Mining strongly correlated item pairs in large transaction databases

Swarup Roy*
Department of Information Technology, North-Eastern Hill University, Umshing, Shillong 793022, Meghalaya, India
E-mail: swarup@nehu.ac.in
E-mail: contact.swarup@gmail.com
*Corresponding author

Dhruba Kr Bhattacharyya
Department of Computer Science and Engineering, Tezpur University, Napaam, Tezpur 784028, Assam, India
E-mail: dkb@tezu.ernet.in

Abstract: Correlation mining is an approach of drawing statistical relationship between items from transaction data. Most of the existing techniques use Pearson’s correlation coefficient as a measure of correlation, which may not always perform well when data are noisy and binary in nature. Moreover, they require multi-pass over the database. This paper presents an effective and faster correlation mining technique to extract most strongly correlated item pairs from large transaction databases. As an alternative to Pearson’s correlation coefficient, it presents a method of computing Spearman’s rank order correlation coefficient from transaction data. The proposed technique found to perform satisfactorily in terms of execution time over several real and synthetic datasets, while comparing to other similar techniques. To justify its usefulness, an application of the proposed technique for extracting yeast genetic network from gene expression data is also reported.

Keywords: correlation mining; correlation coefficient; strongly correlated item pairs; support; Spearman’s rank order correlation; transaction database.


Biographical notes: Swarup Roy is an Assistant Professor of Information Technology at North Eastern Hill University, Shillong, India. He received his MTech in 2005. He is pursuing his PhD in Computer Science and Engineering from Tezpur University. He has published a number of papers in different international journals and peer-reviewed conference proceedings and authored a book. His research interest includes data mining and computational biology.

Dhruba Kr Bhattacharyya is a Professor of Computer Science and Engineering at Tezpur University, Tezpur, India. He received his PhD in 1999 in the field of cryptography and error control coding from the same university. He has more
Traditional association rule mining technique (Agrawal et al., 1993; Han and Kamber, 2006) is based on support-confidence framework. However, support-confidence framework can be misleading; it can identify a rule \( A \Rightarrow B \) as interesting (strong) when in fact, the occurrence of \( A \) might not imply the occurrence of \( B \). Thus, the support and confidence measures are insufficient in filtering out the uninteresting association rules (Brin et al., 1997; Han and Kamber, 2006). It has been well observed that item pairs with high support value may not imply highly correlated. Similarly, a highly correlated item pair may exhibit low support value. To tackle this weakness, a correlation analysis can be used to provide an alternative framework for finding statistically interesting relationships (Brin et al., 1997). It also helps to improve the understanding of some association rules.

Besides providing a statistical meaning of the traditional association mining problem, correlation mining could play a major role in addressing various issues like how sales of a product is associated with the sales of other products which in turn may help in designing sales promotions, catalog design and store layout. It could be helpful in efficient finding of co-citations and term co-occurrences during document analysis. Functional relationship (Kuo et al., 2003) between pairs of genes based on gene expression profiles and their changes in different diseases and conditions may be indicative in determining disease mechanism for diseases like cancer. It has been observed that a simple pair-wise correlation analysis may be helpful in revealing new gene-gene relationship (Slonim, 2002; Butte and Kohane, 2000) which again in turn useful in discovering gene regulatory pathways or gene interaction network.

Xiong et al. (2004), introduced the notion of strongly correlated item pairs, which retrieves all strongly correlated item pairs from transaction database based on user specified \( \theta \) threshold. Later on, a number of techniques have been proposed (He et al., 2005; Jiang et al., 2007; Li and Lang, 2010) to handle this problem.

TAPER (Xiong et al., 2004) is a candidate generation-based technique for finding all strongly correlated item pairs. It employs an effective upper bound-based pruning technique to eliminate large number of item pairs, which are not strongly correlated during candidate generation phase. In case of dense and high dimensional large databases, candidate generation-based techniques generate large number of potential candidates and testing of those candidates is quite expensive. To address this problem, a new FP-tree-based technique (Han et al., 2000), Tcp (He et al., 2005) was proposed to generate strongly correlated item pairs without candidate generation. Tcp constructs
large FP-tree, which sometime difficult to accommodate in the memory and requires much time, especially when the size of the transaction database, in terms of both number of records as well as number of items, becomes very large.

In finding strongly correlated item pairs, a major challenge is to determine the appropriate threshold value, say $\theta$, which requires a prior knowledge about the data. To address this issue an alternative top-$k$ correlated-pairs query problem has been introduced recently to mine $k$ topmost strongly correlated pairs instead of computing all strongly correlated pairs based on $\theta$. Most of the techniques proposed so far are the extension of the strongly correlated item pairs query techniques. Xiong et al. (2008) proposed TOP-COP, which is basically a candidate generation-based technique like TAPER and generates top-$k$ strongly correlated item pairs by pruning and filtering non-potential strong pairs from the candidate sets. Parallely, He et al. (2006) extends Tcp to extract top-$k$ strongly correlated item pairs using FP-tree-based approach. TOP-COP and Tkcp are also not free from the limitations as suffered by TAPER and Tcp.

Most of the existing techniques use Pearson’s correlation coefficient, $\phi$, as a measure of correlation between item pairs. However, Pearson’s correlation coefficient may not always perform well when data are noisy, binary and non-linear in nature. Spearman’s correlation coefficient, $\rho$, can be an alternative measure for computing correlation coefficient between an item pair. In this paper, we present a technique for computing correlation coefficient between an item pair from market basket data using Spearman’s $\rho$.

Existing correlated pair finding techniques require multi-pass over the database which is too costly for large transaction databases. It would be more effective if both strongly correlated pairs as well as top-$k$ strongly correlated item pairs can be extracted using a single pass over the database, without generating any large tree or candidate itemsets.

To address the above issue, this paper presents two faster and effective techniques,

1. SCOPE, which extracts all the strongly correlated item pairs, for any large database
2. $k$-SCOPE an extension to SCOPE, to extract top-$k$ strongly correlated item pairs in only one pass over the database and without generating any candidate set.

