In this paper, we discuss the principal types of emerging patterns, and focus on two principal issues

- *Methods for Efficient Pattern Discovery:* This is a challenge, especially for high volume, high dimensionality datasets, since in the worst case, the number of patterns present in the data can be exponential.
- *Classification Methods*: Constructing emerging pattern classifiers requires a number of different decisions to be made. What type of patterns should be mined, which patterns should be used in classification and the how are patterns weighted in the scoring function.

An outline of the rest of the paper is as follows. In section 2, we give some necessary background and terminology. Section 3 focuses on jumping emerging patterns and discusses their relationship to hypergraph transversals. Section 4 looks at more general kinds of emerging patterns. Section 5 examines algorithms for efficient pattern mining and section 6 discusses how to use emerging patterns in classification. In Section 7, we discuss related work and in section 8 provide a summary and outline directions for future research.

2 Terminology

To help formally define emerging patterns, some preliminary definitions are required. Assume a dataset D, defined upon some set of attributes $\{A_1, A_2, ..., A_n\}$, with each attribute A_i in turn defined by some number of values, its domain $domain(A_i)$. By aggregating all such domain values across all attributes, we obtain all the items $I = \{domain(A_1) \cup domain(A_2) \cup ... \cup domain(A_n)\}$, in our system. The dataset consists of a number of transactions, where each transaction, T, is some collection of the items in the system e.g. $T = \{v_1, v_2, ..., v_n\}$, $v_i \ \epsilon \ domain(A_i)$. An *itemset* is some subset of a transaction. The support of an itemset S in D is the number of transactions S occurs in - i.e. $(count_D(S))$. The relative support of S in D is the support of S divided by the number of transactions in D. $(count_D(S)/|D|))$. Assume two data sets D_p (the positive dataset) and D_n (the negative dataset). The growth rate of an itemset i in favour of D_p is defined as $\rho = \frac{supportD_p(i)}{supportD_n(i)}$. An itemset M satisfying a constraint $C = (support(M) \geq \alpha)$ is maximal if no proper superset of M satisfies C. An itemset N satisfying a constraint $C' = (support(N) \leq \beta)$ is minimal if no proper subset of N satisfies C'. An emerging pattern e is minimal if the itemset e is minimal.

An Emerging Pattern is an itemset whose support in one set of data differs from its support in another. Thus a ρ Emerging Pattern, favouring a class of data D_p, is one in which the growth rate of an itemset (in favour of D_p) is $\geq \rho$.

This growth rate may be finite or infinite. In the case that the growth rate ρ is infinite, we call the pattern a *jumping emerging pattern* (JEP) (i.e. it is present in one and absent in the another). It is this kind of pattern we first focus on.

3 Jumping Emerging Patterns and Hypergraphs

Jumping emerging patterns (JEPs) are particularly interesting due to their ability to represent strong contrasts. Finding all JEPs in D_p, with respect to D_n, requires discovery of all minimal itemsets which occur in at least one transaction of D_p and not in any transaction of D_n. It turns out that JEPs are closely connected to a problem in the theory of hypergraphs, a topic which itself has many applications in discrete mathematics and computer science.

One of the central questions in hypergraph theory is the problem of how to compute the minimal transversals of a given hypergraph. Example application areas related to this question range from minimal diagnosis and propositional circumscription [27], to learning boolean formulae [18], boolean switching theory [11] and discovering functional dependencies in databases [24].

The hypergraph minimal transversal problem is particularly significant from a data mining perspective. Indeed, the algorithmic complexity of mining maximal frequent itemsets and minimal infrequent itemsets is closely linked to the complexity of computing minimal hypergraph transversals [6, 18]. Although there has been considerable research in the data mining community with regard to efficient mining of maximal frequent itemsets (e.g. [4, 7, 17]), the companion problem of data mining of minimal infrequent sets is less well studied.

