Classification Using Constrained Emerging Patterns

James Bailey, Thomas Manoukian and Kotagiri Ramamohanarao
Department of Computer Science & Software Engineering
The University of Melbourne, Australia
{jbailey,tcm,rao}@cs.mu.oz.au

Abstract. Emerging Patterns are itemsets whose supports change significantly from one dataset to another. They are useful as a means of discovering distinctions inherently present amongst a collection of datasets and have been shown to be a powerful method for constructing accurate classifiers. In this paper, we present two techniques for significantly improving emerging pattern classifying power. The first strategy involves mining patterns which have a more targeted description of their relative supports in each dataset. The second technique is to employ a pairwise classification strategy for situations where more than two classes are present. Novel mining algorithms are also presented which emphasise dataset partitioning as a crucial mechanism in reducing the complexity of the task. We provide experimental results demonstrating the value of these techniques and show that in general, the resulting classifier performs demonstrably better than other preeminent methods, while mining time is considerably improved on earlier methods.

1 Introduction

Discovery of powerful distinguishable features between datasets is an important objective in data mining. Addressing this problem, work presented in [3] introduced the concept of *emerging patterns* or EPs. These are itemsets whose support changes significantly from one dataset to another. Because of sharp changes in support, emerging patterns have strong discriminating power and are very useful for describing the contrasts that exist between classes of data. They have been successfully applied for discovering patterns in gene expression data [13]. Work in [11, 4, 12] has shown how to use them as the basis for constructing competitive data classifiers.

Previous strategies for using emerging patterns in classification have, however, suffered from a number of limitations: i) An inability to identify patterns satisfying specified support constraints in each of the datasets (i.e. at least three occurrences in dataset A and less than two occurrences in dataset B). This can mean that powerful patterns may be excluded from the classification process. ii) Poorer classification performance in general for multi class scenarios. Challenges also exist in the mining of emerging patterns. Two well known obstacles are: i) the apriori property no longer holds for EPs ii) a large number of EPs may exist.

In this paper, we tackle these problems, with the aim of improving the overall accuracy of emerging patterns for classification. Our main contributions are:

- We identify a useful new class of emerging pattern that we call a *constrained emerging pattern* (CEP), having known frequency bounds within each of the opposing datasets. We demonstrate its improved classification power in comparison to previous types of emerging patterns.
- We introduce algorithms for efficiently mining CEPs which achieve a 26-165 speedup on currently known techniques.
- We next show how classification accuracy for CEPs can be significantly improved for multi class problems by the use of pairwise classification.
- Finally, we compare the performance of CEPs with several other classifiers - CBA, Naive Bayes and C5.0 and show that CEPs generally have superior accuracy.

An outline of the remainder of this paper is as follows. In section 2 we give some necessary background and terminology. Section 3 motivates the use of CEPs for classification. Section 4 provides a detailed description of the algorithms used to mine CEPs and provides some experimental analysis of their performance. Section 5 compares the accuracy of the CEP based classifier against other emerging pattern methods and other state-of-the-art systems. Section 6 then discusses the use of pairwise classification for CEPs and extends the comparison with techniques seen in section 5. In section 7 we discuss related work. Finally, in section 8 we provide a summary and outline directions for future research.

2 Background and Terminology

Let the support of an itemset i in a dataset D be the count of the number transactions within D containing i, denoted $supportD(i)$. Assume two data sets D_p (positive dataset) and D_n (negative dataset), the growth rate of an itemset i in favour of D_p is defined as $\rho = \frac{supportD_p(i)}{supportD_n(i)}$ ($\frac{0}{0}$ implies the pattern exists in neither dataset). An *emerging pattern* (EP) is an itemset whose support in one set of data differs from its support in another. The growth rate is thus either finite or infinite. Should it be infinite, we say the pattern is a *jumping emerging pattern* (JEP) (i.e. it is present in one and absent in the other).

Classification using emerging patterns is described in [11]. Initially all emerging patterns for each of the two classes are computed. Then, given some test instance, a score is calculated for each class. This score is equal to the sum of the contributions made by each individual emerging pattern contained within the test. The contribution of an individual EP to the overall score is equal to its support. The test instance is deemed to match the class with the highest overall score.

