Efficiently Mining Interesting Emerging Patterns

Hongjian Fan and Kotagiri Ramamohanarao

Dept. of CSSE, The University of Melbourne, Parkville, Vic 3052, Australia
{hfan,rao}@cs.mu.oz.au

Abstract. Emerging patterns (EPs) are itemsets whose supports change significantly from one class to another. It has been shown that they are very powerful distinguishable features and they are very useful for constructing accurate classifiers. Previous EP mining approaches often produce a large number of EPs, which makes it very difficult to choose interesting ones manually. Usually, a post-processing filter step is applied for selecting interesting EPs based on some interestingness measures.

In this paper, we first generalize the interestingness measures for EPs, including the minimum support, the minimum growth rate, the subset relationship between EPs and the correlation based on common statistical measures such as chi-squared value. We then develop an efficient algorithm for mining only those interesting EPs, where the chi-squared test is used as heuristic to prune the search space. The experimental results show that our algorithm maintains efficiency even at low supports on data that is large, dense and has high dimensionality. They also show that the heuristic is admissible, because only unimportant EPs with low supports are ignored. Our work based on EPs for classification confirms that the discovered interesting EPs are excellent candidates for building accurate classifiers.

Keywords: Emerging patterns, measures of interestingness, classification, data mining

1 Introduction

Classification is an important data mining problem. Given a training database of records, each tagged with a class label, the goal of classification is to build a concise model that can be used to predict the class label of future, unlabelled records. Many classification models have been proposed in the literature [14]. Recently a new type of knowledge patterns, called emerging patterns (EPs), was proposed by Dong and Li [4] for discovering distinctions inherently present between different classes of data. EPs are defined as multivariate features (i.e., itemsets) whose supports (or frequencies) change *significantly* from one class to another. The concept of emerging patterns is very suitable for serving as a classification model. By aggregating the differentiating power of EPs, the constructed classification systems [5,10,11,6,7] are usually more accurate than other existing state-of-the-art classifiers. The idea of emerging patterns is also applied in

G. Dong et al. (Eds.): WAIM 2003, LNCS 2762, pp. 189–201, 2003.

bioinformatics successfully, from the discovery of gene structure features to the classification of gene expression profiles [12,9].

A major difficulty involved in the use of EP is how to efficiently mine those EPs which are useful for classification, because it has been recognised that an EP mining algorithm can generate a large number of EPs, most of which are actually of no interest for modelling or classification purpose. Being able to mine only useful EPs is important, since it can save mining of many unnecessary EPs and identifying interesting ones from a huge number of EPs.

What makes emerging patterns interesting? The measures of interestingness are divided into objective measures - those that depend only on the structure of a pattern and the underlying data used in the discovery process, and the subjective measures - those that also depend on the class of users who examine the pattern [15]. We define interestingness of an EP in objective terms. An EP is interesting, if it (1) has minimum support; (2) has minimum growth rate; (3) has larger growth rate than its subset; (4) highly correlated according to common statistical measures such as chi-square value. The first condition ensures an EP is not noise by imposing a minimum coverage on the training dataset; the second requires an EP has sharp discriminating power; the third regards those "minimal" EPs as interesting, because if any subset of an EP has larger growth rate, the EP itself is not so useful for classification; generally speaking, the last states that an EP is interesting, if the distribution (namely, the supports in two contrasting classes) of its subset is *significantly* different from that of the EP itself, where the difference is measured by the χ^2-test [2]. Experiments show that the set of interesting EPs is orders of magnitude smaller than the set of general EPs. In the case that a user wants to use EPs for classification, the subjective measure of EPs can be defined as their usefulness. To evaluate objective interesting EPs against the subjective measure, we have built classifiers using those EPs. High accuracy on benchmark datasets from the UCI Machine Learning Repository [3] shows that mining EPs using our method can result in high quality EPs with the most differentiating power.

The task of mining EPs is very difficult for large, dense and high-dimensional datasets, because the number of patterns present in the datasets may be exponential in the worst case. What is worse, the Apriori anti-monotone property, which is very effective for pruning search space, does not apply to emerging pattern mining. It is because if a pattern with k items is not an EP, its super-pattern with $(k+1)$ or more items may or may not be an EP.