Infrequent itemsets are itemsets whose support is less than α in the dataset being mined (where α is some threshold ≥ 1). Minimal infrequent itemsets are infrequent itemsets such that no proper subset is infrequent. They are closely connected to JEPs. Consider the following example in table 2:

Suppose we are interested in finding jumping emerging patterns between Class A and Class B. Then transactions in Class A contain the following JEPs:

Transaction 1 $\{b\}$
Transaction 2 $\{ce, cg, cj\}$
Transaction 3 $\{cj, fj, hj\}$
Transaction 4 $\{af, ah, fj, hj\}$

Table 2. Sample Dataset

Trans_id_A	Class A	Trans_id_B	Class B
1	b,d,g,j	1	a,d,g,j
2	c,e,g,j	2	a,d,g,i
3	c,f,h,j	3	c,f,h,i
4	a,f,h,j	4	a,e,g,j

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

Each transaction t in Class A can be thought of as inducing a hypergraph with respect to all the transactions in Class B. The vertex set corresponds to the elements of t. Hypergraph edges are individually defined by subtracting a transaction in the negative dataset from the vertex set. So, for transaction $3 = \{c, f, h, j\}$ in class A, we have a hypergraph with vertex set $\{c, f, h, j\}$ and edges:

Hyperedge 1 $\{c, f, h\}$
Hyperedge 2 $\{c, f, h, j\}$
Hyperedge 3 $\{j\}$
Hypedge 4 $\{c, f, h\}$

The three minimal transversals of this hypergraph are $\{cj, fj, hj\}$, which correspond precisely to the JEPs listed above for transaction 3 in class A. They also correspond to minimal infrequent itemsets within class B, using threshold of $\alpha = 1$. Mining of JEPs in class A can thus be achieved by computing the minimal transversals for each possible choice of t in A.

There are several algorithms which can be used for computing minimal transversals. With respect to the data mining context, most of these techniques do not perform efficiently in practice. In particular, there is a performance gap on high dimensional datasets, that have a huge number of relatively short patterns. In section 5, we discuss some techniques to address this issue.

4 The Emerging Pattern Landscape

An emerging pattern (EP) has been defined as an itemset that has some specified growth rate ρ.

If we further specify thresholds α and β, we can also define another type of pattern, called a *constrained emerging pattern* (CEP). This is a minimal itemset i satisfying the following two conditions: i) $supportD_p(i) \geq \alpha$ instances, ii) $supportD_n(i) \leq \beta$ instances. Clearly, by setting $\beta = 0$, CEPs reduce to JEPs. By having a no-zero value of β, however, greater robustness to noise can be achieved, with (hopefully) little sacrifice in discriminating power

An advantage of JEPs was that they represent very sharp contrasts between D_p and D_n. Their disadvantage is that low quality datasets may contain an abundance of low support JEPs and few high support JEPs. This is because the requirement that a JEP never occur within D_n is often too strict. Strong inherent features of D_p sometimes do appear within D_n, due to the presence of noise or recording errors. Such features do not qualify as JEPs and hence would

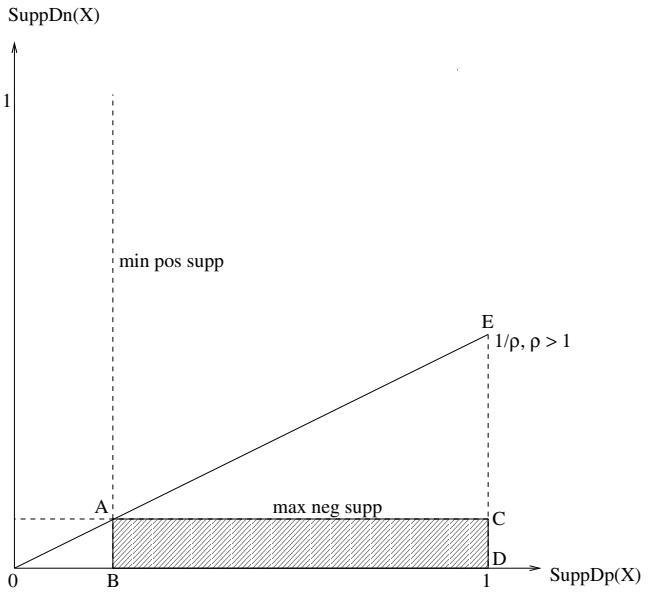

Fig. 1. Space of Emerging Patterns

not be found in JEP discovery, a potentially serious problem when using JEPs as the basis of a classifier.

Unlike JEPs, EPs may occur within D_n, provided the growth rate threshold ρ is satisfied. Hence, they are potentially more stable and resistant to noise. The disadvantage, though, is that the contrasts may no longer be as 'sharp'. e.g. An EP with $\rho = 5$ could have $supportD_p = 100$ and $supportD_n = 20$, values which arguably do not discriminate as sharply between D_p and D_n, as a JEP of $supportD_p = 50$. Indeed, in practice, our experience is that JEP-based classifiers are superior to EP based classifiers for exactly this reason.