3 Constrained Emerging Patterns

We begin by discussing the two kinds of emerging patterns that have previously been used for classification. An EP is a pattern that has some specified growth rate ρ. A JEP is special kind of EP with $\rho = \infty$. The advantage of JEPs is 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. Thus the requirement that a JEP never occur within D_n is often too strict and strong inherent features of D_p may be overlooked.

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. setting $\rho = 5$ is a common choice [17]) so the ability to mine an emerging pattern i based on desired input values of $supportD_p(i), supportD_n(i)$ (from which ρ can be derived) is likely to be valuable. This forms the basis of a new class of emerging pattern that we term a *constrained EP* (CEP). A CEP satisfies the following definition:

Given thresholds α and β, a CEP is an itemset i satisfying the following two conditions: i) $supportD_p(i) \geq \alpha$, ii) $supportD_n(i) \leq \beta$. Clearly, by setting $\beta = 0$, CEPs reduce to JEPs. By varying the value of β, greater robustness to noise can be achieved.

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).

Table 1. Sample Database

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

By observation we see that JEPs for class 1 include $\{a, i\}$, $\{o, p\}$, and $\{b, g\}$. JEPs for class 2 include $\{f\}$, $\{k\}$, and $\{n\}$. Class 1 also contains an EP, $\{o, r\}$ ($\rho = 2$) and a CEP, $\{d, l, o\}$ ($\alpha = 3$ and $\beta = 1$).

4 CEP Mining Issues and Objectives

The first work [3] introducing the mining of emerging patterns showed that the process is two stage.

- Represent both the positive and negative datasets being mined in terms of their border descriptions.
- Mine the emerging patterns by operating on the relevant borders.

The first stage of the procedure constructs two borders. One will represent the positive dataset such that only those itemsets with support $\geq \alpha$ are present and the other will represent the negative dataset where members have support $\geq \beta$.

The second stage implies taking these borders and applying a border-difference [3] operator, thus gaining the desired patterns. This operator extracts all minimal itemsets that occur in the positive border and are absent from the negative one.

Our work in [1] focused on JEP mining and provided an improved method of completing stage ii) of the mining process mentioned above. There $\alpha = 1$ and $\beta = 0$ so the border descriptions are simply the dataset transactions. CEP mining presents a much greater challenge, in this work we choose $\alpha = 3$ and $\beta = 1$, hence the borders are no longer trivially the original datasets themselves. We show that we can significantly improve completion of stage ii) by employing a more sophisticated scheme of data partitioning. We construct partitions with two primary aims:

Property 1 To minimise the number of itemsets present in each partition

Property 2 To minimise the cardinality of itemsets contained in large partitions

4.1 General Principles

The principal objective of the mining algorithm is to form partitions, containing both instances from the positive and negative border, in line with the properties listed above. On forming such a partition, we then apply the border-difference operator to each. This procedure equates to generating and solving many small problems rather than a single (possibly very large) one.

The primary data structure we use is a prefix-tree based upon the FP-Tree [9]. Such a tree can be viewed as containing many root nodes, one for each distinct prefix item. The sub-tree headed by all such root items are termed component trees. The basis of our algorithm is to:

– traverse branches such that each item visited acts as a partitioning item.

Given that all branches within any component tree share the root item this can be seen as a first partition, by selecting any other item (item x for example) we can:

– accumulate all branches (itemsets) that share the root item and x thus achieving a second partition of the database.

Once all relevant branches have been found, we can:

– invoke the border-difference operator on the itemsets.

To ensure that each component tree consists of all the branches which contain the root item we relocate all branches. This process involves the removal of the highest ordered item in the branch and re-inserting into the tree.

4.2 Item Ordering and Quality of Partitioning

Choice of item ordering is crucial for mining EPs, since it is the order of items which dictates the size and dimensionality of the itemsets within a partition.