Recently, the merits of a pattern growth method such as FP-growth [8], have been recognized in the frequent pattern mining. We can use FP-growth to mine EPs: we first find frequent itemsets in one data class for a given support threshold, and then check the support of these itemsets against the other class. Itemsets satisfying the four interestingness measures are interesting EPs. There are several difficulties with this approach: (1) a very large number of frequent patterns will be generated when the support is low; (2) a lot of frequent patterns in one class turn out not to be EPs since they are also frequent in the other class; (3) it selects interesting EPs as post-analysis.

To overcome these difficulties, we propose Interesting Emerging Pattern Miner (iEPMiner) for efficiently extracting only the interesting emerging patterns. iEPMiner uses a tree structure to store the raw data. It recursively partitions the database into sub-database according to the patterns found and search for local patterns to assemble longer global one. iEPMiner operates directly on the data contained in the tree, i.e., no new nodes are inserted into the original tree and no nodes are removed from it during the mining process. The major operations of mining are counting and link adjusting, which are usually inexpensive.

The problem of mining EPs can be seen as to search through the power set of the set of all items for itemsets that are EPs. With low minimum settings on support and growth rate, the candidate interesting EPs embedded in a high-dimensional database are often too numerous to check efficiently. We push the interestingness measures into the pattern growth to reduce the search space. We also use the χ^2-test as heuristic to further prune the search space. The heuristic is admissible because (1) it greatly improves the efficiency of mining; (2) only EPs with the lowest supports are lost. Experiments show that iEPMiner achieves high efficiency on large high-dimensional database with low support and growth rate, and successfully mines the top 90% interesting EPs.

1.1 Related Work

Dong and Li [4] introduced the concept of emerging patterns and they also proposed the notion of borders as a means for concisely describing emerging patterns. They formalised the notion of set intervals, defined as collections S of sets that are interval closed - if X and Z are in S and Y is a set such that $X \subseteq Y \subseteq Z$, then Y is in S. The collection of emerging patterns discovered from different classes of data, which is typically very large, can be represented by borders, defined as the pair of the sets of the minimal itemsets and of the maximal ones, which are usually much smaller. A suite of algorithms, which manipulates only borders of two collections, were proposed for mining emerging patterns. However, they depend on border finding algorithms such as Max-Miner [1]. In fact, the task of mining maximal frequent patterns is very difficult, especially when the minimum support is low (e.g. 5% or even 0.1%). For example, for the UCI Connect-4 dataset, the Max-Miner, one of the most efficient previously known algorithm for finding maximal frequent itemsets, needs more than three hours when minimum support is 10%. Furthermore, the process of extracting the embodied EPs with supports and growth rates from the borders and selecting the interesting one is very time-consuming. In contrast, our algorithm mine interesting EPs directly from the raw data.

ConsEPMiner [16] mines EPs satisfying several constraints including growth-rate improvement constraint. It follows an Apriori level-wise, candidate generation-and-test approach. It is still not efficient when the minimum support is low. For the UCI Connect-4 dataset, ConsEPMiner needs about 6 hours when support is 3%. In comparison, our algorithm can finish in less than 10 minutes, with little loss of interesting patterns.

Recent work in [13] proposed to use "shadow patterns" to measure the interestingness of minimal JEPs. Shadow patterns are those immediate subsets of a minimal JEP. If the growth rates of these shadow patterns are on average around small numbers like 1 or 2, compared with the infinite growth rate of the JEP, it is regarded as *adversely interesting*, because the JEP is "unexpected" and the conflict may reveal some new insights into the correlation of the features. Their interestingness measure is a specific case of our correlation measure, since the level of adversity can be detected by χ^2-test. They do post-analysis of mined JEPs, while we push the interestingness measures into the mining process.

