Asian Journal of Information Technology 4 (11): 1086-1092, 2005

Stock-management-based Expectation Pattern Discovery

[1]Shujie Yang, [2]Xiaofang You, [2]Feng Chen and [2,3]Shichao Zhang
[1]Capital Normal University, Beijing, PR China
[2]Department of Computer, Guangxi Normal University, PR China
[3]Faculty of Information Technology, University of Technology Sydney, Australia

Abstract: We design a new yet efficient strategy for identifying against-expectation patterns in databases. An against-expectation pattern is either an itemset that its support is out of a certain neighbor of its expected value, referred to an against-expectation itemset or an association rule generating by an against-expectation itemset, referred to an against-expectation rule. The techniques for mining against-expectation patterns are previously undeveloped. Present algorithm for identifying against-expectation patterns is based on the nearest- neighbor graph and correlation analysis techniques. We experimentally evaluate our algorithms and demonstrate that our approach is efficient and promising.

Key words: Exception, against-expectation pattern, nearest-neighbor graph, correlation analysis

INTRODUCTION

An against-expectation pattern is either an itemset that its support is out of a certain neighbor of its expected value, referred to an against-expectation itemset; or an association rule generating by an against-expectation itemset, referred to an against-expectation rule. The techniques for mining against-expectation patterns are previously undeveloped. This study studies the issue of mining against-expectation patterns in databases.

Using extant frequent-pattern-discovery algorithms to a market basket dataset, apple can be identified as a frequent pattern (itemset) even though its support (=200) much lesser than its expected sales (=300) because apple is an everyday fruit and is frequently purchased everyday. Comparing to apple, cashew is an expensive fruit and is rarely purchased. In the market basket dataset, cashew cannot be discovered as a frequent pattern of interest, even though its support (=20) much greater than its expected sales (=5). From an applied context, while frequent pattern apple is commonsense, the purchasing increase of cashew is desired in marketing decision-making that is just the against-expectation pattern to be mined in this paper. Similarly, the purchasing decrease of apple is also an against-expectation pattern desired. These against-expectation patterns assist in evaluating the buying amount of the products in next time-lag.

Against-expectation patterns are distinct from frequent patterns (or association rules) because (1) they may be pruned in identifying frequent patterns (or association rules), (2) they can be deviated from frequent patterns (or association rules) and (3) against-expectation patterns are hidden information, whereas traditional frequent patterns (or association rules) are relatively plain information.

To our knowledge, likely related achievements mainly include unexpected pattern[1,2], exceptional pattern[3-6] and negative association rules[7,8]. The first and second ones are referred to exceptions of rules, also known as surprising patterns. The third ones are referred to a negative relation between two itemsets.

An exception of a rule is defined as a deviational pattern to a well-known fact, exhibits unexpectedness. For example, while bird(x)→flies(x) is a well-known fact, mining exceptional rules is to find patterns such as bird(x), penguin(x)→ ~flies(x). The negative relation really implies a negative rule between the two itemsets, including association rules of forms A→ ~B, ~A→ B and ~A→ ~B, which indicate negative associations between itemsets A and B[7,8].

Hence, against-expectation patterns are also different from the unexpected pattern, exceptional pattern and negative association rules. Therefore, against-expectation patterns should be regarded as a new kind of patterns.

AGAINST-EXPECTATION PATTERNS

This presents some basic concepts and describes the issue of mining against-expectation patterns in databases.

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of n distinct literals called items. For a given dataset D over I, we can represent D as follows.

In Table 1, T_j is the identifier of transactions in D; a_{jk} is the state of item i_k in transaction T_j, $i_k = 1$ when i_k occurs in T_j and $i_k = 0$ when i_k does not occur in T_j and f_k is the frequency of i_k in D, i.e. the sum of kth column a_{*k}.

