Stock Movement Prediction And N-Dimensional Inter-Transaction Association Rules

Extended Abstract

Hongjun Lu[1] *Jiawei Han*[2] *Ling Feng*[3]

[1] The Hong Kong University of Science and Technology, China. luhj@cs.ust.hk
[2] Simon Fraser University, Canada. han@cs.sfu.ca
[3] The Hong Kong Polytechnic University, China. cslfeng@comp.polyu.edu.hk

1 Inadequacy in association rule mining for stock movement prediction

Among all the data mining problems, discovering association rules from large databases is probably the most significant contribution from the database community to the field [1, 2, 5, 9, 10, 7]. The most often cited application of association rules is market basket analysis using transaction databases from supermarkets and departmental stores. We can discover rules like

R_1 : *80% of customers who bought diaper also bought beer* (*diaper* $\Rightarrow$ *beer* (20%, 80%)),

where 80% is the *confidence level* of the rule and 20% is the *support level* of the rule indicating how frequent the rule holds.

Association rules for prediction

The same concept can be applied to other applications as well. For example, to predict the stock market price movement, we can construct a transaction database in such a way that: each record (transaction) in the database represents one trading day and contains a list of winners (closing price is x% more than the previous day's closing price where x% is the trading overhead). Thus we can find rules like

R_2 : *When the prices of IBM and SUN go up, 80% of time the price of Microsoft goes up (on the same day).*

While rule R_2 reflects some relationship among the prices, its role in price prediction is limited. It is rather obvious that the traders may be more interested in the following kind of rules:

R_3 : *If the prices of IBM and SUN go up, Microsoft's will most likely (80% of time) go up* **the next day**.

Unfortunately, current association rule miners cannot discover this kind of rules.

Sequential pattern discovery cannot help either

Since the stock movement prediction is time-related, we thought sequential pattern discovery [3] might be of help. To apply sequential pattern mining techniques, we reorganize that database as follows: each stock corresponds to a customer, and transactions are represented by ups and downs. The rules that can be found are like

R_4 : *80% stock will go up after 3 consecutive loses.*

This is not really what we like.

The fundamental difference

There is a fundamental difference between rule R_3 and the other rules. The classical association rules express the associations among items purchased by one customer or share price movement within a day, i.e., *associations among items within the same transaction record.* We call them *intra-transaction association rules.* Sequential pattern discovery is also intra-transaction mining in nature because each sequence is treated as one transaction and the mining process is to find similarities among the sequences. On the other hand, rule R_3 expresses the *association among items from different transaction records.* We call it *inter-transaction association.*

2 N-dimensional inter-transaction association rules

In this stock movement prediction application, the association is along one dimension, the trading days. The concept can be extended further. If a database contains records about the time and location of buildings and facilities of a new city under development, we may be able to find such a rule:

R_5 : *After McDonald and Burger King open branches, KFC will open a branch* **two months later less than a mile away.**

Based on what have been described above, we propose *N-dimensional inter-transaction association rules* with the classical association rules as a special case.

The transaction database

Definition 1 *Let $E = \{e_1, e_2, \ldots, e_u\}$ be a set of literals, called events. Let $D_1, D_2, \ldots, D_n$ be a set of attributes. A transaction database is a database containing records in the form of $(d_1, d_2, \ldots, d_n, E_i)$ such that $\forall k(1 \leq k \leq n)(d_k \in Dom(D_k))$ where $Dom(D_k)$ is the domain of attribute D_k, and $E_i \subseteq E$. A transaction database with n attributes is called an n-dimensional transaction database.*

The attributes in an n-dimensional transaction database are called *dimensional attributes*. They describe the properties associated with the events, such as time and place. There are a wide range of application databases that can be viewed as n-dimensional transaction databases. The stock price movement database is a 1-dimensional transaction database. The example of urban development project can use a 2-dimensional transaction database where the two dimensional attributes are *month* and *block number*, and the event list includes the buildings or facilities completed during the month at a particular block.