2 Interesting Measures of Emerging Patterns

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

Let I denote the set of all items in the encoding dataset D. A set X of items is also called an itemset, which is defined as a subset of I. We say any instance S contains an itemset X, if $X \subseteq S$. The support of an itemset X in a dataset D, $supp_D(X)$, is $count_D(X)/|D|$, where $count_D(X)$ is the number of instances in D containing X. Assume two data classes D_1 and D_2, the growth rate of an itemset X in favour of D_2 is defined as $GrowthRate(X) = GR(X) = supp_{D_2}(X)/supp_{D_1}(X)$ (where $GR(X) = 0$, if $supp_{D_2}(X) = supp_{D_1}(X) = 0$; $GR(X) = \infty$, if $supp_{D_2}(X) > 0$ and $supp_{D_1}(X) = 0$). An Emerging Pattern Y favouring D_2 is an itemset whose growth rate from D_1 to D_2 is at least $\rho(\rho > 1)$. The support of Y in D_2, denoted as $supp(Y)$, is called the support of the EP.

2.1 Interesting Emerging Patterns

We formally define the objective interestingness of an EP. An EP, X, is interesting, if

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. $\forall Y \subset X, GR(Y) < GR(X)$.
4. $|X| = 1$ or $|X| > 1 \wedge (\forall Y \subset X \wedge |Y| = |X| - 1 \wedge chi(X, Y) \geq \eta)$, where $\eta = 3.84$ is a minimum chi-value threshold and $chi(X, Y)$ is computed using the following contingency table [2].

	X	Y	$\sum row$
D_1	$count_{D_1}(X)$	$count_{D_1}(Y)$	$count_{D_1}(X) + count_{D_1}(Y)$
D_2	$count_{D_2}(X)$	$count_{D_2}(Y)$	$count_{D_2}(X) + count_{D_2}(Y)$
$\sum column$	$count_{D_1+D_2}(X)$	$count_{D_1+D_2}(Y)$	$count_{D_1+D_2}(X) + count_{D_1+D_2}(Y)$

The first condition ensures an EP has minimum coverage on the training dataset; the second requires an EP has sharp discriminating power; the third explores the subset relationship of EPs, i.e., interesting EPs are not "subsumed" by their subsets; the last states that for any immediate subset of an interesting EP with length more than 1, its support distribution in both classes are *significantly* different from that of the EP itself. One can use other statistical measures such as the entropy gain, the gini index and the correlation coefficient in place of chi-square value. The bigger the value, the more confident we are to say that their distributions are different. We choose 3.84 as the minimum chi-value threshold, since it gives us 95% confidence, which is enough in many real life applications. If a length-k EP's distribution is *significantly* different from that of any of its length-$(k-1)$ subsets, it shows that adding one item from length-$(k-1)$ subsets makes its behaviour on two classes quite different. It also means that those items which make up of the EP, are highly correlated.

We give an example to see how contingency tests are performed in the process of mining. Let $X = \{a, b, c\}, Y = \{a, b\}$. Suppose $|D_1| = |D_2| = 100$ and $count_{D_1}(Y) = 80$, $count_{D_2}(Y) = 60$, $count_{D_1}(X) = 60$, $count_{D_2}(X) = 35$, then we have the following observed contingency table (left). For each pair $(i, j) \in \{D_1, D_2\} \times \{X, Y\}$, we calculate the expectation under the assumption of independence:

$$E_{ij} = \frac{count_{D_1+D_2}(j) \times (count_i(X) + count_i(Y))}{count_{D_1+D_2}(X) + count_{D_1+D_2}(Y)}.$$

The results are shown in the following expected contingency table (right).

The observed contingency table

	Y	X	$\sum row$
D_1	80	60	140
D_2	85	35	120
$\sum column$	165	95	260

The expected contingency table

	Y	X	$\sum row$
D_1	89	51	140
D_2	76	44	120
$\sum column$	165	95	260

The chi-square value is the normalised deviation of observation from expectation; namely,

$$chi(X, Y) = \sum_{i \in \{D_1, D_2\}} \sum_{j \in \{X, Y\}} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}.$$

From the above two tables, the computed χ^2 value is 5.405. Since $\chi^2 \geq 5.02$ (at 97.5% significance level), we say that the distributions of X and Y are different with a confidence of 97.5%, which is higher than the minimum of 95%.

2.2 Chi-Squared Pruning Heuristic

Our tree based algorithm mines EPs in a pattern growth manner. How do we push the interestingness measures into mining? It is straightforward to push the measure 1 and 2 into the pattern growth (see next section for details). But it is hard to push the measure 3 and 4, because we may not have "seen" all

the subsets of the current pattern. A heuristic is proposed to prune as early as possible the search space, i.e., those patterns which are very likely turn out not to satisfy condition 3 or 4. The heuristic is based on the following lemma.