A preliminary version of both the work can be found in Roy and Bhattacharyya (2008a, 2008b). This technique uses a data structure, i.e., a correlogram matrix for capturing the frequency count of all the 1- and 2-element item sets using single pass over the database. Next, it computes the correlation coefficient of all the item pairs based on the support count from correlogram matrix and extracts all the strongly correlated item pairs and also creates a list of $k$ top most strongly correlated item pairs based on the support count from correlogram matrix. Proposed technique has been established to perform faster, in comparison to other techniques, in light of several real and synthetic transaction datasets.

The rest of the paper is organised as follows: Section 2 reports the background of the work. Section 3 presents a technique to compute Spearman’s correlation coefficient $\rho$, between an item pair. Section 4 introduces the strongly correlated item pair finding problem. Section 5 presents the proposed techniques and Section 6 demonstrates the performance of the proposed techniques in the light of both synthetic and real datasets. Section 7 reports a potential application of the proposed technique and finally in Section 8, the concluding remarks are given.
2 Background

Starting from the market basket data analysis, association mining now spreads its spectrum of applications in different domains like machine learning, soft-computing, computational biology and so on. Standard association mining technique is to extract all subsets of items satisfying minimum support constraint. Unlike traditional association mining, the all-pair-strongly correlated query is to find a statistical relationships among pair of items from transaction database. In statistics, relationships among nominal variables can be analysed with nominal measures of association such as Pearson’s correlation coefficient and measures based on chi-square (Reynolds, 1977). The correlation coefficient (Reynolds, 1977) is the computational form of Pearson’s correlation coefficient for binary variables. An equivalent support measure-based $\phi$ correlation coefficient computation technique has been introduced in Xiong et al. (2004) to find correlation of item pairs in a transaction database based on their support count. For any two items $X$ and $Y$ in a transaction database, the support-based $\phi$ correlation coefficient can be calculated as:

$$\phi = \frac{\text{Sup}(X,Y) - \text{Sup}(X) \times \text{Sup}(Y)}{\sqrt{\text{Sup}(X) \times \text{Sup}(Y) \times (1 - \text{Sup}(X)) \times (1 - \text{Sup}(Y))}}$$

(1)

where $\text{sup}(X)$, $\text{sup}(Y)$ and $\text{sup}(X,Y)$ are the support of item $X$, $Y$ and joint support of $(X,Y)$ respectively.

Majority of the correlation mining techniques use above Pearson’s correlation coefficient for finding strongly correlated item pairs, which is a parametric technique and works fine with continuous variables. Since typical market basket data are binary in nature, parametric approach may not always perform well for binary data. Further, parametric correlation coefficient is sensitive to outliers in a dataset. Non-parametric techniques like Spearman’s $\rho$ (Lehmann and D’Abrera, 1998) can be found to be a better choice in this regard. Below, we propose an approach for computing Spearman’s $\rho$ between an item pair from market basket data.

3 Computing Spearman’s rank order correlation

Parametric techniques like Pearson’s correlation are sensitive to the data distribution (Katz, 2002; Corder and Foreman, 2009). Performance of parametric techniques may not be effective when data are noisy and binary in nature. The alternative solution is to apply non-parametric correlation. If two variables $X$ and $Y$ are metric (i.e., interval or ratio scale measures) and they are to be correlated, then the parametric technique like Pearson’s correlation coefficient is considered as suitable. While desirable, it is not always possible to use a parametric test such as the Pearson’s method. In case of non-parametric (i.e., nominal or ordinal measures) variables, correlation can be determined effectively by the use of one of the non-parametric correlation techniques. Non-parametric correlation coefficients, such as chi-square, point biserial correlation (Gravetter and Wallnau, 2009), Spearman’s $\rho$ (Lehmann and D’Abrera, 1998), Kendall’s $\tau$ (Kendall and Gibbons, 1990), and Goodman and Kruskal’s $\lambda$ (Goodman and Kruskal, 1954) may perform better than the parametric correlation coefficient when outliers are present. The most frequently used one is the rank
order-based Spearman’s $\rho$ correlation. In principle, Spearman’s $\rho$ is simply a special case of the Pearson’s product-moment coefficient in which two sets of data $X_i$’s and $Y_i$’s are converted to rankings $x_i$’s and $y_i$’s before calculating the coefficients. The raw scores are converted to ranks, and the differences $D_i$’s between the ranks of each observation on the two variables are calculated. If there are no tied ranks, then $\rho$ is given by (Lehmann and D’Abrera, 1998):

$$
\rho = 1 - \frac{6 \sum D_i^2}{N(N^2 - 1)}
$$

(2)

where $D_i = x_i - y_i$, is the difference between the ranks of corresponding values $X_i$ and $Y_i$, and $N$ is the number of samples in each dataset (same for both sets).

If tied ranks exist, each tied score is assigned a rank equal to the average of all tied positions. For example, if a pair of scores are tied for the 2nd and 3rd rank, both scores are assigned a rank of 2.5 ($\frac{2 + 3}{2} = 2.5$). In case of binary variables a large number of tied cases are present. Binary market basket data contains score 0 and 1 only. In order to compute the ranking of 0 and 1, their natural ordering can be used to compute tied ranks. For computing the tied ranks, we assigned 0 with greater priority than 1 and used simple frequency count of 1 and 0 score. Below, we discuss the technique for calculating the Spearman’s $\rho$ with tied cases between binary variables.

