Against-Expectation Pattern Discovery: Identifying Interactions within Items with Large Relative-Contrasts in Databases

Dingrong Yuan, Xiaofang You, Chengqi Zhang

***Abstract*—We design a new algorithm for identifying against-expectation patterns. An against-expectation pattern is either an itemset whose support is out of a range of the expected support value, referred to as an against-expectation itemset, or it is an association rule generated by an against-expectation itemset, referred to as an against-expectation rule. Therefore, against-expectation patterns are interactions within those items whose supports have large relative-contrasts in a given database. We evaluate our algorithms experimentally, and demonstrate that our approach is efficient and promising**

***Index Terms*—Exception, against-expectation pattern, nearest-neighbor graph, correlation analysis.**

I. Introduction

Traditionally, association analysis has focused on techniques aimed at discovering interactions within data. It has mainly involved association rules [1,4,21] and negative association rules [18,23]. These rules can be identified from data by using statistical methods and grouping. In real world applications, data marketers seek to identify interactions and predict profit potential in the relative-contrast of sales. Meanwhile, they recognise that principle items, having large relative-contrasts with respect to their supports expected for a given database, may provide larger profit potential than those with low relative-contrasts. In this paper, we refer to interactions within items that have large relative-contrast as against-expectation patterns. Up until now, the techniques for mining against-expectation patterns have been undeveloped. To rectify this, our paper studies the issue of mining against-expectation patterns in databases.

An against-expectation pattern is either an itemset whose support is out of a range of the expected support value (expectation), referred to here as an against-expectation itemset, or an association rule generated from against-expectation itemsets, referred to as an against-expectation rule.

Dingrong Yuan is with College of Computer Science and Information Technology Guangxi Normal University, Guilin, 541004, China (dryuan@mailbox.gxnu.edu.cn).

Xiaofang You is with the College of Computer Science and Information Technology Guangxi Normal University, Guilin, 541004, China.

Chengqi Zhang is with Faculty of Information Technology, University of Technology Sydney PO Box 123, Broadway NSW 2007, Australia(chengqi@it.uts.edu.au).

If we use extant frequent-pattern-discovery algorithms to mine a market basket dataset, the item ‘apple’ can be identified as a frequent pattern (itemset), even though its support (= 200) is much less than expected sales (= 300). This is because ‘apple’ is a popular fruit, and is frequently purchased. Compared 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) is much greater than its expected sales (= 5). In an applied context, while the frequent pattern ‘apple’ is commonsense, the purchasing increase of ‘cashew’ is desired in marketing decision-making, and constitutes the against-expectation pattern which is 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 amount of products purchased in the next time-lag.

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

Related research includes the following: unexpected patterns [14,15], exceptional patterns [6,8,10,19], and negative association rules [18,23]. The first and second are known as ‘exceptions of rules’, and also as ‘surprising patterns’, whereas ‘negative association rules’ represents a negative relation between two itemsets.

An exception of a rule is defined as a deviational pattern to a well-known fact, and exhibits unexpectedness. For example, while ‘bird(x) → flies(x)’ is a well-known fact, mining exceptional rules aims to find patterns such as ‘bird(x), penguin(x) → ~flies(x)’. The negative relation actually 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 [18,23].

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

In addition to those mentioned above, there are also some differences between mining against-expectation patterns and mining the other two patterns change patterns [12], and interesting patterns using user expectation [13]. In these methods, a decision tree is used for mining changes, which is

very distinct from the algorithms employed in this paper. Finding interesting patterns according to user expectation is a kind of subjective measure, while our algorithms aim at objectivity.

As such, while the above achievements provide a good insight into exceptional pattern discovery, this paper will focus on identifying against-expectation patterns. In Section II, we formally define some basic concepts and examine the approach issue of mining against-expectation patterns. In Section III we describe our approach and compare it with existing algorithms. In Section IV, we conduct a set of experiments to evaluate our algorithms. We summarize our contribution in Section V

II. A Framework for Identifying Against-expectation Patterns

In this section we present some basic concepts and describe the issue of mining against-expectation patterns in databases. In particular, we design a new framework for mining against-expectation patterns that consists of interactions within items, with large relative-contrasts referenced to their expectations in a given database. This is based on heterogeneity metrics.

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.

TABLE I
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}
Support	f_1	f_2	.	f_n

In Table I, T_j is the identifier of transactions in D, a_{jk} is the quantity of item i_k in transaction T_j, f_k is the quantity (support) of i_k in D, or the sum of a_{+k} in *k*th column.

In marketing, data marketers must know the sales expectation of each product. This is used to determine how many items should be bought each month (or during a specified time-lag).