In marketing, data marketers must know the sales expectation of each product that is used to determine how

Corresponding Author: Shujie Yang, Capital Normal University, Beijing, PR China

Table 1: A dataset D over I

TID	i_1	i_2	...	i_n
T_1	a_{11}	a_{12}	...	a_{1n}
T_2	a_{21}	a_{22}	...	a_{2n}
...	...	...	...	...
T_m	a_{m1}	a_{m2}	...	a_{mn}
Frequency	f_1	f_2	...	f_n

Table 2: The expectation of items

	i_1	i_2	...	i_n
Expectation	e_1	e_2	...	e_n

many products should be bought in each month (or a certain time-lag). Therefore, for the set of items I, we have the expectations of the supports of items(Table 2).

In Table 2, e_k is the expected frequency of i_k in D. For the example in Section 1, expectation(apple)=300, support(apple)=200, expectation(cashew)=5 and support(cashew)=20. Therefore,

$$\text{support(cashew)-expectation(cashew)} = 20-5 = 15$$

Is three times of its expectation. Hence, cashew is an against-expectation pattern. However,

$$\text{expectation(apple)-support(apple)} = 100$$

e.g. the purchasing decrease is a third of the expectation, we also refer 'apple' as an interesting against-expectation pattern.

Against-expectation patterns are defined as either those itemsets that their supports are out of an e-neighbour of their expected values, referred to against-expectation itemsets; or those rules that are interactions within against-expectation itemsets, referred to against-expectation rules. Formally, an against-expectation itemsets X has its support out of the e-neighbour of its expectation, i.e.

$$|\text{support}(X)-\text{expectation}(X)| > e$$

Where, e is a user-specified positive value. An against- expectation rule is of the form:

$$X \rightarrow Y$$

That is the interaction between the against-expectation itemsets X and Y. For example, apple $\rightarrow$ cashew can be an against-expectation rule between the above two against-expectation itemsets.

We represent and classify against-expectation patterns into three kinds: increment patterns, decrement patterns and negative associations.

- An increment pattern is either an itemset X that its actual support is greater than its expected value, e.g., support(X) -expectation(X) > e (e is a user- specified positive value), referred to an increment itemset; or a rule that is an interaction within increment itemsets, referred to an increment rule.
- An decrement pattern is either an itemset X that its actual support is less than its expected value, e.g., support(X) -expectation(X) < -e (e is a user-specified positive value), referred to a decrement itemset; or a rule that is an interaction within decrement itemsets, referred to a decrement rule.
- A negative association is a rule that its antecedent and action belong to different against-expectation itemsets, i.e. either (1) its antecedent is an increment itemset and its action is a decrement itemset, or (2) its antecedent is a decrement itemset and its action is an increment itemset.

Mining against-expectation patterns is actually a challenging issue because it is very different from those problems faced by discovering frequent patterns (or association rules). Because against- expectation patterns can be hidden in both frequent and infrequent itemsets (with lower frequency), traditional pruning techniques are certainly inefficiency for identifying against-expectation patterns. This indicates that we must exploit alternative strategies to; (a) confront an exponential search space consisting of all possible itemsets, frequent and infrequent in a database; (b) detect which of itemsets can generate against-expectation patterns; (c) which of against-expectation patterns are really useful to applications; and (d) measure the interestingness of against-expectation patterns.

One may note that the expectation can be expensive and dynamic. For a new company, its heads must take efforts on estimating the expectation by analyzing the environment and possible customers. For an old company, we can take the support of items in last time-lag's database as their expectations so as to check the support of items and identify the against- expectation patterns in this time-lag's database, or to predict the support of items and the against-expectation patterns in next time-lag. In the following, a time-lag is a month for simplifying the description.

ALGORITHM DESCRIPTION

Present algorithm for mining against-expectation patterns mainly includes; (1) generating a set of against-expectation items by preprocessing; and (2) identifying interactions within these against-expectation items based on the Nearest-Neighbor Graph and the Correlation Analysis techniques.

Before drafting our algorithm, we illustrate our ideas using an example as follows.

Example 1: Suppose we have two market basket datasets from a grocery store in January (Table 3) and February (Table 4), respectively. Let us focus on the purchase of products. Let min_supp = 33.3% and min_copnf = 70%.