Assuming a binary variable $I$ with $N$ values. The frequency of score 1 in variable $I$ is denoted as $f(I)$. To determine the appropriate rank of tied cases, we need to add the rank positions and divided by the number of tied cases. Since the number of scores in binary variable is only two, thus just by using the $f(I)$, the tied rank of score 0 and 1 can be calculated as:

$$
\text{Rank}_0 = \sum_{i=1}^{N-f(I)} \frac{i}{N-f(I)}
$$

(3)

Similarly, the rank of score 1 can be calculated as:

$$
\text{Rank}_1 = \sum_{i=f(I)}^{N} \frac{i}{f(I)}
$$

(4)

Once the ranks of 0 and 1 are calculated for the target variables (item pairs), say $I_1$ and $I_2$, whose correlation is to be calculated, the difference of their ranks is then used to compute $D_i^2$. In case of two binary item sets, the only possible combination of scores are (0, 0), (0, 1), (1, 0) and (1, 1). Thus just by counting the frequency of above patterns and using the rank of 0 and 1 for both the item sets, the sum of square differences of ranks ($D_i^2$) can be easily calculated as:

$$
\sum D_i^2 = P_{00} (\text{Rank}_0(I_1) - \text{Rank}_0(I_2))^2 + P_{01} (\text{Rank}_0(I_1) - \text{Rank}_1(I_2))^2 + P_{10} (\text{Rank}_1(I_1) - \text{Rank}_0(I_2))^2 + P_{11} (\text{Rank}_1(I_1) - \text{Rank}_1(I_2))^2
$$

(5)

where $P_{00}, P_{01}, P_{10}$ and $P_{11}$ are the frequency of (0, 0), (0, 1), (1, 0) and (1, 1) patterns respectively. $\text{Rank}_0(I_1)$ and $\text{Rank}_1(I_1)$ are the corresponding 0’s and 1’s rank
in item $I_1$ and $\text{Rank}_0(I_2)$ and $\text{Rank}_1(I_2)$ are the corresponding 0’s and 1’s rank in item $I_2$ respectively.

For a long sequence of binary data occurring in large transaction datasets, sometimes it is costly to find $P_{(00)}, P_{(01)}, P_{(10)}$ and $P_{(11)}$ for each pair of items. It would be more effective if that can be computed with minimal information. One possible way is given below, especially when frequency of score 1 in item $I_1$ and $I_2$ ($f(I_1)$ and $f(I_2)$) and joint occurrences in both item pairs ($f(I_1, I_2)$) are known.

$$
\begin{align*}
    P_{(01)} &= f(I_2) - f(I_1, I_2), \\
    P_{(10)} &= f(I_1) - f(I_1, I_2), \\
    P_{(11)} &= f(I_1, I_2), \\
    P_{(00)} &= N - (P_{(01)} + P_{(10)} + P_{(11)})
\end{align*}
$$

### 3.1 Computing Spearman’s $\rho$: an illustration

In this section, we will demonstrate how to compute Spearman’s $\rho$ with tied cases for binary market basket data. For example, let us consider the following item pairs $I_1$ and $I_2$ with $N = 6$ transactions ($T_1$, $T_2$, ... $T_6$) and having similar occurrence pattern.

<table>
<thead>
<tr>
<th></th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>$T_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$I_2$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

From Table 1, it is obvious that the frequency of 1 in $I_1$ and $I_2$, and joint occurrences of 1 in both $I_1$ and $I_2$ are $f(I_1) = 3$, $f(I_2) = 3$, $f(I_1, I_2) = 3$ respectively. Using equation (3) and (4) the value of $\text{Rank}_0(I_1)$ and $\text{Rank}_1(I_1)$ of $I_1$ would be $\text{Rank}_0(I_1) = (1 + 2 + 3)/3 = 2$ and $\text{Rank}_1(I_1) = (4 + 5 + 6)/3 = 5$ Similarly, $\text{Rank}_0(I_2)$ and $\text{Rank}_1(I_2)$ of $I_2$ would be 2 and 5 respectively.

The next step is to calculate $P_{(00)}, P_{(01)}, P_{(10)}$ and $P_{(11)}$ by using joint frequency count $f(I_1, I_2)$.

$$
\begin{align*}
    P_{(01)} &= f(I_2) - f(I_1, I_2) = 3 - 3 = 0, \\
    P_{(10)} &= f(I_1) - f(I_1, I_2) = 3 - 3 = 0, \\
    P_{(11)} &= f(I_1, I_2) = 3, \\
    P_{(00)} &= N - (P_{(01)} + P_{(10)} + P_{(11)}) = 6 - (0 + 0 + 3) = 3
\end{align*}
$$

The summation of square rank difference $D_i^2$ would be:

$$
\begin{align*}
    \sum D_i^2 &= P_{(00)}(\text{Rank}_0(I_1) - \text{Rank}_0(I_2))^2 \\
    &+ P_{(01)}(\text{Rank}_0(I_1) - \text{Rank}_1(I_2))^2 \\
    &+ P_{(10)}(\text{Rank}_1(I_1) - \text{Rank}_0(I_2))^2 \\
    &+ P_{(11)}(\text{Rank}_1(I_1) - \text{Rank}_1(I_2))^2 \\
    &= 3(2 - 2)^2 + 0(2 - 5)^2 + 0(5 - 2)^2 + 3(5 - 5)^2 \\
    &= 0
\end{align*}
$$
Thus, Spearman’s $\rho$ could be calculated as: $\rho = 1 - (6 \times 0)/(6(6^2 - 1)) = 1 - 0 = 1$.

In transaction database, computing $f(I_1)$, $f(I_2)$ and $f(I_1, I_2)$ for any item pair is nothing but finding their individual and joint supports (1- and 2-element item sets). Traditional approach for counting the support for item pair needs at least two scans over the entire dataset. In Section 5, we discuss a correlogram matrix-based technique that counts individual and joint supports for item pair in a single pass over the database. Using support count and the method discussed above, it is straightforward to compute the correlation coefficient between an item pair.

Next, we discuss the problem of finding strongly correlated item pairs formally with the help of suitable illustrations.