Lemma 1. *Let X, Y, Z be itemsets. $Y = X \cup \{i\}$, $Z = Y \cup \{j\}$, $S = X \cup \{j\}$, where i and j are items. If $chi(X, Y) < \eta$, $P(\{i\}|X)P(\{j\}|X) = P(\{i, j\}|X)$, and $\eta = 3.84$, then we have $chi(S, Z) < \eta$ with least 95% confidence.*

Proof. Since $\eta = 3.84$, $chi(X, Y) < \eta \Longleftrightarrow chi(X, X \cup \{i\}) < 3.84$. We say i is independent from X with at least 95% confidence. So we have $P(X \cup \{i\}) \approx P(X)P(\{i\})$. $P(\{i\}|X)P(\{j\}|X) = P(\{i, j\}|X) \Longleftrightarrow$

$$\frac{P(\{i\} \cup X)}{P(X)} * \frac{P(\{j\} \cup X)}{P(X)} = \frac{P(\{i, j\} \cup X)}{P(X)} \Longrightarrow P(\{i\}) * \frac{P(\{j\} \cup X)}{P(X)} \approx \frac{P(\{i, j\} \cup X)}{P(X)}.$$

So $P(X \cup \{i, j\}) \approx P(X \cup \{j\})P(\{i\})$, which means i is independent from $X \cup \{j\}$. Thus, we have $chi(X \cup \{j\}, X \cup \{j, i\}) < 3.84$ with at least 95% confidence.

The lemma has an assumption: $P(\{i\}|X)P(\{j\}|X) = P(\{i, j\}|X)$. Although it is not true for all the cases in real datasets, experiments show that for most cases we have $P(\{i\}|X)P(\{j\}|X) \approx P(\{i, j\}|X)$, which is good enough for mining interesting EPs. When $chi(X, Y) < \eta = 3.84$, from the lemma, Z definitely will not be interesting since it does not satisfy condition 4. Our mining method can stop growing Y immediately to avoid searching and generating unnecessary candidate patterns.

The χ^2-test ($chi()$ function) can be used as an effective heuristic for pruning search space. By pruning long patterns as soon as possible, we usually obtain a relatively small set of EPs. One pass over the set of EPs can select the interesting EPs according to the four interestingness measures. In contrast, if iEPMiner does not use the heuristic, it needs to search a huge space, which produces a lot of uninteresting patterns first and discards them later. Experiments show that the χ^2-test heuristic makes iEPMiner more efficient by an order of magnitude. We also investigate what patterns the heuristic search may lose. Detailed analysis over many datasets from the UCI Machine Learning Repository and high accuracy of the classifiers based on our mined interesting EPs confirm that it loses only unimportant EPs. So the χ^2-test pruning heuristic is admissible, although it is non-monotone.

3 Mining Interesting Emerging Patterns

3.1 Pattern Tree

Without loss of generality, we use lexicographic order as a partial order on the set of all items, denoted as $\prec$.

Definition 1. *A Pattern Tree (P-tree) is a tree structure defined below.*

1. *It consists of one root, a set of item prefix subtrees as the children of the root, and a header table.*

2. *Each node in the item prefix subtrees consists of four fields: item-name, $count_{D_1}$, $count_{D_2}$ and node-link, where item-name registers which item this node represents, $count_{D_1}$ registers the number of transactions in D_1 represented by the portion of the path reaching this node, $count_{D_2}$ registers such number in D_2, and node-link links to the next node in the P-tree carrying the same item or null if there is none.*
3. *Each entry in the header table consists of three fields: (1) item-name; (2) head of node-link, which points to the first node in the P-tree carrying the item; (3) $total_{D_1}$,the sum of all $count_{D_1}$ in the item's corresponding node-link; (4) $total_{D_2}$,the sum of all $count_{D_2}$ in such node-link.*
4. *The tree is ordered: if a node M is the parent of a node N, and item i and j appear in M and N respectively, then $i \prec j$.*

Note that nodes with the same item-name are linked in sequence via node-link, which facilitates tree traversal. Unlike the FP-tree [8], the P-tree is only traversable top-down (from root to leaves), i.e., there is no pointer from child to parent nodes. The construction of the P-tree can be found in [6]. The P-tree of the example dataset from Figure 1 is shown in Figure 2.