From the dataset DF in Table 4, we can use extant mining algorithms to get association rules as follows.

supp (toothbrush → toothpaste) = 33.3%
conf (toothbrush → toothpaste) = 100%
supp (toothbrush → shampoo) = 33.3%
conf (toothbrush → shampoo) = 100%

Different from these rules, we mine against-expectation patterns in DF by referencing the dataset DJ in Table 3. The idea is briefly drafted as follows. Let's compare the support and increment of items apple and cashew as listed in Table 5.

Obviously no matter when is Jan. or Feb., apple is always frequent and cashew is not. However, the increment of cashew is far higher than that of apple. Therefore, our algorithm is designed for identifyin cashew with its change trend as an against-expectation pattern. This kind of against-expectation itemsets cannot be identified using extant association rule mining algorithms.

Finding against-expectation itemsets with square deviation: As mentioned above, confirming the expectation is not an easy thing for them of the supports of items are often expensive and dynamic. So the first key step in the process of finding against-expectation itemsets is to how to make sure the expecatations. Associated with real world, we decide to use store to denote expectation, because store must be confirmed according to previous sale record, market prediction and experts suggestions and represent expected sale of this item in next time-lag. But we think only one or two stores is not enough. What we really want is a trend curve about store so as to

Table 3: J: Transactions in january

Toothbrush, toothpaste
Bread, jam
Cashew
Apple, banana
Toothbrush, toothpaste, bread
Apple, cola, shampoo
Apple, banana, shampoo
Bread, jam, apple
Apple
Apple, banana, cola

Table 4: F: Transactions in february

Toothbrush, toothpaste, bread, jam, shampoo
Bread, cashew, apple, shampoo, jam
Cashew, shampoo, bread, jam
Apple, banana, cola, shampoo
Toothbrush, toothpaste, shampoo
Bread, jam, shampoo
Toothbrush, toothpaste, cola, shampoo
Apple, shampoo, bread, jam
Apple, banana, shampoo
Cashew, apple, banana, cola, shampoo
Toothbrush, toothpaste, apple, shampoo
Bread, jam, apple, shampoo

Table 5: The support and increment of apple with cashew

	Jan's supp	Feb's supp	Increment
Apple(%)	50	58.3	16.7
Cashew(%)	10	25.0	300.0

whether an item is on earth against-expectation item or not. So in this sub-section, an algorithm base on square deviation is designed for seeking against-expectation itemsets.

Definition 1: Let S_{ij} be store of i-th item at j-th time-lag, where $1 \le i \le m$, $1 \le j \le n$. So its math expectation is:

$$E(S_i) = \frac{S_{i1} + S_{i2} \ldots + S_{in}}{n}$$

and its square deviation is:

$$p(S_i) = \sum_{j=1}^{n} (S_{ij} - E(S_i))^2$$

and let prop be threshold. Then against-expectation items are those satisfying:

$$p(S_i) \ge prop$$

Based on the above, our SD algorithm is described in Fig. 1.

Identifying interactions with in against-expectation items: Only finding against-expectation items is not enough, our goal is also to identify interactions within them. So this employs two algorithms to get this goal.

```
Input  Stock, Prop
Output MID
(1)  for  each MID in a dataset
(2)   begin  j-MID;
(3)          calculate E(Xj)□p(Xj);
(4)          if (p(Xj)>=Prop)
(5)                return MID;
(6)  end
```

Fig. 1: SD algorithm finding against-expectation merchandise

The nearest neighbor graph: Let $V = \{v_1, v_2, ...v_n\}$ be a set of points in R^1. The nearest neighbor of v_i is a point v_j, $j \neq i$, with minimum Euclidean distance from v_i. To make the nearest neighbor unique we choose the point v_j with maximum index in case of ties and denote it by $nn(v_i)$. For any v, we define the directed edge $e(v) = \langle v, nn(v)$. The nearest neighbor graph of V, denoted by NNG(V), is directed graph $\langle V,E \rangle$, where $E = \{e(v) \mid v \in V\}$[9,10].

We introduce an example after learning what the nearest neighbor graph. Through this, you can see why it is suitable for settling our problem.

