Clustering Items in Different Data Sources Induced by Stability

Jhimli Adhikari[1], Pralhad Rao[2], and Animesh Adhikari[3]
[1]Department of Computer Science, Narayan Zantye College, India
[2]Department of Computer Scence and Tecnology, India
[3]Department of Computer Science, Chowgule College, India

Abstract: *Many multi-branch companies transact from different branches. Each branch of such a company maintains a separate database over time. The variation of sales of an item over time is an important issue. Thus, we introduce the notion of stability of an item. Stable items are useful in making many strategic decisions for a company. Based on the degree of stability of an item, we design an algorithm for clustering items in different data sources. We have proposed the notion of best cluster by considering average degree of variation of a class. Also, we have designed an alternative algorithm to find best cluster among items in different data sources. Experimental results are provided on three transactional databases.*

Keywords: *Clustering, data mining, dispersion, multiple databases, stability*.

Received March 12, 2007; accepted May 20, 2008

1. Introduction

Due to the liberalization of government policies across the globe, the number of multi-branch companies is increasing over time. Many multi-branch companies deal with multiple databases. Thus, the study of data mining on multiple databases is an important issue. Data mining and knowledge discovery on multiple databases has been recently recognized [1], [17] as an important area of research in data mining community.

Many of these multi-branch companies also deal with transactional time-stamped data. Transactional data collected over time at no particular frequency is called transactional time-stamped data [11]. Some examples of transactional time-stamped data are point of sales data, inventory data, and trading data. Little work has been reported on the area of mining multiple transactional time-stamped databases. Many important and useful applications might involve transactional time-stamped data.

All the transactions in a branch might get stored locally. A transaction could be viewed as a collection of items with a unique identifier. An interesting characteristic of an item is its variation of sales over time. The items having less variation of sales over time are useful in devising strategies for a company. Thus it is important to study such items. In the following example, we consider a few sample time series of supports corresponding to different items.

Example 1: let i^{th} series be the time series of supports corresponding to item x_i, for i = 1, 2, 3, 4, 5.

(1) .03, .20, .31, .11, .07, .35, .82, .62, .44, .13
(2) .19, .20, .18, .21, .20, .20, .19, .18, .21, .20
(3) .05, .11, .07, .20, .16, .12, .13, .08, .17, .10
(4) .03, .04, .03, .07, .08, .12, .09, .15, .17, .12
(5) .04, .04, .03, .05, .04, .06, .04, .05, .06, .05

Among the support series corresponding to different items, we observe that the variation of sales corresponding to item x_5 is the least. Thus, the company would prefer to devise strategy based on item x_5.

Rest of the paper is organized as follows. We discuss related work in section 2. In section 3, we propose a model of mining multiple transactional time-stamped databases. We state our problem in section 4. In section 5, we design an algorithm for clustering of items in multiple databases. Experimental results are provided in section 6.

2. Related Work

Liu *et al.* [12] have proposed stable association rules based on testing of hypothesis. In this case, the distribution of test statistic under null hypothesis is normal for large sample size. Thus, the stable association rules are determined based on some assumptions. Due to these reasons, we define stable items based on the concept of stationary time series data [6].

In the context of interestingness measures, Tan *et al.* [14] have described several key properties of twenty one interestingness measures proposed in statistics, machine learning and data mining literature. Wu *et al*. [16] have proposed two similarity measures for clustering a set of databases.

Zhang *et al.* [19] have proposed an efficient and scalable data clustering method BIRCH based on a

new in-memory data structure called CF-tree. Estivill-Castro and Yang [7] have proposed an algorithm that remains efficient, generally applicable, multi-dimensional but is more robust to noise and outliers. Jain *et al.* [10] have presented an overview of pattern clustering methods from a statistical pattern recognition perspective, with a goal of providing useful advice and references to fundamental concepts accessible to the broad community of clustering practitioners. In this paper, we cluster items in multiple databases based on supports of items. Thus, the above algorithms might not be suitable under this framework.

Yang and Shahabi [18] have proposed an algorithm to determine the stationarity of multivariate time series data for improving the efficiency of many correlation based data analysis.

Zhang *et al.* [20] designed a local pattern analysis for mining multiple databases. Zhang *et al.* [21] studied various issues related to multi-database mining.

3. A Model Of Multiple Transactional Time Stamped Databases

Consider a multi-branch company that has n branches. Let D_i be the transactional time-stamped database corresponding to i^{th} branch, for i = 1, 2, …, n. Web sites and transactional databases contain a large amount of time-stamped data related to an organization's suppliers and / or customers over time. Mining these types of time-stamped data could help business leaders make better decisions by listening to their suppliers or customers via their transactions collected over time [11]. We propose a model of mining global patterns in multi-databases over time.