Root-Level Ordering An ordering of items will result in some particular component tree structure. Trees will exist that contain many branches and others not so many, these branches may be long (relocated few times) or short (relocated many times).

Choosing a frequent ordering (items are ordered in descending order of frequency with reference the set of transactions constituting both the positive and negative border) implies that the component tree headed by the most common item will contain all relevant branches at their maximum original size (remembering that relocating a branch reduces its size by 1). In this scenario the first few component trees contain many long branches. To address this we note that:

– the two-item partition will attract many branches.
– their lengths in many cases will be large.

both are contradictions of **Property 1** and **Property 2**. Irrespective of ordering, the branches that lie under any component tree will be identical in the sense that all information pertinent to that tree will be present. Therefore it is favourable to:

– minimise the length of the branches under the larger component trees.

An obvious solution is to use an infrequent ordering. Such a choice implies that:

– any branch containing a high frequency item will be short.

The actual length is dependent upon the number of relocations the branch has experienced. Before being relocated to some component tree headed by a high frequency each item (item y for example), the branch would have been relocated (shortened in length by 1) n times, where n is the number of items present in the branch that reside higher than y in the item order. In the case of the most frequent item, all branches will be of length 1.

Internal Component-Tree Ordering An infrequent ordering on its own will not guarantee either of our two stated properties. A key contribution in this paper is to demonstrate how the application of a further internal level of ordering within component trees greatly improves the efficiency of CEP mining.

Given a top root-level ordering, we need to calculate a new ordering specific to the branches (itemsets) contained under each component tree. This ordering is based on the frequency of items present in all branches containing the root item of the component tree. For each component tree the process involves:

– finding which itemsets reside under the component tree.
– acquiring a frequency table for all items within the relevant itemsets.

- inserting itemsets using this internal order.

Work to this stage promotes a frequent ordering within each component tree as a useful general approach. The two-level ordering can be seen to tackle both of our mining objectives.

- The root-level ordering ensures that the ratio of branch length to component tree size (number of nodes) is minimised.
- average cardinality of itemsets within component trees headed by high frequency items is small.

These points are a direct consequence of the number of relocations individual branches undergo. The second-level ordering ensures:

- that the number of branches within any one component tree is minimised.
- any two-item partition attracts the smallest possible number of branches.

A frequent item order means that longer prefixes exist within component trees. As a result, component trees contain a minimal number of branches. Figure 1 shows a probable structure of component trees. Trees headed by high frequency items have branches of short length, component trees with long branches in turn have very few of them.

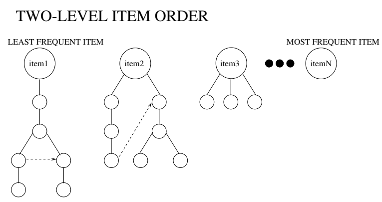

Fig. 1. Component Tree structure

Implementation We now discuss some additional details concerning our algorithm. Maximal Frequent Itemsets (MFIs) were generated using the GenMax [8] algorithm. We present two approaches to mining CEPs that arise from questioning whether we should:

- find the positive border initially (eager approach) *or*
- find all EPs satisfying the negative support constraint and then prune away those not satisfying the minimum positive threshold (lazy approach).

The intention of the lazy approach is to avoid circumstances where the borders generated may be much larger than the initial datasets. By transferring the pruning away from the MFI generation stage, we may avoid dealing with a considerably sized positive border, pruning is then required as a post-processing step. Algorithm 1 provides pseudo-code for the entire mining process.

Algorithm 1 CEP Mining

```
C = {Class_1, Class_2, ..., Class_n}
EPs = ∅
for all c ∈ C do
  pos_border = find_MFI(positive_dataset, α)
  neg_border = find_MFI(negative_dataset, β)
  root_level_order = find_infrequent_order(pos_border, neg_border)
  assign_all_branches_to_relevent_component_trees(pos_border, neg_border)
  for component_tree = 0 to num_component_trees do
    component_tree_level_order = find_frequent_order()
    insert_branches()
    for all branches b ∈ component_tree do
      for all items i ∈ b do
        relevant_branches = gather_all_branches()
        EPs = EPs ∪ border_difference(relevant_branches)
      end for
    end for
  end for
end for
```

We conducted experiments on three datasets from the UCI Machine Learning Repository [2] and compared the original method proposed in [3], in which no tree structures are used and the borders produced in stage i) are operated upon directly, the single-level ordering of [1], the method proposed here and its lazy variant. All experiments were performed on a Dell PowerEdge 2600 2.4GHz with 3GB memory running Solaris 8 /x86 with all times being user times. The 3rd column of the table shows the number of attributes and total number of distinct attribute values (items) present in each problem following discretisation.