Example 2: Suppose shirt and shoes are both against-expectation items and so are other items listed below too.
shirt: business suit, tie, shoes, shoes, T-shirt, belt, tie bar and socks.
shoes: business suit, socks.

The two lists above represent items having influence on shirt and shoes, respectively. And their influence extents are organized with down trend. Dealing with this problem like this can make users know not only interactions between against-expectation items, but also what extant their influences are.

Let shirt $\in$ V, shoes $\in$ V. Then a list corresponding to any of them construct a nearest neighbor list in NNG(V) and all lists NNG(V). So according to this analysis, we have definition 2 and 3 about relative bargaining quantity below.

Definition 2: Suppose $A = \{a_1, a_2,...a_n\}$, $B = \{b_1, b_2,...b_n\}$. Let A and B denote two goods, a_i and b_i sales in I th time-lag. Then A^* and B^* denote sale quantity increments:

$$A^* = \{a_2 - a_1, a_3 - a_2, ..., a_n - a_{n-1}\}$$
$$B^* = \{b_2 - b_1, b_3 - b_2, ..., b_n - b_{n-1}\}$$

Let $Q(A,B) = \{q_1, q_2,...q_{n-1}\}$, where $q_i = \frac{a_{i+1} - a_i}{b_{i+1} - b_i}$. We call Q(A,B) as the relative bargaining quantity of A and B. And interaction extant between A and B, or distance is denoted by:

$$D(A,B) = \sum_{i=1}^{n-1} (q_i - E(Q(A,B)))^2$$

We suppose against-expectation item represents node in NNG(V) and weight of edge D(A,B). Then the problem seeking interactions between items converts into that seeking the nearest neighbors of an item. After finishing the NNG(V), a threshold is given to confirm the number of nearest neighbors (Fig. 2).

```
Input     Sale table, k-threshold
Output    MID neighbors
(1)   SaleTable → Stage table
(2)   for each MID from Fig. 1
(3)   begin for each MID' from Fig. 1
(4)       begin calculate X(MID, MID')□
(5)             calculate d(MID, MID')□
(6)             if(d(MID, MID')>=k-threshold)
(7)                   MID neighbors ← MID';
(8)             end
(9)   end
```

Fig. 2: Algorithm NN, k-nearest neighbor algorithm

Finally, this algorithm will give users a serial of lists each of which is corresponding to each against-expectation item and organized with down trend.
The process of this algorithm is as follows:

Correlation analysis: Different from the nearest neighbor graph, correlation analysis can not only find interactions between against-expectation items, but also find that these interactions are positive or negative.

For explaining our algorithm, there has necessary to introduce some basic concepts about correlation analysis. The below contexts can be found in any book concerning correlation analysis (Fig. 3).

Let $X = \{x_1, x_2,...x_n\}$ be an itemset and each x_i an against-expectation item. Then P(X) denotes the probability that X occurs in a transaction and $P(\overline{X})$ the probability that $\overline{X}$ dose not occur in a transaction. So $P(X) = 1-P(\overline{X})$. And $P(X \cup Y)$ denotes the probability that X and Y occur in a transaction simultaneously. If $P(X \cup Y) \neq P(X)P(Y)$, then X and Y is dependent, otherwise, independent. Correlation itemset is such an itemset each pair of items in which is correlated with each other[11].

Theorem 1: If each item in itemset S is correlated, then all supersets of S are correlated, i.e. correlation is upward closed.

Proof: Let X and Y be correlated, but X, Y and Z are independent one another. Then:

$$\begin{aligned} & P(X \cup Y) \\ &= P(X \cup Y \cup Z) + P(X \cup Y \cup \overline{Z}) \\ &= P(X)P(Y)P(Z) + P(X)P(Y)P(\overline{Z}) \\ &= P(X)P(Y) \end{aligned}$$

So X and Y are independent. This result and the hypothesis conflict, so theorem is right.