Adhikari and Rao [2] have proposed an extended model of mining multiple databases using local pattern analysis. The limitation of this model is that it provides approximate global pattern. Thus, we propose a new model of mining global patterns in multiple transactional time-stamped databases. The proposed model in Figure 1 has a set of interfaces and a set of layers. Each interface is a set of operations that produces dataset(s) (or, knowledge) based on the lower layer dataset(s). There are five distinct interfaces of the proposed model of synthesizing global patterns from local patterns. The function of each interface is described below. Interface 2/1 cleans / transforms / integrates / reduces data at the lowest layer. By applying these procedures we get processed database from the original database. In addition, interface 2/1 applies a filtering algorithm on each database for separating relevant data from outlier data. E.g., if we are interested in studying the durable items then the transactions containing only non-durable items could be treated as outlier transactions. Also, it loads data into the respective data warehouse. At interface 3/2, each processed database PD_i is partitioned into k time databases DT_{ij}, where DT_{ij} is the processed database (if available) for the j^{th} time slot at the i^{th} branch, for j = 1, 2, …, k, and i = 1, 2, …, n. The j^{th} time databases of all branches are merged into a single time database DT_j, for j = 1, 2, …, k. A traditional data mining technique could be applied on database DT_j at the interface 5/4, for j = 1, 2, …, k. Let PB_j be pattern base corresponding to the time database DT_j, for j = 1, 2, …, k. Finally, all the pattern bases are processed for synthesizing knowledge or, making decision at the interface 6/5. Other undirected lines in Figure 1 are assumed to be directed from bottom to top. The proposed model of mining global patterns over time is efficient, since we get the exact global patterns in multiple databases over time.

In layer 4, we have collection of all time databases. If any one of these databases is too large to apply a traditional data mining technique then this data mining model would fail. In this situation, we could apply an appropriate sampling technique to reduce the size of a database. Thus, we get approximate patterns over time.

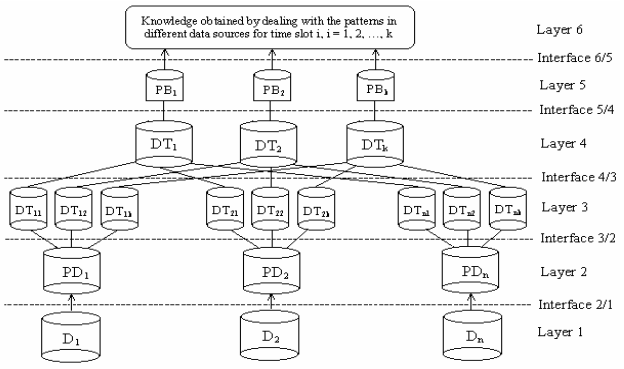


Figure 1. A model of mining global patterns in multiple transactional time-stamped databases.

4. Problem Statement

With reference to Figure 1, let DT_j be the database corresponding to the j^{th} year, for j = 1, 2, …, k. Each of these databases corresponds to a specific period of time. Thus, we could call them as time databases. Each of these time databases is mined using a traditional data mining technique [9], [4]. For the specific requirement of this problem, we need to mine only items in the time databases. Let I be the set of all items in these databases. Each itemset X in a database D is associated with a statistical measure, called support [3], denoted by $supp(X, D)$. The support of an itemset is defined as the fraction of transactions containing the itemset. The variation of sales of an item over the time is an important issue in determining stability of the item. Stable items are useful in many applications, e.g., stable items could be useful to promote sales of other items. Modeling with stable item is more justified than modeling with unstable item.

Let $\mu_{s(x)}(t)$ be the mean support of item x in the database $DT_1, DT_2, \ldots, DT_t$. Thus, $\mu_{s(x)}(t)$ is obtained by the following formula:

$$\mu_{s(x)}(t) = \left(\sum_{i=1}^{t} supp(x, DT_i) \times size(DT_i)\right) / \sum_{i=1}^{t} size(DT_i), \quad t = 1, 2, \ldots, k \tag{1}$$