TABLE II
THE EXPECTATION OF ITEMS

	i_1	i_2	…	i_n
Expectation	e_1	e_2	…	e_n

In Table II, e_k is the expected quantity of i_k in D. For example, in Section I, expectation(apple) = 300, support(apple) = 200, expectation(cashew) = 5, and support(cashew) = 20. Therefore, the support of 'cashew' is far larger than its expectation because

$$\textit{increased}(\text{cashew}) = \text{support}(\text{cashew}) - \text{expectation}(\text{cashew})$$

$$= 20 - 5 = 15.$$

This means that the relative contrast, which is the ratio of increment to expectation used for denoting the gap between support and expectation rec(cashew), is three times its expectation. That is,

$$\textit{rec}(\text{cashew}) = \text{increased}(\text{cashew}) / \text{expectation}(\text{cashew})$$

$$= 15/5 = 3.$$

Hence, 'cashew' is an against-expectation pattern. However, the support of 'apple' is far less than its expectation because

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

$$= 200 - 300 = -100.$$

This means that the relative contrast *rec*(apple) is

$$\textit{rec}(\text{apple}) = \textit{increased}(\text{apple}) / \text{expectation}(\text{apple}) = -100/300 = -1/3.$$

We also refer to 'apple' as an interesting against-expectation pattern if *rec*(apple) is out of a given range (neighbour) of expectation(apple).

Against-expectation patterns are defined as either those itemsets whose supports are out of a δ-neighbour of their expected values, referred to as *against-expectation itemsets*, or those rules that are interactions within against-expectation itemsets, referred to as *against-expectation rules*. An against-expectation itemset X has its support out of the δ_X-neighbour of its expectation (with a large relative contrast), i.e.

$$\textit{recar}(X) = |\textit{increased}(X)| / \text{expectation}(X)$$

$$= |\text{support}(X) - \text{expectation}(X)| / \text{expectation}(X)$$

$$> \delta_X$$

where δ_X is a user-specified minimum relative-contrast for X. Certainly, recar(X) is a heterogeneity metrics, because δ_X can be different with different X.

An against-expectation rule is of the form

$$X \rightarrow Y$$

which is the interaction between the against-expectation itemsets X and Y; or one of X and Y is a frequent itemset and the other an against-expectation itemset. For example, 'apple → cashew' can be an against-expectation rule between the above two against-expectation itemsets.

An against-expectation rule $X \rightarrow Y$ is interesting if its confidence is greater than, or equal to, a user-specified minimum confidence (minconf).

We classify against-expectation patterns as follows: increment patterns, decrement patterns and negative associations.

- An increment pattern is either an itemset X whose actual support is greater than its expected value, e.g., support(X) – expectation(X) > *e* (where *e* is a user-specified positive value) – this is referred to as an *increment itemset*; or it is a rule that is an interaction within increment itemsets, and is referred to as an *increment rule*.
- A decrement pattern is either an itemset X whose actual support is less than its expected value, e.g., support(X) – expectation(X) < -*e* (where *e* is a user-specified positive value) – referred to as a *decrement itemset*; or it is a rule that is an interaction within decrement itemsets, and is referred to as a *decrement rule*.
- A *mutually-exclusive* correlation is a rule in which its antecedent and action belong to different against-expectation itemsets. That is, either (1) its antecedent is an increment itemset and its action a decrement itemset; or (2) its antecedent is a decrement itemset and its action an increment itemset.
- A *companionate* correlation is a rule in which one of its antecedents and actions is a frequent itemset and the other is an against-expectation itemset. That is, either (1) its antecedent is an increment itemset and its action is a frequent itemset; (2) its antecedent is a frequent itemset and its action an increment itemset; (3) its antecedent is a decrement itemset and its action a frequent itemset; or (4) its antecedent is a frequent itemset and its action a decrement itemset;

The problem of mining against-expectation patterns is a challenging issue because it is *very different from* those problems faced when discovering frequent patterns (or association rules). Because against-expectation patterns can be hidden in both frequent and infrequent itemsets, traditional pruning techniques are inefficient for identifying such 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 itemsets can generate against-expectation patterns;

(c) determine which against-expectation patterns are really useful to applications; and

(d) determine the heterogeneity metrics of against-expectation patterns.

One must remember that this process can be expensive and dynamic. The leaders of a new company must make an effort to estimate expectation by analyzing the environment and possible customers. For an old company, data relating to items recorded in a previous time-lag can be used as expectations. Thus, we can check the support of items and identify the against-expectation patterns in that time-lag. We can also predict the support of items and the against-expectation patterns in the next time-lag. In the following, for simplification, we define a time-lag as a month. From this point, our against-expectation patterns are similar to change patterns [12,13], though discovering against-contrast patterns is based on the expected support of items.

The technique to be developed in this paper consists of a two-step approach as follows:

(1) Generating a set of interesting items (i.e., items with a large relative-contrast), and

(2) Identifying interactions within these interesting items based on the Nearest-Neighbor Graph and Correlation Analysis techniques.

III. Algorithm Description

Traditional mining algorithms assume that an association rule is interesting as long as it satisfies minimum support and minimum confidence. But a number of researchers have proved that this can generate many uninteresting rules. Meanwhile, some interesting rules can be missed without taking into account the item's own change trend.

Example 1. Let min_supp = 33.3%. Consider two transaction databases D1 and D2, as shown in Tables III and IV.