Figure 1 illustrates the support plane for EPs, JEPs and CEPs. JEPs occupy the x-axis from O to D. EPs occupy the trapezoid ABDE, while CEPs occupy the shaded rectangle ABDC. CEPs can thus be viewed as occupying an intermediate space between the entire set of EPs and those JEPs that lie on the BD axis

Other varieties of emerging patterns can also be defined, by incorporation of further constraints. One example is "interesting emerging patterns" [14]. These are similar to constrained emerging patterns, but additionally need to satisfy an interestingness constraint, specified using a chi-square statistical measure.

5 Efficient Pattern Mining

We now briefly describe techniques for mining i) JEPs, ii) CEPs and iii) General EPs. In general this is a very challenging problem, since if the dimensionality of

the datasets is n, then the search space is 2^n. For large n, it is prohibitive to examine this exhaustively and more sophisticated methods are required.

As discussed in section 3, mining JEPs with respect to a single transaction from D_p, corresponds to the problem of enumerating the minimal transversals of a hypergraph. We have developed a partitioning algorithm [3], which performs this task faster than any other algorithms we are aware of. The idea is to recursively partition the input hypergraph into a set of smaller, manageable hypergraphs, which can then have their minimal transverals enumerated by an optimisation of the well-known method of Berge [5].

Of course JEPs must be mined from every transaction in D_p, not just a single one. One method is to just iterate through every transaction in D_p and solve the corresponding hypergraph problem to obtain the complete result. It is possible to improve on this, however, by use of a tree structure to store the transactions in D_p and D_n [1]. This allows transactions in both D_p and D_n to overlap, thus reducing the number of separate hypergraph problems which need to be solved.

CEP mining can be accomplished by an extension of JEP mining. Step i) is to represent both the positive and the negative datasets being mined in terms of their border descriptions. Step ii) is to mine the CEPs by operating on the relevant borders.

Step 1 constructs two borders. One representing the positive dataset, such that only those (maximal) itemsets with support $\geq \alpha$ are present and the other border representing the negative dataset with maximal itemsets having support $\geq \beta$. Finding these borders can be achieved by utilising a maximal frequent itemset generator (e.g. any of [4, 7, 17]). Once the borders are computed, the method for mining JEPs can then be applied to gain the desired patterns in step ii). i.e. Finding all minimal itemsets which occur in the positive border and are absent from the negative border. Observe that each pattern output from this process satisfies the original user specified support constraints.

Efficient mining of general EPs (defined using just a nonzero growth rate ρ) appears difficult. Work in [28] employed a set-enumeration tree [28] and optimisations similar to the MaxMiner algorithm [4]. Another possibility is to mine sets of CEPs for different values of α and β and then conduct a post-pruning step to check whether the growth rate ρ is satisfied. This is an incomplete method, since some EPs may not be discovered. Another incomplete EP mining method is presented in [12], where a set enumeration tree is built and patterns are only mined to a certain depth in the tree.

5.1 Incremental Maintenance of JEPs

Efficient algorithms for mining JEPs should aim to avoid having to completely re-do computation, if small changes are made to either the positive dataset D_p or the negative dataset D_n. Assuming that the changes to D_p and D_n are small compared to the volume of the old data, it is likely that many of the JEPs will remain valid. Incremental maintenance techniques aim to avoid expensive re-computation by efficiently making use of the previous set of EPs. Work in [20] gives efficient incremental maintenance rules for i) Inserting new

instances into D_p, ii) Inserting new instances into D_n, iii) Deleting instances from D_p and iv) Deleting instances from D_n. Cases where an item is deleted or new item is inserted are also examined.

6 Classification with Emerging Patterns

We now discuss how a set of emerging patterns may be used in classification. We do not present any experimental results, but work in [2, 12, 14, 13, 19, 22, 10, 20] attests to the high accuracy of classifiers which can be built.

First assume that all continuous attributes from the datasets have been discretised. Our method of choice is the entropy technique from [15].