### 4 Finding strongly correlated item pairs

In this section we introduce the problem of finding strongly correlated item pairs. Apart from the Pearson’s correlation coefficient $\phi$ for finding the correlations in transaction databases, here we consider that correlation measures like Spearman’s rank order correlation or Chi square are also suitable for the same purpose. Thus, instead of restricting the problem to use only $\phi$ as a correlation measure, we try to generalise the definition and redefine the same as follows:

**Definition 1:** Given a user-specified minimum correlation threshold $\theta$ and a market basket database $D = \{T_1, T_2, \ldots, T_N\}$ of $N$ transactions where each transaction $T_i$ is a subset of itemset $I = \{X_1, X_2, \ldots, X_M\}$ of $M$ distinct items, a all-strong-pairs correlation query finds collection of all item pairs $(X_i, X_j)$ (for $i, j = 1 \ldots M$) with correlation coefficient $(\text{Corr})$ above or equal to threshold $\theta$. Formally, it can be defined as:

$$SC(D, \theta) = \{\{X_i, X_j\} | \{X_i, X_j\} \subseteq I, X_i \neq X_j \land \text{Corr}(X_i, X_j) \geq \theta \} \quad (6)$$

To determine the appropriate value of $\theta$, a prior knowledge about data distribution is required. Without specific knowledge about the target data, users will have difficulties in setting the correlation threshold to obtain their required results. If the correlation threshold is set too large, there may be only a small number of results or even no result. In such case, the user may have to guess a smaller threshold and do the mining again, however, it may generate too many results for the users; which may not be meaningful and can consume an exceedingly long time in the computation.

An alternative solution to this problem could be to change the task of mining under pre-specified threshold to mine top-$k$ strongly correlated item pairs, where $k$ is the desired number of item pairs that have largest correlation values. The problem of top-$k$ correlated pairs query problem can be defined as follows:

**Definition 2:** Given a user-specified threshold $k$ and a market basket database $D$ of $N$ transactions where each transaction $T_i$ is a subset of itemset $I$ of $M$ distinct items, a top-$k$ correlated-pair query finds $k$ number of item pairs $T_k = \{\{X_{i1}, X_{j1}\}, \ldots, \{X_{ik}, X_{jk}\}\}$ such that $\text{Corr}(X_{i1}, X_{j1}) \geq \text{Corr}(X_{i2}, X_{j2}) \geq \ldots \geq \text{Corr}(X_{ik}, X_{jk})$ and $\text{Corr}(X_{ik}, X_{jk}) \geq \text{Corr}(X_{il}, X_{jl}), \forall \{X_{il}, X_{jl}\} \subseteq I$ and $\{X_{il}, X_{jl}\} \notin T_k$.

Thus, top-$k$ is a sorted list of $k$ item pairs based on any suitable correlation coefficient $\text{Corr}$. Next we illustrate both the problems discussed above.
4.1 Computing support-based correlated pairs: an illustration

Based on the support measures of traditional association mining, the task of strongly correlated item pair finding is basically to generate a list of pairs from the database where \( \text{Corr} \) value of pair is greater than the user specified \( \theta \). Similarly, the task of top-\( k \) correlated-pair finding is to generate a sorted list of \( k \) pairs in the order of \( \text{Corr} \) from the database. An illustration of both the correlated-pairs query problems have been depicted in Figures 1 and 2. In the example, we have used \( \phi \) as correlation coefficient. The input to the strongly correlated query is a market basket database containing eight transactions and six items. The value of \( \theta \) is set to 0.05 as input. Similarly, for top-\( k \) problem the value of \( k \) is set to 8. Since the database has six items, there are \( \binom{6}{2} = 15 \) item pairs for which correlation coefficient \( \phi \) is to be calculated. To compute \( \phi(4, 5) \) using equation (1), we need the single element support, e.g., \( \text{sup}(4) = 4/8 \) and \( \text{sup}(5) = 3/8 \) and joint support \( \text{sup}(4, 5) = 3/8 \), in order to compute the exact correlation \( \phi(4, 5) \), which is 0.77. Finally, all those pairs are extracted that satisfy \( \theta \) constraint, and the list of strongly correlated pairs is generated as output. Similarly, the list of \( k \) most strongly correlated pairs is generated as an output for the second problem (irrespective of any \( \theta \) value).

**Figure 1** Illustration of strongly correlated pairs query problem

**Figure 2** Illustration of top-\( k \) correlated-pairs query problem
5 Proposed techniques

The proposed techniques attempt to find all the strongly correlated item pairs and $k$ top most correlated pairs from any transaction database using single scan over the database without generating any candidates. It uses a correlogram matrix to store the support counts of all 1- and 2-element item sets. Later, the matrix is used to calculate correlation coefficients of all the item pairs. The background of the proposed technique is reported next.

5.1 Correlogram matrix-based technique

Correlogram matrix is basically a co-occurrence frequency matrix. It is a matrix of size: $n \times (n + 1)/2$, where, $n$ is the number of items present in the database. Each cell of the matrix contains the frequency of co-occurrence of item pairs. Item pairs are specified by row index and column index of the cell. For example, the content of the cell $(4, 5)$ in the matrix (see Figure 3) with an index of row four and column five will indicate the co-occurrence frequency of the item pair $\{4, 5\}$. On the other hand, the diagonal cells having same index for row and column, indicates the frequency of the single element item set. Thus, as can be found in Figure 3, the cell $(3, 3)$ indicates the frequency of the item $3$.

Figure 3 Correlogram matrix

From the correlogram matrix, it is very simple to extract the support count of 1- and 2-element item sets. Using these support counts, next we compute the correlation coefficients of all the item pairs using equation (1) or (5) and extract all the item pairs which satisfy the given $\theta$ (i.e., correlation coefficient) constraint. In case of top-$k$ strongly correlated query, a sorted list of top-$k$ item pairs is created based on their correlation coefficient. The proposed technique can be found to be significant in view of the following points:

1 unlike the other techniques discussed so far, it requires only one database scan; once the correlogram matrix is constructed, no more consultation of the original database is required for the successive processing

2 it is highly scalable in terms of number of transactions and dimensionality
it gives a facility of interactive mining, i.e., computing the correlogram matrix once
and generating the different strongly correlated item pair list as well as top-k list
based on the $\theta$ and $k$ values, respectively.

there is no requirement for candidate generation.