In the current study, we will assume that the domain of a dimensional attribute can be divided into equal length intervals. For example, time can be divided into day, week, month, etc., and distance into meter, mile, etc.. The intervals can be represented by integers 0, 1, 2, ... without losing generality. With such a view, the dimensional attributes form an n-dimensional space and an event instance can be viewed as a point in the space. If we divide the space into n-dimensional cells each of which is identified by the associated n-ary tuple $(d_1, d_2, \ldots, d_n)$, each transaction in the database represents a non-empty cell with some points (events) inside it.

Definition 2 *Let $T_i = (d_{i_1}, d_{i_2}, \ldots, d_{i_n}, E_i)$ be a record in the transaction database. $(d_{i_1}, d_{i_2}, \ldots, d_{i_n})$ is the address of event $e_i \in E_i$. An event associated with its address is called an event instance, denoted by $\bar{e}_i = e_i(d_{i_1}, d_{i_2}, \ldots, d_{i_n})$.*

Figure 1 depicts a 2-dimensional transaction database. The dimensional attribute values of D_1 and D_2 have been mapped to integers; and there are four types of events, a, b, c, and d. The database contains transactions: $T_1(1, 1, a, b, c)$, $T_2(2, 1, b)$, $\cdots$, $T_{24}(5, 5, c)$. From the database, we can identify such event instances as $a(1, 1)$, $b(1, 1)$, $c(1, 1)$, $b(2, 1)$, $c(5, 5)$, and so on.

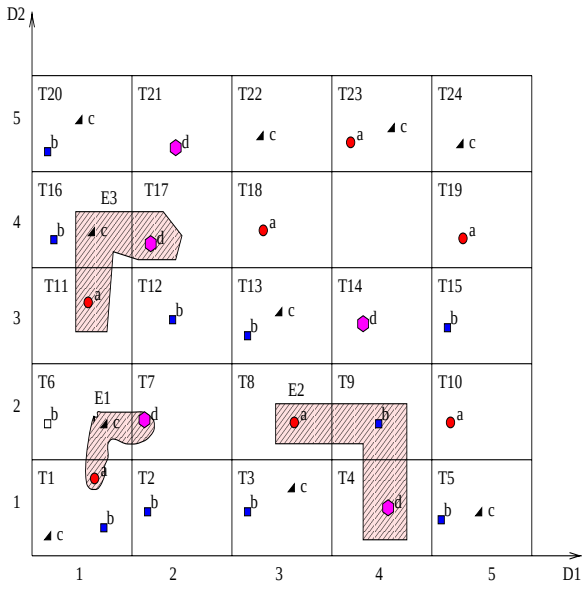

Figure 1: Graphical representation of a 2-dimensional transaction database.

N-dimensional inter-transaction association rules

The objective of inter-transaction association rules is to represent the associations between various events found in different transactions. With the introduction of dimensional attributes, we lose the luxury of simple representational form of the classical association rules. Some definitions are needed before we formally define such rules.

Definition 3 *Given a set of event instances $\bar{E} = \{\bar{e}_1, \bar{e}_2, \ldots, \bar{e}_m\}$ where $\bar{e}_i$ is in the form of $e_i\ (d_{i_1}, d_{i_2}, \ldots, d_{i_n})$ $(1 \leq i \leq m)$. An n-ary tuple $(d_{0_1}, d_{0_2}, \ldots, d_{0_n})$ with $d_{0_k} = Min(d_{i_k})$ $(1 \leq k \leq n, 1 \leq i \leq m)$ is called the base address for event instance set $\bar{E}$, denoted by E-BASE($\bar{E}$). $(d_{i_1} - d_{0_1}, d_{i_2} - d_{0_2}, \ldots, d_{i_n} - d_{0_n})$ is the relative address of $\bar{e}_i$ to the base address. The relative addresses of all member event instances in set $\bar{E}$ form the address of event instance set $\bar{E}$, denoted by E-ADDR($\bar{E}$).*