6.1 Support Aggregation

Suppose we have two dataset classes D_1 and D_2 and assume that the (minimal) EPs have been mined for each of them. Let the set of EPs for D_1 be E_1 and the set of EPs for D_2 be E_2. Now given a test instance t, for which we need to assign a class label, we generate a new set E_1', which contains all patterns from E_1 that are a subset of t. Similarly generate the set E_2'. A score is now calculated for each of the classes. The score for D_1 is the sum of the individual relative supports for each EP in E_1'. Similarly for D_2. The test instance is assigned a label corresponding to the class with the higher score.

If there are more than two classes, then the procedure is similar. If the classes are $D_1, D_2, \ldots D_n$, then the EPs for class D_1 are found by comparing D_1 against the (negative) dataset $D_2 \cup D_3 \cup \ldots D_n$. The EPs for class D_3 are found by comparing D_3 with respect to the (negative) dataset $D_1 \cup D_2 \cup D_4 \cup D_5 \ldots \cup D_n$ etc.

Observe that in the simple aggregation method, the impact of each individual EP is equal to its relative support. Classifiers that use simple aggregation were discussed in [19, 10].

6.2 Bayesian Scoring

Although easy to implement, using simple aggregation to compute a class score is not based on any solid statistical foundation.

A way of attacking this problem is to use a scoring function based on Bayes theorem, which says that the chosen class should be the one which maximises

$$P(C_i|T) = P(T, C_i)/P(T) = P(C_i)P(T|C_i)/P(T)$$

where C_i is the class label, $T = \{a_1 a_2 \ldots a_n\}$ 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 prediction of a single class, the denominator $P(T)$ can be ignored, since it will not affect the relative ordering of classes. Since it is very difficult in practice to calculate the probability $P(T, C_i)$, one must use approximations.

The approximation used here incorporates the use of a set of emerging patterns to derive a product approximation of $P(T, C_i)$, using the chain rule of probability $P(X_1, X_2, \ldots, X_n) = P(X_1)P(X_2|X_1)\ldots P(X_n|X_1, \ldots, X_{n-1})$. The product approximation of the probability of an n-item 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 covered in the previous itemsets. e.g. Consider a test instance $T = \{a_1a_2a_3a_4a_5\}$ and a set of emerging patterns contained in the test instance $B = \{\{a_1, a_2, a_3\}, \{a_2, a_5\}, \{a_3, a_4\}, \{a_1, a_4, a_5\}\}$. The approximation using this set B of EPs is $P(C_i)P(a_1a_2a_3|C_i)P(a_5|a_2C_i)P(a_4|a_3C_i)$. The product approximation is created by adding one EP at a time until no more EPs can be added (either all the items of the remaining EPs are already covered or no more EPs are available). The probabilities $P(a_1a_2a_3|C_i)$, $P(a_5|a_2C_i)$ and $P(a_4|a_3C_i)$ are then computed from the dataset.

Of course the order in which EPs are added will affect the resulting approximation. Adding in order of support is advantageous, since greater support is likely to mean greater reliability of the resulting factor in the approximation.

A Bayesian approach to emerging patterns is described in [13], which in turn is based on the large Bayes classifier from [25].

6.3 Pairwise Classification

Although JEPs and CEPs have good classification performance on two class problems, their performance for multi-class problems is less impressive [2]. The crucial observation here is that EP-based classifiers seem inherently designed for dealing with two-classes, because multi-class problems get collapsed into two class ones, as earlier described.

To overcome this, it is possible to employ a pairwise technique. Suppose there are n different classes. The pairwise mechanism sees an $n(n-1)/2$ number of mining operations (one for each pair), replacing the n number that would normally be performed. For each pair, a winner is determined using a scoring function (e.g. simple aggregation). Following this process, each class has a tally of wins in its favour. The class with the highest number of wins is assigned to the test case. Discretisation is performed for each pair of classes, rather than once at the beginning.

Employing this pairwise strategy greatly improves classification accuracy in such multi-class scenarios. This appears to be because success of each class D_p in generating patterns is related to the negative dataset (D_n). If these sets are similar, a small number of patterns will result. If the negative set size far exceeds the positive set size, few patterns will be generated. Having more balanced pairs of D_p and D_n means that a greater number of EPs can be generated.

6.4 Lazy Classification

The classifiers so far discussed have been eager. i.e. They do a once only computation of all the EPs and then use them if they are contained within the given test instance t. In contrast, it is also possible to compute EPs in a lazy fashion,

based on knowledge of the test. This results in better classifier accuracy, since discretisation can now be targeted towards the test instance [20].