TABLE III
D1: 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 IV
D2: 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

Using existing association rule mining algorithms, we can identify certain association rules in D1, or D2, or D1∪D2. For example, association rules 'toothbrush → toothpaste', 'toothbrush → shampoo' and 'bread → jam' are of interest in D2 because

$$supp(toothbrush \rightarrow toothpaste) = 33.3\%$$
$$conf(toothbrush \rightarrow toothpaste) = 100\%$$
$$supp(toothbrush \rightarrow shampoo) = 33.3\%$$
$$conf(toothbrush \rightarrow shampoo) = 100\%$$
$$supp(bread \rightarrow jam) = 50\%$$
$$conf(bread \rightarrow jam) = 100\%$$

Let us consider the support and increment (relative contrast) of the items 'apple' and 'cashew' as shown in Table V.

TABLE V
COMPARING APPLE WITH CASHEW

	Jan's supp	Feb's sup	increment
Apple	50%	58.3%	16.7%
cashew	10%	25%	300%

Obviously, whenever it is January or February, 'apple' is always frequent and 'cashew' is always infrequent. But the increment (relative contrast) of 'cashew' is far higher than that of 'apple'. Therefore, 'cashew' is of much more interest when considering the relative contrast (i.e., change trend). These interesting itemsets cannot be found by using existing association rule mining algorithms. Certainly, we can identify 'cashew' as a frequent itemset, using Apriori-like algorithms, by decreasing the min_supp to 25%. However, decreasing the min_supp can lead to the generation of a great many uninteresting itemsets. It is not an intelligent way to proceed. In particular, Apriori-like approaches do not provide information concerning the change trend of itemsets. We are, therefore, encouraged to develop new techniques for discovering against-expectation patterns.

Our proposed approach for identifying against-expectation patterns uses the stock of merchandise in the previous month as an expectation, in order to identify against-expectation patterns in the present month.

A. Finding interesting itemsets using square deviation

This subsection presents techniques for finding a candidate set consisting of particularly interesting items (1-itemset) based on the square deviation, and is aimed at mining against-expectation patterns. Generally, if a piece of merchandise (an item) has a small relative contrast referenced to its expectation, the sales trend of this merchandise is in control. And it is uninteresting when mining against-expectation patterns. For efficiency, the item should be deleted from the candidate set.

Stocked items can reflect a decision makers' anticipation of the sale trend for each item in the future. Therefore, in Definition 1, stock is considered to be equal to the expectation mentioned above. The stage below is necessary for determining changing trends, and its span can be decided according to practical situations, such as week, month or year.

Definition 1. Let x_{ij} be the stock of i-th merchandise at j-th stage, where $0 \le j \le m$, $0 \le I \le n$; and p_{ij} denote the increment ratio between the j-th stage and the (j+1)-th stage of the i-th merchandise. Then we have

$$p_{ij} = \frac{x_{i(j+1)} - x_{ij}}{x_{ij}}.$$

Therefore, its incremental ratio math expectation is

$$E(p_i) = \frac{\sum_{j=1}^{m-1} p_{ij}}{m-1}$$

and its square deviation is

$$D(i) = \sum_{j=1}^{m} (p_{ij} - E(p_i))^2.$$

Let propdenote the threshold given. Then merchandise i is of interest if $D(i) \ge$ prop.

The algorithm to identify all interesting items (merchandise) is constructed in Fig. 1.

Input Stock: set of the number of goods, Prop: minimum math expectation;
Output MID: set of candidate 1-itemsets
(1) **for** each j in Stock
(2) **begin**
(3) calculate $E(p_i)$, D(i);
(4) **if** (D(i) ≥ Prop)
(5) MID ← MID ∪ {i};
(6) **end;**
(7) **Output** all items of interest in MID;
(8) **Return**;

Fig.1. FIM: Generating the candidate set (of 1-itemsets)

In algorithm FIM, all items with large relative contrasts referenced to their expectations are identified using square deviation. In the following Sections III.B and III.C, k-nearest neighbor and correlation analysis will be used to calculate the correlation between two items and generate the candidate set of k-itemsets, which consist of all interesting itemsets potentially useful for generating against-expectation patterns.

B. Nearest neighbor graph based method

Definition 2. Let $A = \{a_0, a_1, a_2, \ldots a_{i-1} a_i, \ldots a_n\}$, $B = \{b_0, b_1, b_2, \ldots b_{i-1} b_i, \ldots b_n\}$ be the stocks of goods A and B. The increments between two adjacent phases for A and B are denoted as

$$A^1=\{a_1\text{-}a_0, a_2\text{-}a_1, \ldots a_i\text{-}a_{i-1}, \ldots a_n\text{-}a_{n-1}\}$$

$$B^1=\{b_1\text{-}b_0, b_2\text{-}b_1, \ldots b_i\text{-}b_{i-1}, \ldots b_n\text{-}b_{n-1}\}.$$

Let $x_{i-1} = (a_i - a_{i-1})/(b_i - b_{i-1})$ and $x_{i-1} \in X$, $X = \frac{A^1}{B^1} = \{x_0, x_1, x_2, \ldots x_{i-1} x_i, \ldots x_n\}$ be known as the relative bargaining quantity of A and B.