5.2 The algorithms

The proposed technique, SCOPE accepts the market-basket database, i.e., $D$ and the
correlation coefficient threshold $\theta$ as input and it generates all the strongly correlated
item pairs as output. Step 1 of SCOPE, (see Table 2), is dedicated to the construction
of the correlogram matrix using single scan of the original database. In Step 2a, the
correlation coefficient of each item pair is computed and in Step 2b, all those item
pairs are extracted, whose coefficient values are greater than or equal to $\theta$. Finally, the
algorithm returns list of all strongly correlated item pairs.

Table 2  SCOPE algorithm

<table>
<thead>
<tr>
<th>Input:</th>
<th>$D$ (Original Database), $\theta$ (correlation coefficient threshold)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>$L$ (Strongly correlated item pairs)</td>
</tr>
<tr>
<td>1.</td>
<td>Generate Correlogram Matrix from $D$</td>
</tr>
<tr>
<td>2.</td>
<td>For each item pair $(i, j) \in D$</td>
</tr>
<tr>
<td></td>
<td>a. Compute Correlation coefficient, $Corr(i, j)$ by</td>
</tr>
<tr>
<td></td>
<td>using support count from the correlogram matrix.</td>
</tr>
<tr>
<td></td>
<td>b. if $Corr(i, j) \geq \theta$ then add item pair $(i, j)$ into</td>
</tr>
<tr>
<td></td>
<td>strongly correlated item pairs list $L$</td>
</tr>
<tr>
<td>3.</td>
<td>Return $L$</td>
</tr>
</tbody>
</table>

An extended version of this algorithm for generating top-$k$ correlated pairs, i.e.,
k-SCOPE is reported in Table 3. The algorithm accepts the market-basket database, i.e.,
$D$ and $k$ as inputs and it generates list of top-$k$ strongly correlated item pairs, $L$, as
outputs. The first phase of the algorithm is same as SCOPE. In Step 2b, topmost $k$
correlated item pairs are extracted and added to the top-$k$ list. Top-$k$ list $L$ is a sorted list
(descending order) of item pairs based on the correlation coefficient. For any pair whose
correlation coefficient lower than the $k^{th}$ pair’s correlation coefficient are straightway
pruned. Otherwise, it updates the list by eliminating the $k^{th}$ pair and inserting the new
pair in its appropriate position in the list. Finally, the algorithm returns top-$k$ list $L$.

Table 3  $k$-SCOPE algorithm

<table>
<thead>
<tr>
<th>Input:</th>
<th>$D$ (the original database), $k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>$L$ (sorted list of $k$ strongly correlated item pairs)</td>
</tr>
<tr>
<td>1.</td>
<td>Generate Correlogram Matrix from $D$</td>
</tr>
<tr>
<td>2.</td>
<td>For each item pair $(i, j) \in D$</td>
</tr>
<tr>
<td></td>
<td>a. Compute Correlation coefficient, $Corr(i, j)$ by</td>
</tr>
<tr>
<td></td>
<td>using support count from the correlogram matrix.</td>
</tr>
<tr>
<td></td>
<td>b. if $Corr(i, j) \geq Corr(L[k])$ then add item</td>
</tr>
<tr>
<td></td>
<td>pair $(i, j)$ into last position of the list $L$.</td>
</tr>
<tr>
<td></td>
<td>i. Sort the list $L$ in descending order</td>
</tr>
<tr>
<td></td>
<td>based on $Corr$ of each pair.</td>
</tr>
<tr>
<td>3.</td>
<td>Return $L$</td>
</tr>
</tbody>
</table>
5.3 Analysis of k-SCOPE algorithm

Since $k$-SCOPE is an extension of SCOPE, so here we analyse only $k$-SCOPE in terms of space and time complexity.

5.3.1 Space complexity

TAPER and TOP-COP need memory space for keeping top-$k$ list and support count of all items and also to store a huge number of candidate item pairs, depending on the $\theta$ upper bound value. TOP-COP maintains a list for keeping the pruning status of all the item pairs out of $n$ items, which requires memory space of order $(n^2)$. Tkcp creates an entire FP-tree in the memory with initial support threshold zero (0), which is normally very huge when the number of transactions as well as the dimensionality is large and also depends on the number of unique patterns of items in the database. Sometimes it is difficult to construct such tree in the memory. However, $k$-SCOPE always requires a fix memory of size, $n \times (n + 1)/2$, for $n$ dimensional market basket database to construct correlogram matrix and array of size $k$ to store top-$k$ strongly correlated item pairs. Thus, the total space requirement is:

$$\text{SPACE}_{k\text{-SCOPE}} = O(n \times (n + 1)/2) + O(k)$$

5.3.2 Time complexity

The computational cost for $k$-SCOPE consists of two parts:

1. correlogram matrix construction cost ($C_{CM}$)
2. cost for extraction of top-$k$ strongly correlated item pairs ($C_{EX}$).

5.3.2.1 Construction of correlogram matrix

Let us consider that the database contains $T$ number of transactions and maximum $N$ number of items in each transaction. So, to scan the database once it requires $(T \times N)$ times. For storing and updating support count for item pairs in correlogram matrix with respect to each transaction, it requires to find out all pair combinations of the items present in that transaction. Thus, the time requirement for updating the count is $N^2$. The total time complexity for construction of correlogram matrix is: $C_{CM} = O(T \times N^2)$.