According to theorem1, we decide to only seek minimum correlation itemsets for enhancing efficiency and reducing time cost by set-enumeration tree[12]. And the formula to make sure whether an itemset is positive or negative is as follows:

$$Corr(X) = \frac{P(x_1 \cup x_2 \cup ... \cup x_n)}{P(x_1) \times P(x_2) \times ... \times p(x_n)}$$

If Corr (X)>1, items in X are positive correlated; if Corr (X)<1, items in X are negative correlated; if Corr (X) = 1, items in X are independent. But in fact, the last situation is hardly to happen in real world. So we give a new threshold proRange and modify the judgment criterions:

Items in X are independent if X satisfies:

$$|1\text{-pro range}| \le Corr(X) \le |1 + \text{pro range}|$$

Items in X are positive correlated if X satisfies:

$$Corr(X) > |1 + \text{prorange}|$$

Items in X are negative correlated if X satisfies:

$$Corr(X) < |1 - \text{prorange}|$$

The process of this algorithm is as follows:

Input Transaction
Output Pos corr, NegCorr
(1) **Construct** set-enumeration tree for MID from Fig. 1;
(2) **Scan** each node in set-enumeration tree
(3) **if** (Corr>(1+Prorange))
PosCorr←node;
(4) **if** (Corr<(1-Prorange))
NegCorr←node
(5) **if** (|1-ProRange|<=Corr<=|1+Prorange|)
pruning node

Fig. 3: Algorithm CA: correlation analysis algorithm

In addition to this measure, chi-square is another way for identifying correlation of itemset and its formula is below:

$$\chi^2 = \sum_{r \in R} \frac{(O(r) - E[r])^2}{E[r]}$$

Where,

$$R = \{i_1, \bar{i}_1\} \times \{i_2, \bar{i}_2\} \times ... \times \{i_k, \bar{i}_k\}, \quad r = r_1, r_2, ..., r_k \in R$$

R is possible count of all frequent itemsets occurring and r is a count of a item occurring singly. O(r) denotes the actual count of r occurring and E(r) the expected count of r occurring.

Particularly, the formula for measuring correlation of two items is as follows:

$$\chi^2 = \frac{(E[XY] - O(XY))^2}{E[XY]} + \frac{(E[X\bar{Y}] - O(X\bar{Y}))^2}{E[X\bar{Y}]} + \frac{(E[\bar{X}Y] - O(\bar{X}Y))^2}{E[\bar{X}Y]} + \frac{(E[\bar{X}\bar{Y}] - O(\bar{X}\bar{Y}))^2}{E[\bar{X}\bar{Y}]}$$

Although these two measures are both used for identifying correlated itemsets, but for us, the former is better than the latter, because it can distinguish positive itemsets from negative ones. So here, we employ the former measure.

EXPERIMENTS AND ANALYSIS

Experiments: To evaluate our algorithm, we have conducted extensive experiments on a DELL Workstation PWS650 with 2G main memory, 2.6G CPU and WINDOWS 2000.

The experiment have made on a synthetic database that includes 100 transactions consisting of 15 items. We not only evaluate our algorithms on this database, but also evaluate Apriori on it and make a comparison between them.

Below is comparison aiming at Apriori's three drawbacks.

- Generating false association rules: In this experiment, the result of Apriori is:

$$supp(8 \rightarrow 9) = supp(10 \rightarrow 13) = 60\%$$
$$conf(8 \rightarrow 9) = conf(10 \rightarrow 13) = 100\%$$

while k-nearest neighbor graph get: (Table 6)
8: 3, 4, 6, 13,
9: 12, 13,
10: 13,
13: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,

Table 6: K-nearest neighbor

MID	Nearest neighbor	MID	Nearest neighbor
0	5,7,13	8	3,4,6,13
1	5,7,11,13	9	12,13
2	5,13	10	13
3	8,5,6,4,13	11	1,13,14
4	6,5,3,8,13	12	9,13
5	0,1,2,3,4,7,13	13	0,1,2,3,4,5, 6,7, 8,9,10,11,12
6	4,3,8,13	14	11
7	0,1,5,13		