The nearest neighbor graph is constructed as follows. For all items in a given database, we first draft a figure with the items as its points. Any two points are connected by an edge, and the edge is associated with the square deviation distance between the two points. For instance, in Definition 2, A and B stand for the points *a* and *b* in a figure respectively. Then the distance between A and B (*a* and *b*) is defined as follows.

$$d_{A,B} = P(X) = P\left(A^1/B^1\right) = \sum_{i=0}^{n-1} \left(X_i - E[X]\right)^2.$$

After finishing the figure accordingly, we then get rid of some of the edges when their distances are larger than the threshold. The rest of the figure is simply the nearest neighbor graph for the database [10]. In this nearest neighbor graph, we refer to certain items as the nearest neighbors of the point a, if they are linked through edges to a. We can reconstruct the nearest neighbor graph by ranking the nearest neighbors for a certain point in decreasing order.

We now describe the algorithm for generating the candidate set of j-itemsets using the above k-nearest neighbor graph.

Input SaleTable: dataset; k: number of nearest neighbors;

Output MID: set of j-itemsets;

(1) StageTable ← SaleTable;
(2) **for** A in MID (firstly generated in Algorithm FIM)
(3) **begin for** B in MID (firstly generated in Algorithm FIM)
(4) **begin** calculate X(A, B);
(5) calculate d(A, B);
(6) **if** (d(A, B) ≥ k)
(7) A.neighbors ← B;
(8) **generate** a candidate itemset i by A and B;
(9) MID ← MID ∪ {i};
(10) **end**
(11) **end**
(12) **Output** all itemsets in MID;
(13) **Return;**

Fig.2. KNNG: Generating the candidate set of *j*-itemsets using the *k*-nearest neighbor graph

From the above description, the algorithm KNNG generates the candidate set of 2-itemsets, using all 1-itemsets in MID, generated first in the Algorithm FIM; the algorithm KNNG generates the candidate set of 3-itemsets, using all 2-itemsets in MID, generated in the Algorithm KNNG; and the algorithm KNNG generates the candidate set of j-itemsets, when using all (j-1)-itemsets in MID, generated in the Algorithm KNNG.

From the definition of against-expectation patterns, all candidate j-itemsets identified by the Algorithms FIM and KNNG are against-expectation itemsets. By way of these against-expectation itemsets, we can generate some against-expectation rules of interest. Here we omit the algorithm for generating against-expectation rules because it is similar to the Apriori algorithm.

C. Correlation-analysis based approach

In this section we design a correlation-analysis based algorithm for generating a candidate set of k-itemsets. The correlation-analysis based algorithm detects whether the correlation between two goods is positive or negative. That is, whether the correlation has a acceleration action or is restricted. In particular, this algorithm can provide pattern quality superior to that of the nearest-neighbor-graph based algorithm.

Definitions and theorems

Definition 3. Let $P(X)$ denote the probability of X occurring in a transaction, and $P(\overline{X})$ the probability of X not occurring in a transaction, where $P(\overline{X})$=1- $P(X)$; and let $P(X \cup Y)$ denote the probability of X and Y both occurring in a transaction. If $P(X \cup Y) \neq P(X)P(Y)$, then X and Y are dependent on each other; otherwise, independent of each other.

Definition 4. If $X_{a1}, X_{a2}, \ldots, X_{a3}$ in a transaction database, are correlated with each other, then $X_{a1}, X_{a2}, \ldots, X_{a3}$ is known as a correlation rule.

Definition 5. Let $X' = X_{a1}, X_{a2}, \ldots, X_{a3}$ and $X = \{X_1, X_2, \ldots, X_{n-1}, X_n\}$, then X' is referred to as a subset of X, and X is referred to as a superset of X'.

Each itemset consists of certain attributes, and a superset consists of all the attributes in its subsets. Therefore, reducing time cost is possible by getting rid of those itemsets that contain at least one uncorrelated subset (a non-correlation rule).

Definition 6. The correlation-analysis quantity of goods A and B is defined as

$$Corr(A,B) = \frac{P(A \cup B)}{P(A) \times P(B)}.$$

For n goods, each denoted as Gi, where 1<=i<=n, the correlation-analysis quantity is

$$Corr(G_1, G_2, ..., G_n) = \frac{P(\bigcup_{i=1}^{n} G_i)}{\prod_{i=1}^{n} P(G_i)}$$

The more closed to 1 the correlation-analysis quantity of *A* and *B* is, the better the independency of *A* and *B*. If $Corr(A,B) > 1$, then *A* and *B* are positively correlated, and if $Corr(A,B) < 1$, then *A* and *B* are negatively correlated .

Algorithm design

The correlation-analysis based algorithm is a post-process based upon a transaction database. It is suitable for static data classification, and is an improvement on Apriori-like algorithms for mining association rules using a support-confidence framework.

Reducing the time complexity of our correlation-analysis based algorithm can be implemented by way of a pruning algorithm that utilizes the closure property of correlation itemsets. Using a pruning algorithm, we can obtain all minimum correlation itemsets of interest. The pruning process is shown in Fig. 3 where, if $\mathrm{Corr}(0,1) \neq 1$, then 0 and 1 are correlated, as are all its supersets. Pruning its left subtree at (0,1) is actually a minimum correlation itemset.