Table 2. CEP Mining

Dataset	#classes	#attr:items	#instances	original	single-level	CEP-Lazy	CEP-Eager	Speedup
census	2	14:154	32561	5327.01	226.83	44.47	**32.24**	165.22
chess	2	36:73	3196	2050.85	206.90	153.94	**78.30**	26.19
mushroom	2	22:125	8124	519.58	14.09	**4.50**	6.16	115.46

The results of Table 2 indicate the superiority of the algorithms presented in this work for the mining of CEPs. In all cases, considerable speedups exist between these methods and those previously developed. In comparison to the method presented in [3] we see improvements between 26 and 165 times. In contrast to the single-level ordering of [1] the improvement ranges from 2 to 7 times faster. The performance of the lazy algorithm is dependent upon the ratio of EPs generated prior to pruning and those that remain following it. The fact that the lazy approach is outperformed by the eager method suggests that this ratio is high.

5 Classification Using CEPs

We have implemented a system for mining CEPs setting $\alpha = 3$ and $\beta = 1$ (values shown through experimentation to provide strong overall classification behaviour) We also use the heuristic that $\beta = 0$ if $|D_n| < 100$ - meaning that for situations where the negative dataset is sufficiently small, JEPs are mined instead.

A ten-fold cross-validation methodology was employed, with the same folds being used in all classifiers. The JEP classifier is implemented as described in [1]. The EP classifier is implemented as described in [4, 17]. For each of the emerging pattern methods, the entropy technique in [6] was used for discretising continuous attributes.

Table 3. Classification Results: Two Class Problems

Dataset	# classes	CEP	EP	JEP	CBA[14]	C5.0[16]	NB
australian	2	85.36	**85.80**	84.64	84.10	84.78	76.96
breast-w	2	96.85	**97.28**	96.71	95.80	94.28	96.43
cleve	2	**82.84**	79.86	80.19	76.80	74.88	81.52
heart	2	82.22	-	81.48	80.00	76.30	**83.70**
crx	2	**86.23**	86.09	85.36	84.00	86.09	77.54
tic-tac-toe	2	92.07	89.66	97.28	**100**	86.43	69.63
hepatitis	2	84.50	84.46	**85.17**	81.30	74.21	83.21
diabetes	2	75.26	73.83	72.40	74.10	72.92	**76.83**
horse-colic	2	83.72	82.03	83.44	82.40	**87.81**	79.08
house-votes-84	2	92.64	91.49	94.25	96.00	**97.24**	89.66
labor-neg	2	89.67	89.67	89.67	85.00	82.67	**93.33**
hypothyroid	2	98.74	96.43	98.61	98.20	**99.21**	97.72
Two Class Ave		**87.50**	86.96	87.43	86.47	84.73	83.80

Observing the results for the two class datasets (Table 3), we see that the three emerging pattern classifiers all perform very strongly. Within the three, CEP has the best average accuracy and also has the most number of top accuracies.

Classification with multi class problems traditionally involved n mining operations where mining for any particular class sees it act as the positive dataset and the remaining $n-1$ classes aggregated to form the negative dataset. The performance of the emerging pattern methods is far less competitive with respect to the non emerging pattern methods (see Table 4). This represents an important deficiency and addressing this problem is the subject of the next section.