Table 7: Correlation analysis

Positive correlated itemsets	Negative correlated itemsets
(0,1),(0,4),(0,8),(0,10),(1,4)	(0,3),(0,5),(0,11),(1,2),(1,3)
(1,8),(1,10),(1,14),(2,3),(2,5)	(1,5),(1,6),(1,11),(1,12),(2,4)
(2,10),(2,13),(2,14),(3,5),(3,6)	(2,8),(2,11),(2,12),(3,4),(3,8)
(3,11),(3,12),(4,8),(4,10),(4,14)	(4,5),(4,6),(4,11),(5,6),(5,7)
(5,10),(5,14),(6,7),(6,11),(6,12)	(5,8),(5,12),(6,10),(6,14)(7,14)
(7,12),(8,9),(8,12),(10,13)	(8,11),(9,11),(10,11),(10,12)
(12,13),(0,7,9),(0,7,13),(0,9,13)	(12,14)
(5,9,13),(6,8,13),(6,9,13) (9,10,14)	

Table 8: Apriori algorithm

1-freq emsets	2-freq itemsets	3-freq itemsets
3,7,8,9,10,13	(8,9),(8,13),(9,10)	(8,9,13)
	(9,13),(10,13)	(9,10,13)

It can be seen easily that 8 toothbrush is closest to 3 toothpaste and has nothing about 9 bread. 10 and 13 are beer and milk respectively and their result is Corr(8,9)=1.2<Corr(10,13)=1.4. So toothbrush and bread is a rule.

- Generating abundant rules: Apriori gets supp (9→13) = 73%. But it is not useful for decision makers because of closed correlation between them. While k-nearest neighbor graph reveals that they are not very closed and correlation analysis sees that they are not independent.

- Missing some rules:

Case 1: Two algorithms both demonstrate that tie is correlated with business suit, perfume and cigarette, but on the contrary, in Apriori, it cannot be frequent given minsupp = 0.4. It is not difficult to understand that tie is supposed to affect the sale of business suit, which is justified. Although tie can be found with Apriori just reducing the threshold but in the meantime, more and more abundant rules generates too.

In addition, correlation analysis also found: (Table 7)
5: positive 2, 3, 10, 14, (9, 13)
negative 1, 0, 4

That is, diaper have positive correlation with 2 cigarette, 3 toothpaste, 10 beer, 14 makeup, 9 and 13 bread and milk and negative correlation with 0 tie and 4 business suit.

Case 2: In Apriori, 12 cashew can not be found until minsupp = 0.2, but the number of one frequent item is up to 13. But finding out almost all items is obviously not significant (Table 8) supp(7) = 0.55 of apple can be gotten in this process. It is reasonable that the sale of apple is higher than that of cashew. For example, in prior stage, sale of cashew is 2 kg and that of apple is 50 kg, then in next stage, the former is 6 kg and the latter is 60 kg. But apple is still frequent and cashew still not although sale of cashew increases 3 times while that of apple increases just one-sixth. So in this situation, compared with Apriori, k-nearest neighbor graph works better because it can keep

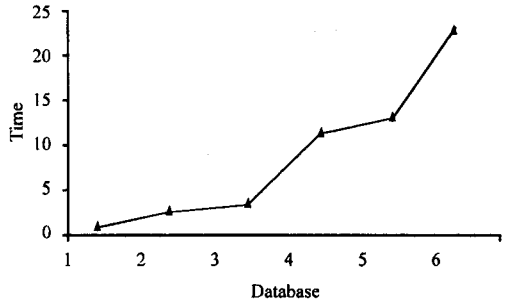

Fig. 4: Consuming time of databases with different size based on the nearest neighbor graph

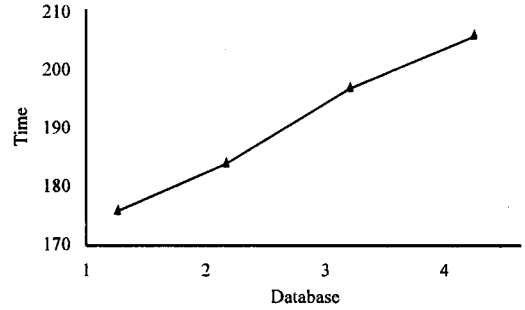

Fig. 5: Consuming time of databases with different size based on correlation analysis

out the mistakes arising from item's own properties through introducing relative bargain quantity and calculating correlation with increment.