Assume the sets $D_p = \{M_1, \ldots, M_m\}$ and $D_n = \{N_1, N_2, \ldots, N_n\}$ and assume continuous valued attributes have *not* been discretised. Consider a fixed test instance t. The following steps are used

- Take the intersection of each training instance with t, namely $t \cap M_1, \ldots, t \cap M_m$ and $t \cap N_1, \ldots, t \cap N_n$. This operation is equivalent to removal of irrelevant training values.
- Select the maximal itemsets from $\{t \cap M_1, \ldots, t \cap M_m\}$ and similarly from $\{t \cap N_1, \ldots, t \cap N_m\}$. Denote the former collection of maximal itemsets as max_{R_m} and the latter as max_{R_n}.
- Find all JEPs in max_{R_m} and sum their relative supports to get a score for Class 1. Similarly find all JEPs in max_{R_n} and sum their relative supports to get a score for Class 2.

Intersection is performed as follows. Assume $attr_A$ is a continuous valued attribute having domain $[0, 1]$ (if not, it can be scaled into this range). Given a training instance s, $t \cap s$ will contain the $attr_a$ value of t, if the $attr_A$ value of s is in the neighbourhood $[x_1 - \alpha, x_1 + \alpha]$, where x_1 is the attribute value for t. The parameter α is called the neighbourhood factor, which can be used to adjust the length of the neighbourhood. Experiments have shown 0.12 to be a suitable value in practice.

7 Related Work

The concept of emerging patterns was first introduced in [8]. Jumping emerging patterns appeared first appeared in [19, 9] and constrained emerging patterns in [2].

Classification has a long history and their exist a multitude of machine learning algorithms for it. In this paper, we have only focused on emerging pattern methods for classification.

The CBA classifier [23] uses association rules for classification that can be interpreted as being a kind an emerging pattern. Rules must satisfy a minimum threshold (typically 1% relative support) and a minimum confidence (typically 50%). Translated into our framework, this would mean mining all emerging patterns having minimum relative support in D_p of 1% and maximum relative support of β in D_n, where $\beta \leq \alpha$. This constraint on the β value is much more permissive than what is used in the CEP classifier in [2], which uses the low value of $\beta = 1$.

Pairwise classification was discussed in [16], where it was applied to a number of classifiers and was seen to result in increased accuracies on some datasets. It was also observed that the gains achieved were roughly proportional to the number of classes. This is in line with our reported results in [2].

Emerging patterns are similar to version spaces [26]. Given a training set of positive instances and a set of negative instances, a version space is the set of

all generalisations that each match (or are contained in) every positive instance and no negative instance in the set. Therefore the consistency restrictions with the training data are quite different for JEP spaces.

8 Summary and Future Work

In this paper, we have examined a number of kinds of emerging patterns and described techniques for using them in classification and methods for efficiently mining them. Future research directions include i) Methods for discovery of the top k EPs, ranked by support. ii) Further developing an understanding of the foundations of EPs and their relationship to other structures in logic and machine learning.

References

[1] J. Bailey, T. Manoukian, and K. Ramamohanarao. Fast Algorithms For Mining Emerging Patterns. In *Proceedings of the Sixth European Conference on Principles of Data Mining and Knowledge Discovery*, pages 39–50, 2002. 6

[2] J. Bailey, T. Manoukian, and K. Ramamohanarao. Classification using constrained emerging patterns. In *Proceedings of the 4th International Conference on Web Age Information Management*, 2003. 7, 8, 9

[3] J. Bailey, T. Manoukian, and K. Ramamohanarao. A fast algorithm for computing hypergraph transversals and its use in mining emerging patterns. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, 2003. 6

[4] R. J. Bayardo. Efficiently Mining Long Patterns from Databases. In *Proceedings of the International Conference on Management of Data*, pages 85–93, 1998. 3, 6

[5] C. Berge. *Hypergraphs, North Holland Mathematical Library*, volume 45. Elsevier Science Publishers B.V (North-Holland), 1989. 6

[6] E. Boros, V. Gurvich, L. Khachiyan, and K. Makino. On the complexity of generating maximal frequent and minimal infrequent sets. In *STACS 2002, 19th Annual Symposium on Theoretical Aspects of Computer Science, Antibes - Juan les Pins, France, March 14-16, 2002, Proceedings*, pages 133–141, 2002. 3