In Figure 1, there are three shadowed areas indicating three sets of event instances:
$\bar{E}_1 = \{a(1,1), c(1,2), d(2,2)\}$, $\bar{E}_2 = \{a(3,2), b(4,2), d(4,1)\}$, and $\bar{E}_3 = \{a(1,3), c(1,4), d(2,4)\}$. According to the above definition, we have E-BASE($\bar{E}_1$) = (1,1), E-BASE($\bar{E}_2$) = (3, 1), and E-BASE($\bar{E}_3$) = (1,3). The addresses for $\bar{E}_1$ and $\bar{E}_2$ are E-ADDR($\bar{E}_1$) = {(0,0), (0,1), (1,1)} and E-ADDR($\bar{E}_2$) ={(0,1), (1,1), (1,0)}, respectively. Since two events in a set may have the same address, the address of the set will have those

duplicates removed. For example, referring to Figure 1, the address of event instance set {a(1,1), b(1,1), c(1,2), d(2,2)} will be {(0,0), (0,1), (1,1)}. Note that, by using relative address, two sets of event instances may have the same address with respect to their base addresses. For set $\bar{E}_3$ = {a(1,3), c(1,4), d(2,4)}, E-BASE($\bar{E}_3$) = (1,3) and E-ADDR($\bar{E}_3$) = {(0,0), (0,1), (1,1)}, which is the same as E-ADDR($\bar{E}_1$). In other words, event instance sets $\bar{E}_1$ and $\bar{E}_3$ have the same address but different base addresses.

The notion of base address and address can be extended to the set of transactions.

Definition 4 *Given a set of transactions* $T = \{T_1, T_2, \ldots, T_s\}$ *where transaction* T_j *is in the form of* $(d_{j_1}, d_{j_2}, \ldots, d_{j_n}, E_j)$ $(1 \leq j \leq s)$. *An n-ary tuple* $(d_{0_1}, d_{0_2}, \ldots, d_{0_n})$ *with* $d_{0_k} = Min(d_{j_k})$ $(1 \leq k \leq n,\ 1 \leq j \leq s)$ *is called the base address for transaction set T, denoted by T-BASE(T). The relative addresses of all member transactions in T form the address of transaction set T, denoted by T-ADDR(T).*

In our example, for transaction set $T = \{T_1, T_2, T_6\}$ and $T' = \{T_4, T_8, T_9\}$, T-BASE(T) = (1, 1) and T-BASE(T') = (3, 1). The addresses of these two transaction sets are T-ADDR(T)={(0,0), (1,0), (0,1)} and T-ADDR(T')={(1,0), (0,1), (1,1)}, respectively.

Definition 5 *Given a set of transactions* $T = \{T_1, T_2, \cdots, T_s\}$ *where* T_j *is in the form of* $(d_{j_1}, d_{j_2}, \cdots, d_{j_n}, E_j)$ $(1 \leq j \leq s)$, *and a set of event instances* $\bar{E}_T = \{\bar{e}_1, \bar{e}_2, \cdots, \bar{e}_m\}$ *where* $\bar{e}_i$ *is in the form of* e_i $(d_{i_1}, d_{i_2}, \cdots, d_{i_n})$ $(1 \leq i \leq m)$. *T is said to contain* $\bar{E}_T$ *if (1) for every* $\bar{e}_i \in \bar{E}_T$, *there exists a transaction* $T_j \in T$ *so that* $e_i \in E_j$, *and the relative address of* $\bar{e}_i$ *in E-ADDR(*$\bar{E}_T$*) is the same as the relative address of* T_j *in T-ADDR(T).*
(2) $|$*E-ADDR(*$\bar{E}_T$*)*$| = |$*T-ADDR(T)*$|$ [1].

In the definition, the first condition guarantees that each event is among certain event list of a record in the transaction database. The second condition requires the transaction set is a minimum set. In our example, transaction set $\{T_1, T_6, T_7\}$ contains event instance set {a(0,0), c(0,1), d(1,1)}. $\{T_{11}, T_{16}, T_{17}\}$ and $\{T_8, T_{13}, T_{14}\}$ contain the same set of event instances.