6 Pairwise Classification for Multi Class Datasets

The results for multi-class problems are not so impressive. The crucial observation lies in our belief that all EP-based classifiers are inherently designed for

Table 4. Classification Results: Multi Class Problems

Dataset	# classes	CEP	EP	JEP	CBA	C5.0	NB
shuttle-small	4	99.76	98.24	**99.81**	99.70	99.79	90.91
iris	3	94.67	92.00	94.67	93.60	94.67	**96.67**
glass	6	64.00	63.05	63.51	**67.10**	66.80	43.40
yeast	10	41.79	53.64	38.28	56.50	54.05	**57.95**
lymphography	4	81.10	81.76	78.38	80.40	79.05	**83.76**
vehicle	4	69.15	64.30	71.04	70.50	**72.22**	47.17
flare	6	61.62	69.33	65.73	**73.80**	72.78	71.20
anneal	5	93.09	94.32	93.76	**96.10**	92.53	81.41
automobile	6	75.14	68.81	77.57	71.70	**82.45**	52.21
soybean-small	4	95.50	**100**	95.50	97.50	95.50	98.00
wine	3	89.41	97.19	93.86	94.10	95.56	**97.78**
nursery	5	96.98	84.71	**98.85**	92.50	97.13	90.30
zoo	7	94.09	92.09	**94.18**	93.40	90.18	92.18
Multi Class Ave		81.25	81.49	81.93	83.60	**84.05**	77.14

dealing with two-classes given the traditional approach mentioned in the last section. It is this procedure that we will alter so that this inherent feature is fully exploited.

Suppose there are n different classes. Our mechanism sees an $n(n-1)/2$ number of mining operations (one for each pair) replacing the n number traditionally performed. Following this process each class has a tally of votes in its favour. The class with the highest number of votes is assigned to the test case. Discretisation is performed for each pair of classes, rather than once at the beginning, a choice which we found greatly improves classification accuracy.

Table 5 shows the effect of performing binarisation on those multi-class problems examined previously. We only show the effect of pairwise classification for CEP, since it earlier outperformed both EPs and JEPs on two class problems (see Table 3, CEP had 6 top accuracies, EP had 2 and JEP had 3. CEP also had the best two class average.).

We also performed binarisation for C5.0 and NB, the overall effect was negligible in terms of average accuracy (C5.0 -0.36%, NB -0.67%) and top accuracies. We also conducted the experiment using JEPs with average accuracy rising to 83.86%. The results are clear in indicating the substantial increase in CEPs' performance with regard the multi-class problems. Noticeable improvement was found with *yeast* (14.07%), *glass* (7.49%), *wine* (5%) and *flare* (9.58%). The average accuracy increase (2.95%) over all multi-class datasets is also meaningful.

The most significant result is that CEP is now globally competitive when compared to the non emerging pattern classifiers. Our explanation of why this process is successful, centers around the underlying method of mining EPs.

- success of each class in generating patterns is related to the negative dataset.
- if these sets are similar a small number of patterns will result.

Table 5. CEP using Pair-Wise Algorithm

Dataset	# classes	CEP	CBA	C5.0	NB
shuttle-small	4	**99.81**	99.70	99.79	90.91
iris	3	94.00	93.60	94.67	**96.67**
glass	6	**71.49**	67.10	66.80	43.40
yeast	10	55.86	56.50	54.05	**57.95**
lymphography	4	79.81	80.40	79.05	**83.76**
vehicle	4	65.96	70.50	**72.22**	47.17
flare	6	71.20	**73.80**	72.78	71.20
anneal	5	94.54	**96.10**	92.53	81.41
automobile	6	74.67	71.70	**82.45**	52.21
soybean-small	4	**100**	97.50	95.50	98.00
wine	3	94.41	94.10	95.56	**97.78**
nursery	5	**97.77**	92.50	97.13	90.30
zoo	7	**95.09**	93.40	90.18	92.18
Multi Class Ave		**84.20**	83.60	84.05	77.14

– if the negative set size far exceeds the positive set size few patterns will be generated.