- These two algorithms are also good from maturity, i.e. they can find all itemsets which Apriori can find. K-nearest neighbor graph is to find nearest neighbors of each against-expectation item, so it can do well employing suitable threshold. And correlation analysis must not miss some against-expectation itemsets for its operating of visiting and pruning on tree.
- The advantages of two algorithms still exist even database changes.

Time complexity: We have performed several experiments on databases with different size to illustrate the time complexity.

We employ 6 transaction databases including 7500, 10000, 20000, 30000, 40000, 50000transactions respectively. From Fig. 4, we can see that consuming time increases with the growth of size of database.
Similar trend can be found in correlation analysis (Fig. 5).

Comparison between two algorithms: K-nearest neighbor graph and correlation analysis are evaluated, respectively,

either of which can make up drawbacks of association rule with support-confidence model and is good at static classification.

The former enhances the correctness by introducing relative bargain quantity and considering increment to be quotient. And its manner of result output can be understood because the sequence of all nearest neighbors of each item is from strong to weak.

Correlation analysis can enhance correctness and reduce time cost by pruning. Introducing fuzzy theorem makes its result more reasonable. Finally, it is convenient for decision makers to distinguish positive correlation from negative correlation for each against-expectation item.

CONCLUSIONS

We have designed a new yet efficient strategy for identifying against-expectation patterns in databases. The techniques for mining against-expectation patterns are previously undeveloped. Present algorithms for identifying against-expectation patterns are based on the Nearest-Neighbor Graph and Correlation Analysis techniques. We have conducted experiments for evaluating our algorithms and demonstrated that our approach can efficiently discover against-expectation patterns.

ACKNOWLEDGEMENT

This work is partially supported by Australian large ARC grants (DP0449535 and DP0559536), a China NSFC major research Program (60496321) and a China NSFC grant (60463003)

REFERENCE

1. Padmanabhan, B. and A. Tuzhilin, 1998. A belief-driven method for discovering unexpected patterns. SIGKDD, pp: 94-100.
2. Padmanabhan, B. and A. Tuzhilin, 2000. Small is beautiful: Discovering the minimal set of unexpected patterns. SIGKDD, pp: 54-63.
3. Hussain, F., H. Liu, E. Suzuki and H. Lu, 2000. Exception rule mining with a relative interestiness measure. PAKDD, pp: 86-97.
4. Hwang, S., S. Ho and J. Tang, 1999. Mining exception instances to facilitate workflow exception handling. DASFAA, pp: 45-52.
5. Liu, H., H. Lu, L. Feng and F. Hussain, 1999. Efficient search of reliable exceptions. PAKDD, pp: 194-204.
6. Suzuki, E. and M. Shimura, 1996. Exceptional knowledge discovery in databases based on information theory. KDD, pp: 275-278.
7. Savasere, E. Omiecinski and S. Navathe, 1998. Mining for strong negative associations in a large database of customer transactions. ICDE, pp: 494-502.
8. Wu, X., C. Zhang and S. Zhang, 2002. Mining both positive and negative association rules. In: Proc. 21st Intl. Conf. Machine Learning (ICML), pp: 658-665.
9. Karypis, G., E. Han and V. Kumar, 1999. CHAMELEON: A hierarchical clustering algorithm using dynamic modeling. IEEE Computer, pp. 68-75.
10. Eppstein, D., M.S. Patterson and F.F. Yao, 1997. On nearest-neighbor graphs. Discrete Comput. Geom., 17: 263-282.
11. Brin, S., R. Motwani and C. Silverstein, 1997. Beyond Market Baskets: Generalizing association rules to correlations. SIGMOD, pp: 265-276.
12. Bay, S.D. and M.J. Pazzani, 2000. Discovering and describing category differences: What makes a discovered difference insightful? In: Proc. 22nd Annual Meeting of the Cognitive Science Society.