The final output looks like this:

A: Positive correlated item: (*B*), (*E*,*F*), i.e., (*A*,*B*) and (*A*,*E*,*F*) are two minimum correlation itemsets concerning item *A*.

Negative correlated item: (*D*), (*G*).

So, it is easy to detect those interesting items that correlate with A

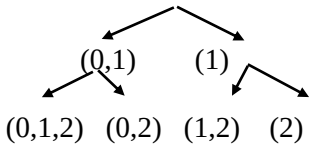


Fig. 3. Set-enumeration tree

We now construct the correlation-analysis based algorithm as follows.

Input TD: set of transactions; ProRange: a threshold for independency;

Output PosCorr: set of all positive correlation itemsets;
NegCorr: set of all negative correlation itemsets;

(1) PosCorr ← ∅; NegCorr ← ∅;

(2) **Construct** a set-enumeration tree for a 1-itemset in MID for TD (generated in Algorithm FIM);

(3) **Scan** each node A in the set-enumeration tree

(4) **If** (Corr > 1 + ProRange)
PosCorr ← PosCorr ∪ {A};

(5) **If** (Corr < 1-ProRange)
NegCorr ← NegCorr ∪ {A};

(6) **If** (|1-ProRange| ≤ Corr ≤ |1+ProRange|)
Delete A from the tree;

(7) **Output** PosCorr and NegCorr;

(8) **Return**;

Fig. 4. CA: Generating the candidate set of k-itemsets using correlation analysis

From the above description, the algorithm CA generates the candidate sets PosCorr and NegCorr, using all 1-itemsets in MID generated in the Algorithm FIM. The candidate sets consist of j-itemsets of interest.

From the definition of against-expectation patterns, all candidate j-itemsets identified by the Algorithms FIM and CA are against-expectation itemsets. Through these against-expectation itemsets, we can generate some against-expectation rules of interest. Like in the KNNG algorithm, we omit the algorithm for generating against-expectation rules.

In our CA algorithm, a threshold $proRange$ is used to measure independency. For example, items in X are independent when:

$$|1 - proRange| \le Corr(X) \le |1 + proRange|.$$

In addition, positive correlation is defined as

$$Corr(X) > |1 + proRange|$$

and negative correlation is defined as

$$Corr(X) < |1 - proRange|.$$

Note that we can use, for example, Chi-square, as our another metrics for measuring the correlation of itemsets by statistical means.

IV. Experiments

To evaluate our two metrics, we have conducted extensive experiments on a DELL Workstation PWS650 with 2G main memory, 2.6G CPU, and WINDOWS 2000. We evaluate our approaches using the databases generated from http://www.kdnuggets.com/ (Synthetic Classification Data Sets from the Internet).

To evaluate our algorithm, we used several databases of different sizes, where the largest database included 40000 transactions involving over 1000 items. But for narrating

convenience, we choose below to analyze the result of the smallest of our databases.

The database consists of 100 transactions involving over 15 items. Experimental results of the k-nearest neighbor, correlation analysis, and Apriori are shown in Tables VI to VIII.

TABLE VI
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 VII
CORRELATION ANALYSIS

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

TABLE VIII
COMPARISON BETWEEN SUPPORT AND VARIANCE

Merchandise ID	Support	Variance
0	0.43	1. 374705
1	0.43	0. 7838024
2	0.20	1. 7
3	0.50	1. 310159
4	0.40	1. 099296
5	0.24	7. 706056
6	0.35	1. 471311
7	0.55	3. 420516
8	0.60	1. 004835
9	0.83	0. 7899691
10	0.60	1. 109732
11	0.09	1. 05
12	0.18	25. 85556
13	0.74	1. 00284
14	0.12	2. 422

A. Algorithm Analysis

Simulation database

(1) Our algorithms are satisfactory from maturity

From Table VIII we can see clearly that there is no relation between the merchandise's variance and support. For example, in this experiment the information for merchandise 7 and 12 is shown.

(1) supp(7apple)=0.55>>supp(12cashew)=0.18
(2) variance(7apple)=3.2886<<variance(12cashew)=23.1667

Here, (1) means the sale of apples is higher than that of cashews, but in this experiment cashews are obviously more significant than apples. Meanwhile, the variance calculated by our algorithm has shown that, (2) means that the item cashew will be found more easily than apple by our algorithm. Our algorithm can find this type of against-expectation itemset, so the algorithm for finding interesting itemsets satisfies our needs.

Two points can confirm this. First, we use the increment rate of each of two adjacent phases to calculate the variance. This way, we can generate some against-expectation itemsets, which are always omitted easily because they are infrequent. Second, because the pruning structure is based on calculating the variance, we can keep all the interesting itemsets we need. We just require an appropriate threshold.

In addition, the k-nearest neighbor graph is needed to find the nearest neighbors of each interesting goods item, so it can be successful in employing a suitable threshold. Further, the correlation analysis must not miss interesting itemsets during its operation of visiting and pruning the tree.