Now we are ready to define n-dimensional inter-transaction association rules.

Definition 6 *An inter-transaction association rule is an implication of the form* $X \Rightarrow Y$, *where (1) X and Y are sets of event instances in the form of* $e_i(d_{i_1}, d_{i_2}, \cdots, d_{i_n})$ *where* $(d_{i_1}, d_{i_2}, \cdots, d_{i_n})$ *is the address of* e_i *relative to E-BASE(*$X \cup Y$*), i.e.,* $(d_{0_1}, d_{0_2}, \cdots, d_{0_n})$; *(2)* $e_i \in E$, $d_{0_k} \in Dom(D_k)$, $(d_{i_k} + d_{0_k}) \in Dom(D_k)$ $(1 \leq i \leq u,\ 1 \leq k \leq n)$, *and (3)* $X \cap Y = \emptyset$.

[1] $|S|$ denotes the cardinality of the set S.

For the database shown in Figure 1, one such association rule is

$$a(0,0), c(0,1) \Rightarrow d(1,1).$$

Since the inter-transaction association rules involve more than one transaction, the definitions of *support* and *confidence*, which are widely used as the objective interestingness measure of association rules in intra-transaction association rules, need to be modified. The reason is that, the number of transactions in the database can no longer be used as the measure. To address the problem, we introduce the following notion.

Definition 7 *A set of transactions* $T = \{T_1, T_2, \cdots, T_n\}$ *is said to possibly contain event instance set* $\bar{E}_T = \{\bar{e}_1, \bar{e}_2, \cdots, \bar{e}_m\}$, *if and only if T-ADDR(T) = E-ADDR(*$\bar{E}_T$*).*

In the above example, both transaction sets $\{T_1, T_6, T_7\}$ and $\{T_2, T_7, T_8\}$ possibly contain the event instance set $\{a(0,0), b(0,0), c(0,1), d(1,1)\}$ since both transaction sets have address {(0,0), (0,1), (1,1)} (although with different base address) which is the same as the address of the event instance set.

Definition 8 *Let* T_{xy} *be the set of transaction sets containing event instance set* $X \cup Y$, $\mathcal{T}_{xy}$ *be the set of transaction sets that possibly contain* $X \cup Y$, *and* T_x *be the set of transaction sets containing X, the support and confidence of an inter-transaction association rule* $X \Rightarrow Y$ *are defined as*

$$support = \frac{|T_{xy}|}{|\mathcal{T}_{xy}|}, \qquad confidence = \frac{|T_{xy}|}{|\mathcal{T}'_{xy}|}$$

respectively, where $\mathcal{T}'_{xy} = \{\tau \mid (\tau \in \mathcal{T}_{xy}) \wedge \exists \iota\ (\iota \in T_x)\ (\iota \subset \tau)\}$

As an example, we compute the support and confidence of the association rule

$$a(0,0), c(0,1) \Rightarrow d(1,1).$$

in database shown in Figure 1. Here, $X = \{a(0,0), c(0,1)\}$ and $Y = \{d(1,1)\}$. There are three transaction sets that contain the event instance set $X \cup Y$: $T_{xy} = \{\{T_1, T_6, T_7\}, \{T_8, T_{13}, T_{14}\}, \{T_{11}, T_{16}, T_{17}\}\}$, $|T_{xy}| = 3$. The transaction database contains 24 records. The number of transaction sets that possibly contain $X \cup Y$ is $|\mathcal{T}_{xy}| = 13$. Note that, the database does not contain any transactions with address (4,4), which reduces the number of transaction sets that possibly contain the event instance set. In addition to the transaction sets in T_{xy}, $\{T_{18}, T_{22}, T_{23}\}$ is a transaction set that possibly contains $X \cup Y$ and surely contains X: $\mathcal{T}'_{xy} =$

$\{\{T_1, T_6, T_7\}, \{T_8, T_{13}, T_{14}\}, \{T_{11}, T_{16}, T_{17}\}, \{T_{18}, T_{22}, T_{23}\}\}$, $|\mathcal{T}'_{xy}| = 4$. Therefore, the support and confidence for the above rule is 3/13 and 3/4, respectively. Note that we do not count the event a and c in transaction T_{19} and T_{24} when computing the confidence, as no transaction set can be formed with T_{19} and T_{24} that possibly contains $X \cup Y$.