[7] Doug Burdick, Manuel Calimlim, and Johannes Gehrke. MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases. In *Proceedings of the 17th International Conference on Data Engineering*, pages 443–452, 2001. 3, 6

[8] G. Dong and J. Li. Efficient Mining of Emerging Patterns: Discovering Trends and Differnces. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pages 43–52, 1999. 9

[9] G. Dong, J. Li, and X. Zhang. Discovering Jumping Emerging Patterns and Experiments on Real Datasets. In *Proceedings of the Ninth International Database Conference on Heterogeneous and Internet Databases*, pages 15–17, 1999. 9

[10] G. Dong, X. Zhang, L. Wong, and J. Li. CAEP: Classification by Aggregating Emerging Patterns. In *Proceedings of the Second International Conference on Discovery Science*, pages 30–42, 1999. 1, 7

[11] T. Eiter and G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *Siam Journal of Computing*, 24(6):1278–1304, 1995. 3

[12] H. Fan and K. Ramamohanarao. An Efficient Single-Scan Algorithm for Mining Essential Jumping Emerging Patterns. In *Proceedings of the Sixth Pacific Asia Conference on Knowledge Discovery in Databases*, pages 456–462, 2002. 6, 7

[13] H. Fan and K. Ramamohanarao. A bayesian approach to use emerging patterns for classification. In *Database Technologies 2003, Proceedings of the 14th Australasian Database Conference, ADC 2003, Adelaide, South Australia*, volume 17. Australian Computer Society, 2003. 7, 8

[14] H. Fan and K. Ramamohanarao. Efficiently mining interesting emerging patterns. In *Proceedings of the 4th International Conference on Web Age Information Management*, 2003. 5, 7

[15] U. M. Fayyad and K. B. Irani. Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1022–1027, 1993. 7

[16] J. Furnkranz. Pairwise Classification as an Ensemble Technique. In *Proceedings of the Thirteenth European Conference on Machine Learning*, pages 97–110, 2002. 9

[17] K. Gouda and M. J. Zaki. Efficiently Mining Maximal Frequent Itemsets. In *Proceedings of the First International Conference on Data Mining*, pages 163–170, 2001. 3, 6

[18] Dimitrios Gunopulos, Roni Khardon, Heikki Mannila, and Hannu Toivonen. Data mining, hypergraph transversals, and machine learning. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 12-14, 1997, Tucson, Arizona*, pages 209–216. ACM Press, 1997. 3

[19] J. Li, G. Dong, and K. Ramamohanarao. Making use of the most Expressive Jumping Emerging Patterns for Classification. In *Proceedings of the Fourth Pacific Asia Conference on Knowledge Discovery in Databases*, pages 220–232, 2000. 1, 7, 9

[20] J. Li, G.Dong, and K. Ramamohanarao. DeEPs: Instance-Based Classification by Emerging Patterns. In *Proceedings of the Fourth European Conference on Principles of Data Mining and Knowledge Discovery*, pages 191–200, 2000. 1, 6, 7, 9

[21] J. Li, H. Liu, J. R. Downing, A. Yeoh, and L. Wong. Simple rules underlying gene expression profiles of more than six subtypes of acute lymphoblastic leukemia (ALL) patients. *Bioinformatics*, 19:71–78, 2003. 1

[22] J. Li and L. Wong. Emerging Patterns and Gene Expression Data. In *Proceedings of the Twelfth Workshop on Genome Informatics*, pages 3–13, 2001. 1, 7

[23] B. Liu, W.Hsu, and Y. Ma. Integrating Classification and Association Rule Mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 80–86, 1998. 9

[24] H. Mannila and R. Kari-Jouko. Algorithms for inferring functional dependencies from relations. *Data and Knowledge Engineering*, 12(1):83–99, 1994. 3

[25] Dimitris Meretakis and Beat Wüthrich. Extending naïve bayes classifiers using long itemsets. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 165–174. ACM Press, 1999. 8

[26] T. M. Mitchell. Generalization as Search. *Artficial Intelligence*, 18(2):203–226, 1982. 9

[27] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987. 3

[28] X. Zhang, G.Dong, and K. Ramamohanarao. Exploring Constraints to Efficiently Mine Emerging Patterns from Large High-Dimensional Datasets. In *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining*, pages 310–314, 2000. 6