(2) Our algorithm is very accurate

This experiment demonstrates that the desired result can be obtained. First, there are no uninteresting association rules. Although

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

Meanwhile, with the k-nearest neighbor graph, we get:

8: 3, 4, 6, 13,

9: 12, 13,

10: 13,

13: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,

It can easily be seen that 8 toothbrush is closest to 3 toothpaste, and has nothing to do with 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 an

uninteresting rule. Although supp(9→13) = 73%, it is not useful in decision-making because of the closed correlation between the two. Meanwhile, the k-nearest neighbor graph reveals that they are not very closed, and the correlation analysis reveals that they are not independent.

The most important is that the k-nearest neighbor graph can avoid being misled by the actual sale of merchandises, making the increment rate of the analysis objective. In addition, the output patterns of the two algorithms are very clear.

(3) **The advantages of the two algorithms still exist, even if the database changes**

In addition to the above experimental analysis, we have performed several experiments on databases of different sizes to illustrate the efficiency of the algorithms. In Tables IX, X and XI, we have chosen the same transactions, the same items ((T, I) = (100, 10), (T, I) = (100, 15), (T, I) = (150, 10), (T, I) = (150, 15)), and the different (or 'various') average items of transactions (A). In Table IX, we present the performance of the Correlation analysis, based on the same transaction (T) and different items (I), the same item and different transactions, and the average number of items of transactions (A) = 5. We do the same in Table X and Table XI, (A) = 8, (A) = 10 respectively. Table X shows the runtime of the k-nearest neighbor graph based on different simulation databases. We evaluat the number of positive correlation items (pc), negative correlation items (nc), and overall correlation items (oc), and also the runtime (rt).

TABLE IX
CORRELATION ANALYSIS ON (A)=5

(T, I)	pc	nc	oc	rt
(100,10)	38	9	47	0.156
(100,15)	45	43	88	100.906
(150,10)	32	8	40	0.172
(150,15)	49	49	98	101.141

TABLE X
CORRELATION ANALYSIS ON (A)=8

(T, I)	pc	nc	oc	rt
(100,10)	48	0	48	0.172
(100,15)	125	17	142	100.89
(150,10)	56	0	56	0.218
(150,15)	131	17	148	100.672

TABLE XI
CORRELATION ANALYSIS ON (A)=10

(T, I)	pc	nc	oc	rt
(100,10)	50	0	50	0.234
(100,15)	203	10	213	103.359
(150,10)	40	0	40	0.312
(150,15)	259	10	269	101.141

TABLE XII
K-NEAREST NEIGHBOR GRAPH

(T, I, A)	rt	(T, I, A)	rt
(100,10,5)	0	(150,10,5)	0
(100,10,8)	0	(150,10,8)	0
(100,10,10)	0	(150,10,10)	0.016
(200,10,5)	0.015	(100,15,5)	0
(200,10,8)	0	(100,15,8)	0
(200,10,10)	0	(100,15,10)	0

The runtime equals 0 because the system's capability is restricted, so we have chosen runtime = 0 when runtime < 0.001.)

Tables IX, X, XI and XII reveal the following results:

1: The number of correlation itemsets is most deeply influenced by the number of items (I). The greater the number of items, the more correlation itemsets there are.

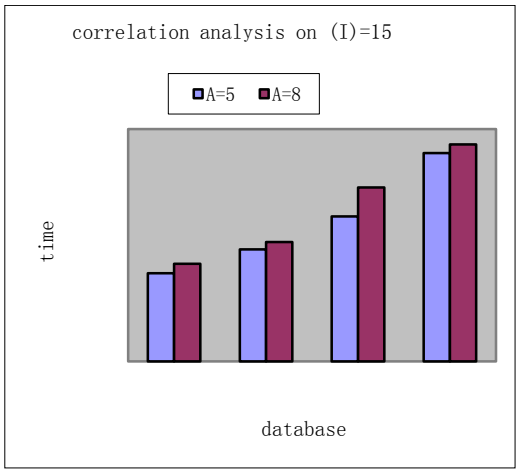
2: The length of the correlation itemsets is affected by the average length of items (A). Long average length results in long correlation itemsets.

3: Compared with the other two factors, running time places most stress on the number of items. Therefore, the algorithm is more effective on a dense database than on a sparse one. Meanwhile, from Table XII, we can see clearly that the k-nearest neighbor graph algorithm can work satisfactorily.

4: The number of negative correlation items is influenced by the average length of items (A), as well as the number of items (I). As the average length of items (A) is invariant, the more the number of items is, the more the number of itemsets. As the number of items (I) is invariant, the less the length of the item, and the more the number of negative correlation items.

Simulation on larger databases

We have performed these experiments on databases which involve 2000, 3000, 5000 and 7500 transactions on 10 or 15 items, where the average length of items is 5 or 8. The results are shown in Fig. 5.