5.3.2.2 Extraction of top-$k$ strongly correlated item pairs

Since for calculating the correlation of each pair $k$-SCOPE has to traverse the correlogram matrix once, thus the time requirement for extracting correlation coefficient of all item pairs is $O(n \times (n + 1)/2) \approx O(n^2)$. In order to create the top-$k$ list, for each item pair it compares the correlation coefficient ($Corr$) of the new pair with the $(k - 1)^{th}$ pair in the list. If $Corr$ of new pair is greater than the $k^{th}$ pair, then the $k^{th}$ pair is eliminated from the list and a new pair is inserted and placed in the list according to the descending order of the $Corr$. Thus, for placing a new pair it requires at most $k$ times comparison and swapping. Since, the problem is to find $k$ top most item pairs out of $n \times (n - 1)/2$ item pairs, thus considering worst case scenario, the time requirement for creating list of top $k$ item pairs can be denoted as:

$$C_{EX} = O(n^2) + O(k \times (n \times (n - 1))/2)$$
Thus, the total cost incurred by $k$-SCOPE is:

$$COST_{k-SCOPE} = C_{CM} + C_{EX}$$
$$= O(T \times N^2) + O(n^2) + O(k \times (n \times (n - 1))/2)$$
$$\approx O(T \times N^2) + O(n^2) + O(k \times n^2).$$

The computational cost of TOP-COP and TAPER algorithm are almost similar, except the cost of computing exact correlations for remaining candidates may be less in case of TOP-COP, as it prunes more number of non-potential item pairs based on 2-D monotone property. The cost of TOP-COP algorithm could be modelled as,

$$COST_{TOP-COP} = C_{Sort} + C_{Bound} + C_{Exact} + C_{k-list}$$

where $C_{Sort}, C_{Bound}, C_{Exact}$ and $C_{k-list}$ are cost of creating sorted list of items on non-increasing order of support, cost of computing upper bounds, computing cost of exact correlation of remaining pairs and $k$-top list maintenance cost respectively. On elaborating and simplifying the above cost computation parameters, we have

$$COST_{TOP-COP} = O(n \log n) + O(n^2) + O(n^2) + O(k^2)$$

However, the above cost model, do not consider the cost of scanning the database. It requires one scan for creating initial sorted item list and at least another whole scan (when any hash-based data structure is used) of the database for computing exact correlation of existing pairs after pruning. On incorporating such cost, it would be:

$$COST_{TOP-COP} = O(T \times N) + O(n \log n) + O(T \times N) + O(n^2)$$
$$+ O(n^2) + O(k^2)$$
$$\approx 2 \times O(T \times N) + O(n \log n) + 2 \times O(n^2) + O(k^2)$$

Similarly the cost of Tkcp algorithm could be modelled as:

$$COST_{Tkcp} = C_{Sort} + C_{DSort} + C_{FP} + C_{k-list}$$
$$= (O(T \times N) + O(n \log n)) + O(T \times N^2)(O(T \times N) + C_{FP,Tree})$$
$$+ (O(n) \times C_{Cond,base} + O(P \times k^2))$$

where $C_{Sort}$ is the cost of creating initial sorted list of items based on support count by one pass of the database, $C_{DSort}$ is the cost incurred during sorting the database based on descending order of item support, $C_{FP}$ -total cost of creating FP-tree. Creation of the complete FP-tree requires one complete scan over the database and the cost of creating pattern tree is $C_{FP,Tree}$. To compute the correlation of each pair and to maintain $k$-top list, it requires additional cost $C_{Cond,base}$ for creating conditional pattern base (P) for each item. As it can be seen that the cost of scanning a database is much larger than the other computational parameters, so the computational savings of $k$-SCOPE, i.e., ($O(T \times N)$) can be found to be larger when the number of records in a transaction database is very high.
6 Performance evaluation

To evaluate the performance of SCOPE and \( k \)-SCOPE in comparison to other techniques, we tested them in light of several synthetic as well as real-life datasets. Since, TAPER in its original form cannot generate top-\( k \) list, we modified TAPER, so that it can generate such top-\( k \) strongly correlated item pair list. As TAPER is dependent on the correlation threshold \( \theta \), in order to generate same result by TAPER we set \( \theta \) with the correlation coefficient of the \( k \)-th pair from the top-\( k \) list generated by \( k \)-SCOPE. The ideal \( \theta \) value for TAPER for different datasets are presented in Table 6. We also report results showing the performance of Spearman’s \( \rho \) as correlation coefficient, compared to Pearson’s \( \phi \) with respect to market basket data.

6.1 Environment used

We used Java for implementation of SCOPE, \( k \)-SCOPE, TAPER and the modified version of TAPER. For TOP-COP, we used the code as provided by the original author. Since performance of Tcp and Tkcp highly dependent on FP-tree implementation, we adopted third party FP-tree implementation from Borghelt (2005) for Tcp and Tkcp to avoid implementation bias.

6.2 Dataset used

We have generated several synthetic datasets according to the specifications given in Table 4. The synthetic datasets were created with the data generator in ARMiner software (http://www.cs.umb.edu/laur/ARMiner/), which also follows the basic spirit of well-known IBM synthetic data generator for association rule mining. The size of the data (i.e., number of transactions), the number of items and number of unique patterns in the transactions are the major parameters in the synthesised data generation. We also used real life mushroom dataset from UCI ML repository (http://www.ics.uci.edu/mllearn/MLRepository.html) and Pumsb (http://fimi.cs.helsinki.fi/data/) from IBM corresponds to binarised versions of a census dataset (Table 5). Pumsb is often used as the benchmark for evaluating the performance of association mining algorithms on dense datasets. Next we present our experimental results for the various real and synthetic datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>No. of transactions</th>
<th>No. of items</th>
<th>Avg. size of transaction</th>
<th>No. of patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>T10I400D100K</td>
<td>100,000</td>
<td>400</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>T10I600D100K</td>
<td>100,000</td>
<td>600</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>T10I800D100K</td>
<td>100,000</td>
<td>800</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>T10I1000D100K</td>
<td>100,000</td>
<td>1,000</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>T10P1000D100K</td>
<td>100,000</td>
<td>1,000</td>
<td>10</td>
<td>1,000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset (market basket)</th>
<th>No. of transactions</th>
<th>No. of items</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mushroom</td>
<td>8,124</td>
<td>128</td>
<td>UCI</td>
</tr>
<tr>
<td>Pumsb</td>
<td>49,046</td>
<td>2,113</td>
<td>IBM Almaden</td>
</tr>
</tbody>
</table>
Table 6  Suitable θ value for different datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mushroom</td>
<td>0.49</td>
<td>0.37</td>
<td>0.31</td>
<td>0.25</td>
<td>0.23</td>
</tr>
<tr>
<td>Pumsb</td>
<td>0.97</td>
<td>0.869</td>
<td>0.764</td>
<td>0.703</td>
<td>0.647</td>
</tr>
<tr>
<td>T10I400D100K</td>
<td>0.51</td>
<td>0.027</td>
<td>-0.006</td>
<td>-0.011</td>
<td>-0.016</td>
</tr>
<tr>
<td>T10I600D100K</td>
<td>0.81</td>
<td>0.27</td>
<td>0.001</td>
<td>-0.006</td>
<td>-0.009</td>
</tr>
<tr>
<td>T10I800D100K</td>
<td>0.63</td>
<td>0.290</td>
<td>0.001</td>
<td>-0.003</td>
<td>-0.005</td>
</tr>
<tr>
<td>T10I1000D100K</td>
<td>0.96</td>
<td>0.95</td>
<td>0.94</td>
<td>0.93</td>
<td>0.89</td>
</tr>
<tr>
<td>T10P1000D100K</td>
<td>0.95</td>
<td>0.92</td>
<td>0.87</td>
<td>0.83</td>
<td>0.80</td>
</tr>
</tbody>
</table>