3 Mining 1-dimensional inter-transaction association rules

Mining n-dimensional inter-transaction rules is obviously a computation intensive problem. Comparing to the classical association rules, the search space is much bigger as the number of possible rules increases dramatically with both the number of transactions and the number of dimensions.

To investigate the feasibility of mining inter-transaction rules, we implemented two algorithms by extending the Apriori-based algorithm to mine 1-dimensional inter-transaction association rules and applied it to the problem of stock price movement prediction. To limit the search space, we used an additional mining parameter, $MAX_INTERVAL$, to define a *sliding window*. Only the associations among the events that co-occurred within the window are interested.

In general, the mining process of n-dimensional inter-transaction rules can be divided into three phases: *data preparation*, *large itemset discovery*, and *association rule generation*.

Data preparation

The transaction database is prepared for mining from operational databases. The major task in this phase is to organize the transactions based on intervals of the dimensional attribute(s). For example, to find the long term movement regularities of stock prices across different weeks (months), we need to transform daily price movement into weekly (monthly) group. After such transformation, each record in the database will contain an interval value and a list of items.

Frequent-Itemset discovery

In this phase, we find the set of all frequent itemsets. A k-itemset is of the form $\{i_1(d_{i_1}), i_2(d_{i_2}), \ldots, i_k(d_{i_k})\}$, where event i_j, $1 \leq j \leq k$, is attached by a non-negative value d_{i_j} indicating the relative address with respect to the base address of the set. For example, a 3-itemset $\{a(0), b(1), c(3)\}$ contains three event instances expressed in relative addresses along the dimension. That is, taking a transaction containing event a as the base transaction, $b(1)$ is an event b contained in a transaction with 1 unit distance away from the base transaction, and $c(3)$ represents an event c in a transaction 3 unit distances away from the base transaction. This is quite different from the classical definition of itemset $\{i_1, i_2, \ldots, i_k\}$ in which all items lie within the same transactions.

To find the frequent item sets, two algorithms, *E-Apriori* and *EH-Apriori*, were implemented which are extensions of Apriori based algorithms [2, 8]. Let L_k represent the set of frequent k-itemsets, and C_k the set of candidate k-itemsets. Both algorithms make multiple passes over the database. Each pass consists of two phases. First, the set of all frequent (k-1)-itemsets L_{k-1}, found in the (k-1)th pass, is used to generate the candidate itemset C_k. The candidate generation procedure ensures that C_k is a superset of the set of all frequent k-itemsets. The algorithms now scan the database. For each list of consecutive transactions, they determine which candidates in C_k are contained and increment their counts. At the end of the pass, C_k is examined to check which of the candidates are actually frequent, yielding L_k. The algorithms terminate when L_k becomes empty.

As previously reported in [8], the processing cost of the first two iterations (i.e., obtaining L_1 and L_2) dominates the total mining cost. The reason is that, for a given minimum support, we usually have a very large L_1, which in turn results in a huge number of itemsets in C_2 to process. In the inter-transaction association rules, this situation becomes much more serious as a lot of additional 2-itemsets like $\{a(0), a(1)\}$ may be added into C_2, thus leading to a huge amount of $|C_2|$. In order to construct a significantly smaller C_2, *EH-Apriori* adopts a similar technique of hashing as [8] to filter out unnecessary candidate 2-itemsets. When the support of candidate C_1 is counted by scanning the database, *EH-Apriori* accumulates information about candidate 2-itemsets in advance in such a way that all possible 2-itemsets are hashed to a hash table. Each bucket in the hash table consists of a number to represent how many itemsets have been hashed to this bucket thus far. Such resulting hash table can be used to greatly reduce the number of 2-itemsets in C_2.