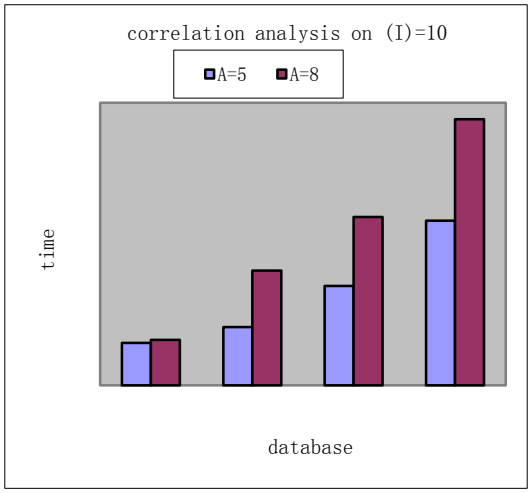
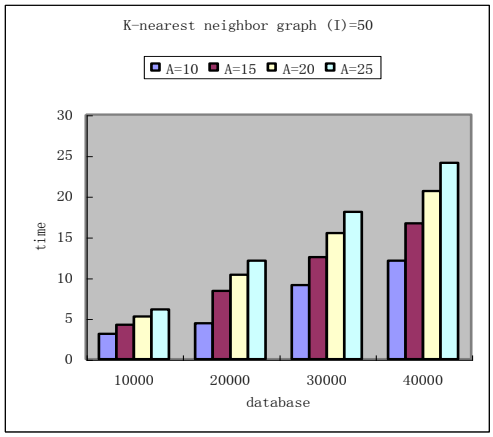
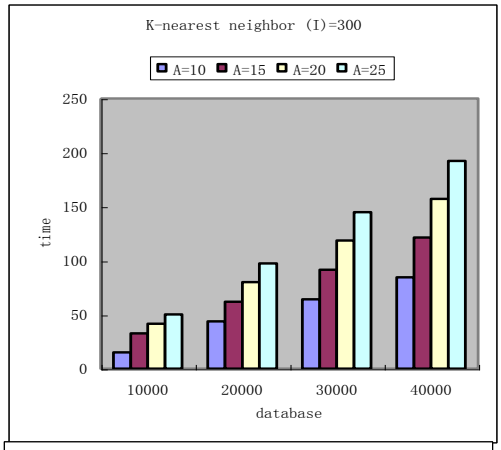
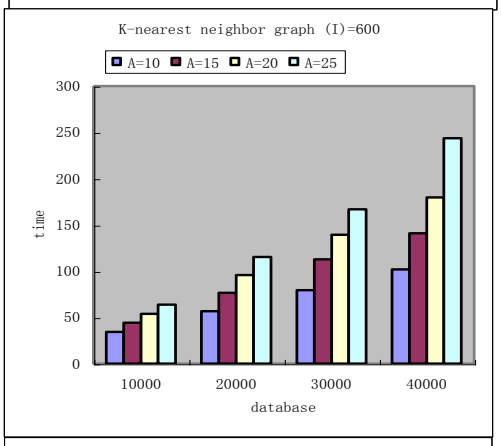


Fig. 5. Runtime of the correlation analysis graph based on different databases







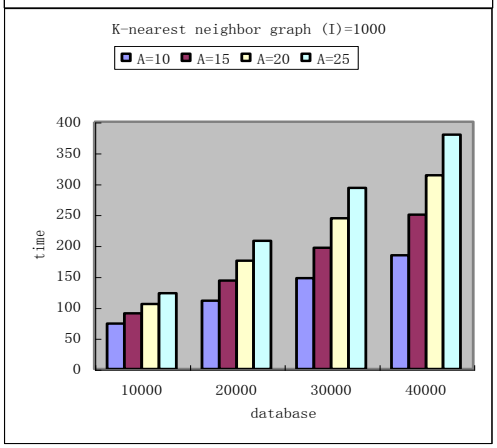


Fig. 6. The runtime of the k-nearest neighbor graph based on different factors

From Fig. 6, we can see clearly that the consuming runtime increases with the growth of the size of the database, namely, the number of transactions(T), items(I), and the average length of items(A).

Meanwhile the rules, which were found in the simulation small databases, as shown in Tables XI–XII, are still appropriate.

B. Comparison between two algorithms

The k-nearest neighbor graph and correlation analysis are evaluated, either one of which can compensate for the drawbacks of the association rule with the support-confidence model, and is useful for static classification.

The former enhances correctness by introducing relative bargain quantity, and considering the increment to be the

quotient. Further, the manner of its result output can be understood, because the sequence of all the nearest neighbors of each of the goods goes from strong to weak.

Correlation analysis can enhance correctness and reduce time costs through pruning. Introducing the fuzzy theorem makes the result more reasonable. Finally, it is convenient for decision makers to distinguish positive correlation from negative correlation for each interesting item.

V. Conclusion

We have designed a new algorithm for identifying against-expectation patterns. These patterns are interactions within items, with large relative-contrasts referenced to their expectations in a given database. This is based on heterogeneity metrics. The techniques for mining against-expectation patterns were previously undeveloped. We have experimentally evaluated our algorithms and demonstrated that our approach is efficient and promising.