D_1					D_2				
a		c	d	e	a	b			
a							c		e
	b			e	a	b	c	d	
	b	c	d	e				d	e

Fig. 1. A dataset containing 2 classes as an example

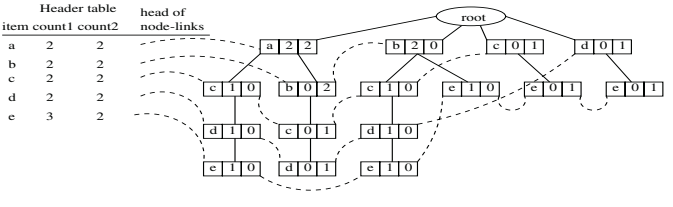


Fig. 2. The P-tree of the example dataset

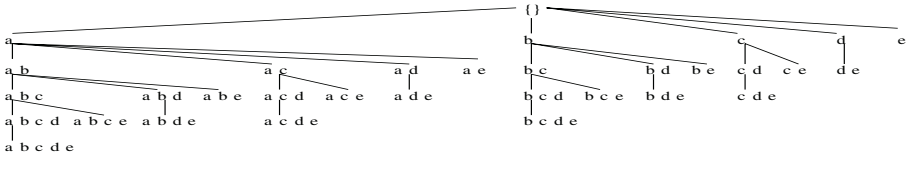


Fig. 3. A complete set enumeration tree over I, with items lexically ordered

3.2 Using P-Tree to Mine Interesting Emerging Patterns

We show the ideas of mining by using the tree shown in Figure 2. Let $\xi = 1$ be a minimum support threshold, and $\rho = 2$ a minimum growth rate threshold. Let us examine the mining process based on the constructed tree shown in Figure 2. Basically, we have to calculate the supports in both D_1 and D_2 for the power set

of $I = \{a, b, c, d, e\}$ and then check each itemset against the four interestingness measures. We use the set enumeration tree shown in Figure 3 as the conceptual framework to explore the itemset space. The itemsets are "generated" in the specific order: first visit the node, then visit the right and left subtree. Namely, the itemsets are considered in the following order:

- $\{e\}$
- $\{d\}$, $\{d, e\}$
- $\{c\}$, $\{c, e\}$, $\{c, d\}$, $\{c, d, e\}$
- $\{b\}$, $\{b, e\}$, $\{b, d\}$, $\{b, d, e\}$, $\{b, c\}$, $\{b, c, e\}$, $\{b, c, d\}$, $\{b, c, d, e\}$
- $\{a\}$, $\{a, e\}$, $\{a, d\}$, $\{a, d, e\}$, $\{a, c\}$, $\{a, c, e\}$, $\{a, c, d\}$, $\{a, c, d, e\}$, $\{a, b\}$, $\{a, b, e\}$, $\{a, b, d\}$, $\{a, b, d, e\}$, $\{a, b, c\}$, $\{a, b, c, e\}$, $\{a, b, c, d\}$, $\{a, b, c, d, e\}$

For e, we get its counts in both classes from the head table, denoted as [e:3; 2] (the two numbers after ":" indicate the supports in D_1 and D_2, respectively). {e} is not an EP since its growth rate $1.5 < \rho$.

For d, we have [d:2; 2]. {d} is not an EP. We try to grow {d} via concatenation of e with it. e's counts in both classes change from [e:3; 2] to [e:2; 1], when only those e co-occurring with d are counted. This can be done by going through d's node-links and visit those d's subtrees. We simply refer the process to recounting e under {d}, which is frequently used in the following. Note that the other two e are not counted since they are not in such subtrees. Then we get [d:2; 2, e:2; 1], where $\{d, e\}$ is an EP of growth rate 2.