In the following, we describe how *E-Apriori* and *EH-Apriori* generate candidates and count their supports.

Candidate Generation

. Pass 1

Let $I = \{1, 2, ..., m\}$ be a set of items in a database. To generate the candidate set C_1 of 1-itemsets, we need to associate all possible intervals with each item. That is,

$$C_1 = \{ \begin{array}{l} 1(0), 1(1), \ldots, 1(MAX\_INTERVAL), \\ 2(0), 2(1), \ldots, 2(MAX\_INTERVAL), \\ \ldots \\ m(0), m(1), \ldots, m(MAX\_INTERVAL) \end{array} \}$$

Starting from transaction T_c $(1 \leq c \leq |D|)$, T_{c+d_i} transaction is scanned to determine whether item i exist. If so, the count of $\{i(d_i)\}$ increases by one. Through one scan of the database, we can get the large set L_1.

. Pass 2

For any two 1-itemsets of L_1, $a(0)$ and $b(d_b)$ $((d_b = 0 \wedge a < b) \vee (d_b \neq 0))$, we generate a 2-itemset $\{a(0), b(d_b)\}$, That is, $C_2 = \{\{a(0), b(d_b)\} | (d_b \neq 0) \vee (d_b = 0 \wedge a < b)\}$. Of all 2-itemsets in C_2, the minimal interval value is always 0.

. Pass k > 2

Given L_{k-1}, the set of all frequent (k-1)-itemsets, the candidate generation procedure returns a superset of the set of all frequent k-itemsets. This procedure has two parts. In the *join* phase, we join L_{k-1} with L_{k-1}:

insertinto	C_k
select	$p.item_1(d_{item_1}), p.item_2(d_{item_2}), \ \ldots\ ,$
	$p.item_{k-1}(d_{item_{k-1}}), q.item_{k-1}(d_{item_{k-1}})$
from	$L_{k-1}\ p,\ L_{k-1}\ q$
where	$p.item_1(d_{item_1}) = q.item_1(d_{item_1}), \ \ldots\ ,$
	$p.item_{k-2}(d_{item_{k-2}}) = q.item_{k-2}(d_{item_{k-2}}),$
	$p.item_{k-1}(d_{item_{k-1}}) < q.item_{k-1}(d_{item_{k-1}}).$

We define the comparison operators "=" and "<" between two item-interval pairs as follows:

Definition 9 $item_i(d_{item_i}) = item_j(d_{item_j})$ if and only if both of the conditions hold:

(1) $item_i = item_j$ (2) $d_{item_i} = d_{item_j}$.

Definition 10 $item_i(d_{item_i}) < item_j(d_{item_j})$ if and only if either of the conditions holds:

(1) $d_{item_i} < d_{item_j}$

(2) $(d_{item_i} = d_{item_j}) \wedge (item_i < item_j)$.

All itemsets $c \in C_k$, which have some (k-1)-subsets with the support less than the threshold are deleted in the pruning phase. Furthermore, addresses of all events in the frequent set are converted to relative address if the minimum value of the dimensional attribute of the events in the set is not 0.

Counting Support of Candidates

To facilitate the efficient support counting process, a candidate C_k of k-itemsets is divided into k groups, with each group G_o containing o number of items whose interval is 0 $(1 \leq o \leq k)$. For example, a 3-item set

$$C_3 = \{ \begin{array}{l} \{a(0), a(1), b(2)\}, \{c(0), d(0), d(2)\}, \\ \{a(0), b(0), h(3)\}, \{l(0), m(0), n(0)\}, \\ \{p(0), q(0), r(0)\} \ \} \end{array}$$

is divided into three groups:

$G_1 = \{\{a(0), a(1), b(2)\}\}$,

$G_2 = \{\{c(0), d(0), d(2)\}, \{a(0), b(0), h(3)\}\}$,

$G_3 = \{\{l(0), m(0), n(0)\}, \{p(0), q(0), r(0)\}\}$.