Our research indicated that these scenarios are more likely when the earlier method of handling of multi-class problems is used. In the case of *yeast*, each class is essentially paired against nine others and in the worst case (for the smallest class) over 99% of the entire dataset. In such circumstances the likelihood that such a class will produce any number of significant patterns (and thus be a viable alternative during classification) is remote. By reducing the size of the negative set that any one class competes against during mining, the chances that they will produce patterns increases. The gains made can be attributed to the outcome of this reasoning.

Looking at all the results in Table 5, using CEPs with pair-wise mining is extremely successful. In comparison to results obtained without its use, the CEP classifier, on multi-class problems, has improved by almost 3% on average accuracy. When contrasting against other popular classifiers we see that CEP achieves a higher overall average accuracy (+0.15% on nearest rival C5.0) and 5/13 top accuracies (C5.0 2, CBA 2, and NB 4). Coupled with the fact that it outperforms these three classifiers on two classes, a greater average accuracy of 1.03%, and 5/12 overall top accuracies (C5.0 3, CBA 1, NB 3), CEP is shown to be a very powerful alternative to those systems currently available.

7 Related Work

Emerging patterns first appeared in [3], which also introduced the notion of the border for concisely representing emerging patterns. Constrained emerging patterns were also defined in [10, 3], where they were termed extended emerging patterns. However, no study was made of their efficient mining nor of their effectiveness in classification.

Jumping emerging patterns (JEPs) are similar to version spaces [15]. Given a set of positive and a set of negative training 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 training set. In contrast, a jumping emerging pattern space is the set of all item patterns that each match (or are contained in) one or more (not necessarily every) positive instance and no negative instance in the set.

Work in [5] uses trees for mining of emerging patterns. The methodology is different, however, since the algorithm is not complete (i.e. it does not discover all emerging patterns that may exist within a dataset).

Our work in this paper examines the use of emerging patterns for eager classification. Work in [12] developed a lazy emerging pattern classifier. In a similar way, we could also adapt our results to the lazy context.

The CBA classifier [14] 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 our CEP classifier, which uses the low value of $\beta = 1$.

Pairwise classification was discussed in [7], 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 section 6.

8 Summary and Future Work

In this paper we have developed techniques for improving classification by emerging patterns. We identified a new class of pattern known as a constrained emerging pattern. This pattern allowed targeted specification of supports within each dataset. We then showed how pairwise classification dramatically improves the power of such patterns on multi class problems. The resulting classifier is superior to state of the art methods on a large number of datasets. Our algorithms for mining CEPs were also shown to be highly efficient in comparison to other applicable techniques. As part of future work, we plan to investigate

- Techniques for choosing the most interesting patterns within a collection of CEPs, based on their actual supports within D_p and D_n.
- Improving the sophistication of the scoring function used to determine the overall winner.

Acknowledgements: This work was supported in part by an Expertise Grant from the Victorian Partnership for Advanced Computing.

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.
2. C. L. Blake and P. M. Murphy. UCI Repository of Machine Learning Databases. www.ics.uci.edu/~mlearn/MLRepository.html.
3. G. Dong and J. Li. Efficient Mining of Emerging Patterns: Discovering Trends and Differences. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pages 43–52, 1999.
4. 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.
5. 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. 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. J. Furnkranz. Pairwise Classification as an Ensemble Technique. In *Proceedings of the Thirteenth European Conference on Machine Learning*, pages 97–110, 2002.
8. 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.
9. J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns Without Candidate Generation. In *Proceedings of the International Conference on Management of Data*, pages 1–12, 2000.
10. J. Li. *Mining Emerging Patterns to Construct Accurate and Efficient Classifiers.* PhD thesis, University of Melbourne, 2001.
11. 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.
12. 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.
13. J. Li and L. Wong. Emerging Patterns and Gene Expression Data. In *Proceedings of the Twelfth Workshop on Genome Informatics*, pages 3–13, 2001.
14. 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.
15. T. M. Mitchell. Generalization as Search. *Artificial Intelligence*, 18(2):203–226, 1982.
16. J. R. Quinlan. *C4.5 Programs for Machine Learning.* Morgan Kaufmann, 1993.
17. X. Zhang, G.Dong, and K. Ramamohanarao. Eploring 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.