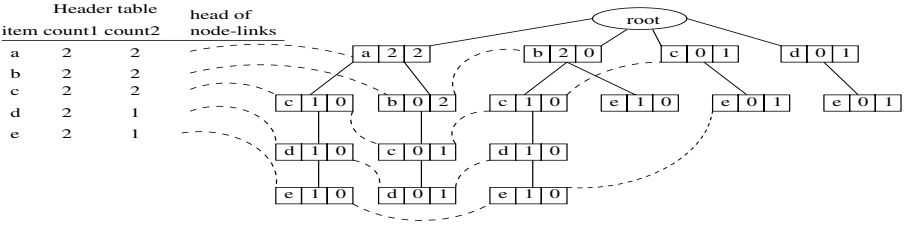


Fig. 4. The P-tree after adjusting the node-links and counts of d and e under c

For c, we have [c:2; 2]. {c} is not an EP. Now we have e and d to concatenate with c. The P-tree after the node-links and counts of e and d are adjusted is shown in Figure 4. We try e first. After recounting e under {c}, we obtain [c:2; 2, e:2; 1], where $\{c, e\}$ is an EP of growth rate 2. We then try d. After recounting d under {c}, we obtain [c4:2; 2, d:2; 1], where $\{c, d\}$ is an EP of growth rate 2. Because $\{c, d\}$ has supports in D_1 and D_2 quite different from {c}, it may produce interesting patterns to further grow $\{c, d\}$ by adding e. After recounting e under {c,d}[1], we obtain [c:2; 2, d:2; 1, e:2; 0], where $\{c, d, e\}$ is an EP of infinite

[1] Since only those d under c are linked by its node-links, it is easy to go through d's node-links looking for e.

growth rate. Usually, an EP with infinite growth rate is called a JEP(Jumping EP).

For b, we have $[b{:}2; 2]$. $\{b\}$ is not an EP. Now we have e, d and c to concatenate with b. We try e first. After recounting e under $\{b\}$, we obtain $[b{:}2; 2, e{:}2; 0]$, where $\{b, e\}$ is a JEP. We try d next. After recounting d under $\{b\}$, we obtain $[b{:}2; 2, d{:}1; 1]$. Because the support distributions of $\{b, d\}$ and $\{b\}$ are the same, it is very unlikely that we can get interesting EPs by further growing $\{b, d\}$. In fact, $\{b, d, e\}$ with support counts 1 and 0 in D_1 and D_2, is not interesting since its subset $\{b, e\}$ is also a JEP. It can be seen that our chi-squared heuristic effectively prunes a lot of uninteresting patterns from consideration. We then try c. After recounting c under $\{b\}$, we obtain $[b{:}2; 2, c{:}1; 1]$. For the same reason, we do not further grow $\{b, c\}$.

For a, we have $[a{:}2; 2]$. $\{a\}$ is not an EP. Now we have e, d, c and b to concatenate with a. We try e first. After recounting e under $\{a\}$, we obtain $[a{:}2; 2, e{:}1; 0]$, where $\{a, e\}$ is a JEP. We try d next. After recounting d under $\{a\}$, we obtain $[a{:}2; 2, d{:}1; 1]$. For the above reason, we do not further grow $\{a, d\}$. We then try c. After recounting c under $\{a\}$, we obtain $[a{:}2; 2, c{:}1; 1]$. Again we do not further grow $\{a, c\}$. Lastly, we try b. After recounting b under $\{a\}$, we obtain $[a{:}2; 2, b{:}0; 2]$, where $\{a, b\}$ is a JEP. We do not further grow a JEP, since supersets of a JEP is not interesting.

```
;; assume I = {1, ..., N}
;; and 1 ≺ ... ≺ N
Procedure iEP-Miner(root) {
  for i = N downto 1 do {
    β = {i};
    if is-iEP(β), then output β;
    mine-subtree(β);
  }
  prune uninteresting EPs;
}

Procedure mine-subtree(β) {
  Let k be the last item of β;
  for all items which appear in k's subtrees,
    adjust their node-links and accumulate counts;
  for j = N downto k+1 do {
    γ = β ∪ j;
    if is-iEP(γ), then output it;
    if chi(γ, β) ≥ η, mine-subtree(γ);
  }
}
```