Each group is stored in a modified *hash-tree*. Only those items with interval 0 participate the construction of this hash-tree, e.g., in group 2, only $\{a(0), b(0)\}$, $\{c(0), d(0)\}$ enter the hash-tree. The construction process is similar to that of Apriori [2]. The rest items, e.g. $h(3)$, $d(2)$, are simply attached to the corresponding itemsets, e.g., $\{a(0), b(0)\}$ and $\{c(0), d(0)\}$ respectively, in the leaves of the tree.

Upon reading one transaction of the database, every hash-tree is tested. If one itemset is contained, its attached itemsets whose intervals are larger than 0 will be checked against the successive transactions. In the above example, if $\{a(0), b(0)\}$ exists in the current transaction t_c, then t_{c+3} transaction will be scanned to see whether it contains item h. If so, the support of 3-itemset $\{a(0), b(0), h(3)\}$ will increase by 1.

E-Apriori and *EH-Apriori* share the same procedures, except that in Pass 1, *EH-Apriori* hashes all 2-itemsets like $\{i_1(0), i_2(d_{i_2})\}$ $(d_{i_2} \neq 0)$ contained in the current series of transactions into the corresponding buckets of a *HashTable* [2] and prunes unnecessary 2-itemsets from C_2 in pass 2, whose corresponding bucket values in the *HashTable* are less than support threshold.

Association rule generation

Using sets of frequent itemsets, we can find the desired inter-transaction association rules. The generation of inter-transaction association rules is similar to the generation of the classical association rules, except the calculation of rules' *confidence*s as mentioned in the previous section.

Experimental results

To assess the performance of the proposed algorithms, some preliminary experiments were conducted using synthetic data. Table 1 listed one set of the results obtained using a transaction database with 10,000 records with each records containing 5 items on the average. The total number of items is 500. The maximum interval is set to 3. (T - ave_tran_size, N -

[2] The hash function used here is $\text{func}(\{i_1(0), i_2(d_{i_2})\}) = i_1 + 10 * i_2^{d_{i_2}}$.

T5-R3-N500-D10k				
	support=0.6%		support=0.7%	
	E-Apriori	EH-Apriori	E-Apriori	EH-Apriori
$\|L_1\|$	348	348	319	319
t_1	0.4 s	3.9 s	0.4 s	3.9 s
$\|C_2\|$	363312	43047	305283	17403
$\|L_2\|$	86	86	29	29
t_2	537.6 s	65.5 s	224.8 s	18.1 s
$t_1 + t_2$	538.0 s	69.4 s	225.2 s	22.0 s
Total Mining Time	539.0 s	70.4 s	226.2 s	23.0 s

Table 1: Comparison of E-Apriori and EH-Apriori

item_num, D - tran_num, R - max_interval). The results indicate that, with the given setting, the execution time is acceptable, especially if *EH-Apriori* algorithm is used. It is also found that although the execution time of the first pass of *EH-Apriori* is slightly longer than that of *E-Apriori* due to the extra overhead required for building $HashTable$, it incurs significantly smaller execution time than *E-Apriori* in later Pass 2, and less $|C_2|$ results in much less time to test against each transaction of the database.

Some tests also conducted using the data set collected from Singapore Stock Exchange (SES). The available stock price data was used to generate two data sets, WINNER and LOSER. A stock is a winner if its closing price of the day is 3% more than the previous day closing. A stock is a loser otherwise. The WINNER(LOSSER) data set contains the date and the winners(losers) of that day. Each data set contains 250 records corresponding to 250 trading days in 1996. Since the major trend for SES in 1996 is down side, there are a few of winners everyday but a large number of losers. From the LOSER set, one example rule found is $\{UOL(0), SIA(1)\} \Rightarrow DBS(2)$. That is, if UOL goes down and SIA goes down the following day, DBS will go down the second day with confidence more than 99%. Since the WINNER data set is small, we do not have rules with large support. However, if after lowering the support, we can find rules such as $\{HAISUNWT(0), KIMENGWT(0)\} \Rightarrow HAISUNWT(1)$.