Acknowledgment

This work was supported in part by the Australian Research Council (ARC) under grant DP0988016, the Nature Science Foundation (NSF) of China under grant 90718020, the China 973 Program under grant 2008CB317108, and the Guangxi NSF under grant GKZ0640069.

References

[1] R. Agrawal, T. Imielinski, and A. Swami (1993), Mining association rules between sets of items in large databases. In: *Proceedings of the ACM SIGMOD Conference on Management of Data*, pp. 207-216.

[2] Brock Barber and Howard J. Hamilton (2003): Extracting Share Frequent Itemsets with Infrequent Subsets. Data Min. Knowl. Discov. 7(2): 153-185.

[3] S. Bay and M. Pazzani (2001), Detecting Group Differences: Mining Contrast Sets. *Data Mining and Knowledge Discovery*, 5(3): 213-246.

[4] S. Brin, R. Motwani and C. Silverstein (1997), Beyond Market Baskets: Generalizing Association Rules to Correlations. In: *Proceedings of the ACM SIGMOD Conference on Management of Data*, pp. 265-276.

[5] Y.B. Cho, Y.H. Cho and S. Kim (2005), Mining changes in customer buying behavior for collaborative recommendations. *Expert Systems with Applications*, 28(2): 359-369.

[6] L. Egghe and C. Michel (2003), Construction of weak and strong similarity measures for ordered sets of documents using fuzzy set techniques. *Information Processing and Management*, 39: 771-807.

[7] Ping-Yu Hsu, Yen-Liang Chen and Chun-Ching Ling (2004), Algorithms for mining association rules in bag databases**.** *Information Sciences*, Volume 166, Issues 1-4: 31-47.

[8] F. Hussain, H. Liu, E. Suzuki, and H. Lu (2000), Exception Rule Mining with a Relative Interestiness Measure. In: *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining* (PAKDD), pp. 86-97.

[9] S. Hwang, S. Ho, and J. Tang (1999), Mining Exception Instances to Facilitate Workflow Exception Handling. In: *Proceedings of the International Conference on Database Systems for Advanced Applications* (DASFAA), pp. 45-52.

[10] G. Karypis, E. Han, and V. Kumar (1999), CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling. *IEEE Computer*, pp. 68-75.

[11] Xuemin Lin, Yijun Li and Chi-Ping Tsang (1999), Applying on-line bitmap indexing to reduce counting costs in mining association rules. *Information Sciences*, Volume 120, Issues 1-4: 197-208.

[12] B. Liu, W. Hsu, H. Han and Y. Xia (2000), Mining changes for real-life applications. In: Second International Conference on Data Warehousing and Knowledge Discovery, 337-346.

[13] B. Liu, W. Hsu, L. Mun and H. Lee (1999), Finding interesting patterns using user expectations. In: IEEE Transactions on Knowledge and Data Engineering, (11)6, 817-832.

[14] H. Liu, H. Lu, L. Feng, and F. Hussain (1999), Efficient Search of Reliable Exceptions. In: *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining* (PAKDD), pp. 194-204.

[15] B. Padmanabhan and A. Tuzhilin (1998), A Belief-Driven Method for Discovering Unexpected Patterns. In: *Proceedings of the International Conference on Knowledge Discovery and Data Mining* (KDD), pp. 94-100.

[16] B. Padmanabhan and A. Tuzhilin (2000), Small is beautiful: discovering the minimal set of unexpected patterns. In: *Proceedings of the International Conference on Knowledge Discovery and Data Mining* (KDD), pp. 54-63.

[17] G. Piatetsky-Shapiro (1991), Discovery, analysis, and presentation of strong rules. In: *Knowledge discovery in Databases*, G. Piatetsky-Shapiro and W. Frawley (Eds.), AAAI Press/MIT Press, pp229-248.

[18] Savasere, E. Omiecinski, and S. Navathe (1998), Mining for Strong Negative Associations in a Large Database of Customer Transactions. In: *Proceedings of the International Conference on Data Engineering* (ICDE), pp. 494-502.

[19] E. Suzuki and M. Shimura (1996), Exceptional Knowledge Discovery in Databases Based on Information Theory. In: *Proceedings of the International Conference on Knowledge Discovery and Data Mining* (KDD), pp. 275-278.

[20] K. Wang, S. Zhou, A. Fu and X. Yu (2003). Mining Changes of Classification by Correspondence Tracing. SIAMDM'03, 2003.

[21] G. Webb (2000). Efficient search for association rules. In: *Proceedings of the International Conference on Knowledge Discovery and Data Mining* (KDD), pp. 99-107.

[22] G. Webb, S. Butler and D. Newlands (2003), On detecting differences between groups. *KDD'03*, pp. 256-265.

[23] X. Wu, C. Zhang and S. Zhang (2002). Mining both positive and negative association rules. In: *Proceedings of 21st International Conference on Machine Learning* (ICML), pp. 658-665.

[24] Shichao Zhang, Jingli Lu and Chengqi Zhang (2004), A fuzzy logic based method to acquire user threshold of minimum-support for mining association rules. *Information Sciences*, Volume 164, Issues 1-4: 1-16.