Figure 4  Execution time comparison between SCOPE, Tcp and TAPER
6.3 Experimental results

To evaluate the performance of the proposed algorithms, we compared them with other similar techniques in terms of execution time for different values of $\theta$ and $k$. In case of TcP and Tkcp, it is found to consume maximum time compared to other two techniques, since TcP and Tkcp generate entire FP-tree with initial minimum support value 0. It can also be observed that TcP and Tkcp cannot perform when number of items is more than 1,000. In case of T10P1000D100K dataset, both TcP and Tkcp failed to mine, due to large number of items and unique patterns. However, in all cases, SCOPE exhibits comparatively better performance than TAPER and TcP. With the decrease in $\theta$, the running time requirements of TAPER also increases, since low $\theta$ value generates large number of candidate sets. But, SCOPE and TcP keep stable running time for the whole range of correlation threshold in different datasets. It further confirmed the fact that like TcP, SCOPE is also robust with respect to input parameters (Figure 4).

Figure 5 Execution time comparison of k-SCOPE with TAPER (mod), Tkcp and TOPCOP on synthetic dataset
From the performance graph in Figures 5 and 6, it could be easily observed that modified TAPER performs much better than TOP-COP, even though TOP-COP is an improved and modified version of TAPER. It happens because of the use of efficient hash data structure, which is lacking in original TOP-COP implementation. This further indicates that the performance of correlation mining algorithms could be improved through efficient implementation. However, in all cases, \(k\)-SCOPE exhibits comparatively better performance than TAPER (modified), TOP-COP and Tkcp. TOP-COP exhibits an exponential performance graph (Figures 5 and 6) in increasing number of items. But \(k\)-SCOPE and Tkcp maintains stable running time in different datasets, since both algorithms are independent of \(\theta\). It further confirms the fact that SCOPE and \(k\)-SCOPE is robust with respect to input parameters \(\theta\) and \(k\) respectively.

Figure 6 Execution time comparison of \(k\)-SCOPE with TAPER (mod), Tkcp and TOPCOP on real dataset

6.4 Scalability of \(k\)-SCOPE

The scalability of \(k\)-SCOPE algorithm with respect to the number of transactions and number of items in the databases is shown in Figure 7. In this experiment, we used ARMiner to generate synthetic datasets. We generated four datasets with the number of transactions ranges from 100,000 to 500,000, keeping number of items as 1,000 to test the scalability in terms of increasing number of transactions. In order to test the scalability in terms of number of items, we generated another five transaction datasets with number of items ranging from 2,000 to 10,000 keeping number of transactions equal to 100,000. It is observed that, the execution time increases linearly with the increase in the number of transactions and items at different \(k\) values. Figure 7 reports the scalability test results for \(k\) ranging from 500 and 2,500. From the graph it is clear that the performance of \(k\)-SCOPE is insensitive to input parameter \(k\). Thus \(k\)-SCOPE is robust in handling large transaction databases for different values of \(k\).

Next we report few results to establish that Spearman’s \(\rho\) is superior in comparison to Pearson’s \(\phi\) over market basket dataset in terms of

1. finding number of correlated item pairs
2. correlation coefficient values for different \(k\) values.
6.5 Pearson’s \( \phi \) vs. Spearman’s \( \rho \) in correlated item pair findings

For measuring the performance of \( \phi \) and \( \rho \) as a correlation coefficient for finding correlated item pairs, we used the same real Mushroom dataset. We have measured the number of possible correlated item pairs for various \( \theta \) values, generated by both Pearson’s \( \phi \) and Spearman’s \( \rho \). From Figure 8, it can easily be observed that Spearman’s \( \rho \) generates more number of correlated pairs compared to Pearson’s \( \phi \). Similarly, while measuring the different correlation coefficient values computed by both the methods for different \( k \) values, Spearman’s \( \rho \) gives higher value than Pearson’s \( \phi \). From the results it can be observed that Spearman’s \( \rho \) is able to detect some hidden correlated pairs undetected by Pearson’s \( \phi \). Moreover, in some cases \( \rho \) gives higher correlation value compared to \( \phi \), for a particular pair. We feel that this is due to the problems associated with Pearson’s \( \phi \) as already discussed.

An application of SCOPE in extracting genetic network in yeast from gene expression data, is discussed next.

Figure 8 Performance comparison of Pearson’s \( \phi \) and Spearman’s \( \rho \) (see online version for colours)
7 Finding gene interaction networks using support-based correlation: an application

Gene expression technology, such as high density DNA micro-array, allows us to monitor gene expression patterns at the genomic level. Advent of this technology leads to the new challenges of extracting biologically relevant knowledge from such large gene expression datasets. As a result, mining of gene expression data has become an important area of research for biologists. Development of suitable mining techniques will contribute to get into the insight of the gene-gene relationships and that may further lead to discover hidden facts related to any species or microbes. Gene-gene relationship can be described through biological pathways. Again biological pathways can be represented as networks and broadly classified (Tavazoie et al., 1999) as metabolic pathways, signal transduction pathways and gene regulatory networks or gene interaction networks. Biological network connects genes, gene products (in the form of protein complexes) to one another. A network of co-regulated genes may form gene clusters that can encode proteins, which interact amongst themselves and take part in common biological processes.