4 Discussions

The necessity of having N-dimensional inter-transaction association rules is clear. The definition of such rules is lengthy (based on our study). However, we believe that, the proposed n-dimension inter-transaction association rules represent a uniform treatment to a few association-related data mining problems. Furthermore, there seems to be highly promising to apply such notions in textual mining, spatial data mining, multi-media data mining, etc.

A general view of association rules

The problem defined here gives a general view of associations among events.

- **Traditional association rule mining:** If the dimensional attributes are not present in the knowledge to be discovered, or those attributes are not inter-related, then they can be ignored in analysis, and the problem becomes the traditional transaction-based association rules.
- **Mining in multi-dimensional databases or data warehouse:** Multi-dimensional database organizes data into data cubes with dimensional attributes and measures [6]. Some of the dimensions can be ignored and the measures can be collapsed to form a transaction database as defined in this paper.
- **Mining spatial association rules:** Existence of spatial objects can be viewed as events. The coordinates are dimensional attributes. Related properties can be added as other dimensional attributes.

Mining n-dimensional inter-transaction association rules

Mining n-dimensional inter-transaction association rules is more complex than mining classical association rules. One of the major difficulties is the much larger search space, and also the much larger number of possible rules to be generated, than that in the classical association rule case.

To reduce the search space, one approach is to define a *maximum span* along dimensional attributes as mining parameters. In 1-dimensional case, it works like a slide window. Only those rules covered by the window is considered, which thus limits the number of possible rules. This is also reasonable: in stock movement

prediction, the span represents the interests of a user on expected number of days between the predicted rises of price relative to current date. A preliminary study on mining 1-dimensional inter-transaction association indicated that the traditional association rule mining algorithms can be extended to mine inter-transaction association rules within a reasonable amount of time [4].

Another possible source of difficulty is that, strictly speaking, the item the nice Apriori property in classical association rule mining that all the subsets of a frequent item set must also be frequent may not hold since in the definition of *support* and *confidence*, the denominator varies when the cardinality of the frequent sets varies. For example, in a database with three transactions: $T_1(0, a), T_2(2, b)$ and $T_3(3, c)$, the set {a(0), c(3)} has support of 100%. Obviously, its subset {a(0)} does not have such high support. Therefore, the property on which most traditional association rule mining algorithms are based does not hold in this case. However, if the number of transactions in the database is very large, and the window size used in mining process is relatively small, we may be still able to use the property as the base of mining algorithms.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 207–216, Washington D.C., USA, May 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th Conference on Very Large Data Bases*, pages 478–499, Santiago, Chile, September 1994.

[3] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. of the International Conference on Data Engineering*, Taipei, Taiwan, march 1995.

[4] L. Feng, H. Lu, and J. Han. Beyond intra-transaction association analysis: Mining multi-dimensional inter-transaction association rules. Submitted for publication, February 1998.

[5] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proc. of the 21st Conference on Very Large Data Bases*, pages 420–431, Zurich, Switzerland, September 1995.

[6] M. Kamber, J. Han, and J.Y. Chiang. Metarule-guided mining of multi-dimensional association rules using data cubes. In *Proc. of the International Conference on Knowledge Discovery and Data Mining*, pages 207–210, California, USA, August 1997.

[7] R.J. Miller and Y. Yang. Association rules over interval data. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 452–461, Tucson Arizona, USA, May 1997.

[8] J.-S. Park, M.-S. Chen, and P.S. Yu. An effective hash based algorithm for mining association rules. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 175–186, San Jose, CA, May 1995.

[9] R. Srikant and R. Agrawal. Mining generalized association rules. In *Proc. of the 21st Conference on Very Large Data Bases*, pages 409–419, Zurich, Switzerland, September 1995.

[10] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 1–12, Montreal, Canada, June 1996.