Fig. 5. iEP-Miner pseudo-code

3.3 iEP-Miner

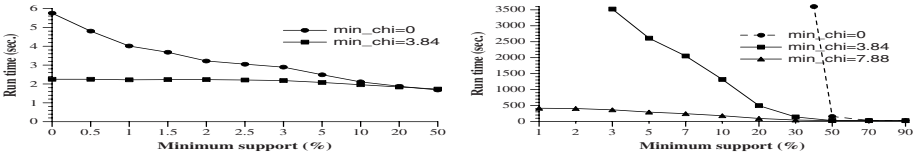
The high-level description of iEP-Miner is given in Figure 5. The main procedure iEP-Miner takes the root of the P-tree as input and performs the mining solely in the P-tree. The procedure mine-subtree() is called recursively. It always tries to grow the current pattern β by adding a new item. The function is-iEP() checks whether an itemset satisfies the interestingness measure 1, 2 and 4. The chi-squared pruning heuristic, the test "$chi(\gamma, \beta) \geq \eta$", is used to prune a huge number of patterns which are definitely uninteresting. The set of the generated candidate interesting EPs is relatively small, and one pass over the set can filter out those which does not satisfies the interestingness measure 3. The final set is our defined interesting EPs.

4 Performance Study

We now report a performance evaluation of iEP-Miner. We carried out experiments on many datasets from the UCI Machine Learning Repository, and all of them exhibited significant improvement in performance. For lack of space, we only present the results on the following large, high-dimensional datasets.

Dataset	Records	Avg. Record Width	No. of Binary items
adult	45,225	15	154
connect-4[2]	61,108	43	128
mushroom	8124	23	121

All experiments were conducted on a Dell PowerEdge 2500 (Dual P3 1GHz CPU, 2G RAM) running Solaris 8/x86, shared by many users of the University of Melbourne.



(a) Mushroom ($\rho = 1000$) (b) Connect-4 ($\rho = 2$)

Fig. 6. Scalability with support threshold

The interestingness of emerging patterns is determined by three parameters, where ξ is a minimum support threshold, ρ is a minimum growth rate threshold and η is a minimum chi-square value threshold. The scalability of iEP-Miner with support threshold is shown in Figure 6.

Table 1. The effectiveness of the chi-squared pruning heuristic

# of EP searched	adult	connect4	mushroom
without heuristic	6,977,123	16,525,078	4,373,265
with heuristic	191,765	369,443	89,624
Ratio	36.4	44.7	48.8

To show the effectiveness of the chi-squared pruning heuristic, we investigate how many candidate EPs we need to "look at" before interesting EPs are generated. The results are shown in Table 1. It can be seen that a huge amount of search space is pruned because our heuristic stops early growing many unpromising branches.

Table 2. Comparison between general EPs and interesting EPs

ADULT ($\xi = 0\%, \rho = 10000$, maximum length = 11)

	0-1%	1-2%	2-3%	3-4%	4-5%	5-6%	6-8%	8-100%	0-100%
all EPs	10,490,845	4,366	963	255	126	51	16	0	10,496,622
all iEPs	10,072	92	20	9	6	3	4	0	10,206
mined iEPs	9,239	83	19	9	6	3	4	0	9,363
Ratio	1,041.6	47.5	48.2	28.3	21	17	4	1	1,028.5
missing iEPs	8.3%	9.8%	5%	0	0	0	0	0	8.3%

CONNECT-4 ($\xi = 1\%, \rho = 2$, maximum length = 10)

	1-2%	2-4%	4-6%	6-10%	10-40%	40-100%	0-100%
all EPs	13,837,899	5,938,079	1,372,383	729,788	242,461	0	22,120,610
all iEPs	2,064	2,130	747	487	407	0	5,835
mined iEPs	1,940	1,993	712	487	407	0	5,539
Ratio	6704.4	2787.8	1837.2	1498.5	595.7	1	3791
missing iEPs	6%	6.4%	4.7%	0	0	0	5.1%

MUSHROOM ($\xi = 0\%, \rho = 10000$, maximum length = 6)

	0-2%	2-4%	4-7%	7-10%	10-30%	30-70%	70-100%	0-100%
all EPs	8,113,592	312,120	123,256	18,861	44,480	2,015	0	8,614,333
all iEPs	1,032	546	360	175	416	72	0	2,606
mined iEPs	1,002	536	360	175	416	72	0	2,526
Ratio	7862	571.6	342.4	107.8	106.9	30	1	3312
missing iEPs	2.9%	1.8%	0	0	0	0	0	3.1%