Interaction network is the highly connected gene-gene relationship network with high association. We consider correlation coefficient as a measure of association between gene pairs. Gene pair showing strong correlation coefficient is considered for constructing the genetic networks. A high correlation between two genes means that one gene is non-randomly associated with the other. In this way, correlation coefficient can be used as a metric between two genes related to their degree of independence. We hypothesise that the higher the correlation between two genes, the more likely may have a biological relationship between them. We are interested in extracting network with strong positive correlations. To calculate correlation coefficient between each pair of genes we used support-based correlation mining technique, SCOPE. To find the interaction network, strongly correlated gene pairs are extracted and adjacency matrix constructed for the network.

We used publicly available yeast sporulation data (http://cmgm.stanford.edu/pbrown/sporulation). It is a collection of 6,118 genes under seven time points (0, 0.5, 2, 5, 7, 9 and 11.5 h) measured during the sporulation process of budding yeast. The data are then log-transformed. Among the available genes in original dataset, genes whose expression levels did not change significantly during the harvesting have been ignored. This is determined with a threshold level of 1.6 for the root mean square of the log2-transformed ratios. The resulting set consists of 474 genes. We try to compare two gene expressions both in terms of angle and regulation pattern between two adjacent conditions, simultaneously. In order to capture both regulation pattern and angular deviation of each gene, we read row of original data with S number of expression values or conditions and converted into another row of \((S - 1)\) number of columns, each column of which contains angle and regulation pattern of two adjacent conditions. Here, regulation information are denoted by triplet values \([1, 0, -1]\) to represent up-regulation, no changes and down regulation respectively. For calculating angular deviation we take arc tangent between two adjacent expression levels. We say two gene expressions in a particular condition is same (denoted as 1) if the regulation is same and angular deviation difference is within specified threshold \(\tau\). Otherwise two expressions in a particular condition are considered as different (denoted by 0). Following this process we convert the original data into binary market basket dataset.
Later, we compute $\phi$ correlations of all the pairs using SCOPE and construct an adjacency matrix using strongly correlated pairs. From the adjacency matrix network is constructed and the network is visualised with the help of graph visualisation tool GUESS (http://guess.wikispot.org). Figure 9 presents two sub-networks.

**Figure 9** Constructed networks from yeast sporulation data (13 and 14 genes) (see online version for colours)

Biological interpretation of the results can be assessed by functional annotation of the genes participated in a cluster. We determined the biological relevance of the smaller groups comprise of all the genes participated in a common genetic network from yeast sporulation data, in terms of the statistically significant GO terms validated using GO annotation database (http://db.yeastgenome.org/cgi-bin/GO/goTermFinder). In this annotation database, genes are assigned to three structured, controlled vocabularies (ontologies) that describe gene products in terms of associated biological processes, components and molecular functions in a species-independent manner. The significant shared GO terms associated with set of genes (13 and 14) from the above networks are reported below.

Out of 13 genes in the first network, the genes (YJR145C, YPL198W, YML063W, YHR203C, YJR123W, YGL076C, YPL131W, YLR075w) are involved in the process of regulation of translation, while genes (YJR145C, YPL090C, YLR075w, YPL198W, YHR203C, YML063W, YJR123W, YGL076C, YPL131W, YOR096W, YOL040C) are involved in translation process. On the other hand genes (YNL069C, YOR096W, and YOL040C) are involved in structural molecule activity. Similarly, out of 14 genes from the second network, the most significant processes are cell differentiation (YDR273W, YBR148W, YGL170C, YIL045W, YHR185C, YLR307W, YJL038C, YHR184W) and spore wall biogenesis (YDR273W, YHR185C, YLR307W, YHR184W), however, no significant GO terms found in case of molecular function. A network of 16 genes is extracted with the above parameters. Out of which, majority of the genes (YMR242C, YPL081W, YLL045c, YNL301C, YIL052C, YER126c, YNL098C, YDR418W, YER074w, YNL119W, YER056c-a, YJL177W, YPL079W, YHR010W) are involved in gene expression and cellular protein metabolic process (YMR242C, YPL081W, YLL045c, YNL301C, YIL052C,
YDR418W, YER074w, YNL119W, YER056c-a, YJL177W, YPL079W, YHR010W). In case of cellular components, genes from first network (YJR145C, YPL090C, YLR075w, YPL198W, YHR203C, YML063W, YJR123W, YGL076C, YPR131W, YOR096W, YOL040C) belongs to ribosomal subunit and genes (YJR145C, YML063W, YHR203C, YJR123W, YPL090C, YOR096W, YOL040C) are involved in cytosolic small ribosomal subunit. Similarly, from second network, the significant components are prospore membrane (genes YDR273W, YBR148W, YHR184W). Genes (YMR242C, YPL081W, YLL045c, YNL301C, YIL052C, YER126c, YDR418W, YER074w, YER056c-a, YJL177W, YPL079W, YHR010W) from third network are belonging to intracellular non-membrane-bounded organelle and ribonucleo protein complex.

8 Conclusions

Two effective techniques for finding strongly correlated item pairs and top-$k$ strongly correlated item pairs from market basket data are reported in this paper. An alternative way of measuring correlation coefficient between item pair from market basket data using Spearman’s rank order correlation is also reported. The major advantages of these techniques while compared to existing similar techniques are

a requires single pass over the whole database
b require no candidate generation
c supports interactive mining.

Both the techniques were evaluated in terms of several synthetic and real life datasets and results are found quite satisfactory. This paper also reports a potential application of the proposed technique (SCOPE) in constructing genetic network from yeast.

Acknowledgements

The authors would like to thank all three anonymous reviewers and the editor of this journal for their constructive suggestions.

References

96  S. Roy and D.K. Bhattacharyya