In order to have some ideas of what proportion of EPs are interesting, we compare the set of "all EPs"(satisfying the support and growth rate threshold only, and their maximum length is no more than the maximum length of all interesting EPs), "all iEPs"(satisfying the four interestingness measures) and "mined iEPs"(satisfying the four interestingness measures, but not complete due to the heuristic) in terms of their distributions in support intervals. The results are shown in Table 2. The ratio is the number of "all EPs" in the specified interval divided by that of "all iEPs". The last row gives the percent of missing iEPs over "all iEPs" due to heuristic searching. We highlight some interesting points:

- The set of all interesting EPs is 1000-3000 times smaller than the set of all general EPs.
- The ratios decreases from left to right, which means that our interestingness measures eliminate a large number of EPs with low supports, while tend to keep EPs with higher supports. This is desirable and reasonable, since EPs with higher supports are definitely more preferable for classification given the same growth rate. On the other hand, an EPs with high support does not necessarily mean that it is useful, since its subset may have higher support.
- We stress that the set of our mined interesting EPs is very close to the set of true interesting EPs: they are *exactly the same* at high support; only at

very low support, some interesting EPs are ignored. The chi-squared pruning heuristic is very effective since the top 90% interesting EPs are discovered by our algorithm.

5 Conclusions

In this paper, we have proposed four objective interestingness measures for EPs and developed an efficient algorithm, iEPMiner, for mining only the interesting EPs based on a tree data structure. The chi-squared pruning heuristic is used to mine EPs by growing only promising branches. This achieved considerable performance gains: the heuristic makes iEPMiner orders of magnitude faster. Although it gives up the completeness of interesting EPs, the heuristic always discovers the top 90% interesting EPs, which are sufficient to build high accurate classifiers in many real life applications.

References

1. R.J. Bayardo. Efficiently mining long patterns from databases. In *Proc. ACM-SIGMOD'98*, pages 85–93, Seattle, WA, USA, June 1998.
2. Robert M. Bethea, Benjamin S. Duran, and Thomas L. Boullion. *Statistical methods for engineers and scientists.* New York : M. Dekker, 1995.
3. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
4. G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proc. ACM-SIGKDD'99*, pages 43–52, San Diego, CA, Aug 1999.
5. G. Dong, X. Zhang, L. Wong, and J. Li. Classification by aggregating emerging patterns. In *Proc. the 2nd Intl. Conf. on Discovery Science,* pages 30–42, Tokyo.
6. H. Fan and K. Ramamohanarao. An efficient single-scan algorithm for mining essential jumping emerging patterns for classification. In *Proc. 2002 Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD'02)*, Taipei, Taiwan.
7. H. Fan and K. Ramamohanarao. A bayesian approach to use emerging patterns for classification. In *Proc. 14th Australasian Database Conference (ADC2003).*
8. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. ACM-SIGMOD'00*, pages 1–12, Dallas, TX, USA, May 2000.
9. L. Wong J. Li. Identifying good diagnostic genes or genes groups from gene expression data by using the concept of emerging patterns. *Bioinformatics*, 18(5):725–734, 2002.
10. J. Li, G. Dong, and K.Ramamohanarao. Making use of the most expressive jumping emerging patterns for classification. *Knowledge and Information Systems*, 3(2):131–145, 2001.
11. J. Li, G. Dong, K. Ramamohanarao, and L. Wong. DeEPs: A new instance-based discovery and classification system. *Machine Learning*, To appear.
12. J. Li, H. Liu, J. R. Downing, and L. Wong A. Yeoh. Simple rules underlying gene expression profiles of more than six subtypes of acute lymphoblastic leukemia (all) patients. *Bioinformatics*, 19(1):71–78, 2003.
13. J. Li and L. Wong. Geography of differences between two classes of data. In *Proc. 6th European Conf. on Principles of Data Mining and Knowledge Discovery*, pages 325–337, Helsinki, Finland, Aug 2002.

14. T. Lim, W. Loh, and Y. Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40:203–228, 2000.
15. A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):970–974, 1996.
16. X. Zhang, G. Dong, and K.Ramamohanarao. Exploring constraints to efficiently mine emerging patterns from large high-dimensional datasets. In *Proc. ACM-SIGKDD'00*, pages 310–314, Boston, USA, Aug 2000.