Let $\sigma(\mu_{s(x)})$ be the standard deviation of $\mu_{s(x)}(t)$, for t = 1, 2, …, k. We call $\sigma(\mu_{s(x)})$ as the *variFexapation of means* corresponding to support of x. Let $\gamma_{s(x)}(t, t + h)$ be the autocovariance of $supp(x, DT_t)$ at lag h, for t = 1, 2, …, k-1. Thus, $\gamma_{s(x)}(t, t + h)$ is obtained by the following formula:

$$\gamma_{s(x)}(t,t+h) = \frac{1}{k}\sum_{t=1}^{k-h} \left(supp(x, DT_t) - \mu_{s(x)}(k)\right) \left(supp(x, DT_{t+h}) - \mu_{s(x)}(k)\right) \tag{2}$$

$\sigma(\gamma_{s(x)}(t, t + h))$ be the standard deviation of $\gamma_{s(x)}(t, t + h)$, for h = 1, 2, … , k-1. We call this $\sigma(\gamma_{s(x)}(t, t + h))$ as *variation of autocovariances* corresponding to support of x. We have chosen standard deviation as a measure of dispersion [5]. Standard deviation and mean deviation about mean are relevant measures of dispersion. These measures take into account of variation due to each support unlike the measure range. Skewness, being a descriptive measure of dispersion is not suitable in this context. Before we define stability of an item, we study the following time series of supports corresponding to an item. In the following example, we compute $\sigma(\mu)$ and $\sigma(\gamma)$ of support series corresponding to different items.

Example 2: we continue with Example 1. The variations of means and autocovariances of above series are given as follows: (1) $\sigma(\mu) = 0.09342$, $\sigma(\gamma) = 0.01234$, (2) $\sigma(\mu) = 0.00230$, $\sigma(\gamma) = 0.00002$, (3) $\sigma(\mu) = 0.02351$, $\sigma(\gamma) = 0.00039$, (4) $\sigma(\mu) = 0.02114$, $\sigma(\gamma) = 0.00076$, (5) $\sigma(\mu) = 0.0027986$, $\sigma(\gamma) = 0.0000124$. We observe that the value of total variation, $\sigma(\mu) + \sigma(\gamma)$, is the least corresponding to item x_5.

We define stable items based on the concept of stationary time series data [6]. In finding $\sigma(\mu)$, we first compute a set of means of support values. Then we compute standard deviation of these mean values. Thus, we find standard deviation of a set of fractions. In finding $\sigma(\gamma)$, we first compute a set of autocovariances of support values. Then we compute standard deviation of these autocovariances. An autocovariance of supports is an average of a set of squared fractions. Thus, we find standard deviation of a set of squared fractions. So, $\sigma(\mu) \geq \sigma(\gamma)$. In fact, $\sigma(\gamma)$ is close to 0. Thus, we define our first measure of stability $stable_1$ as follows.

Definition 1: an item x is stable if $\sigma(\mu_{s(x)}) \leq \delta$, where δ is user defined maximum threshold.

More strictly, we may wish to impose restrictions on both $\sigma(\mu)$ and $\sigma(\gamma)$. Thus we define our second measure of stability $stable_2$ as follows.

Definition 2: an item x is stable if $\sigma(\mu_{s(x)}) + \sigma(\gamma_{s(x)}) \leq \delta$, where δ is user defined maximum threshold.

In Definition 2, the expression $\sigma(\mu_{s(x)}) + \sigma(\gamma_{s(x)})$ is the determining factor of stability of an item. We define degree of variation of an item x as follows.

$$degOfVar(x) = \sigma(\mu_{s(x)}) + \sigma(\gamma_{s(x)}) \tag{3}$$

Higher value of *degOfVar* implies lower degree of stability of the item. Based on above discussion, we state our problem as follows.

Let D_i and DT_j be the databases corresponding to i^{th} branch and j^{th} year of a multi-branch company as depicted in Figure 1, respectively for $i = 1, 2, \ldots, n$, and $j = 1, 2, \ldots, k$. Each of the time (year) databases has been mined using a traditional data mining technique. Based on the mining results, degree of variation of each item has been computed as discussed above. Find the best non-trivial partition (if it exists) of the items in $D_1, D_2, \ldots, D_n$ based on degree of variation of an item.

A partition [13] is a specific type of clustering. Formal definition of non-trivial partition is given in section 4.

5. Clustering Items

The proposed clustering technique is based on the notion of degree of stability of an item. Again, the degree of stability is based on the variations of means and autocovariances. The clustering technique requires computing the degree of variation for each item in the databases. Let I be the set of all items in the databases. Given a set of yearly databases, the difference in variations between every pair of items could be expressed by a square matrix, called *difference in variation* (*diffInVar*). We construct *diffInVar* as follows.

$$diffInVar(i, j) = |\, degOfVar(x_i) - degOfVar(x_j)\,|, \quad x_i, x_j \in I. \tag{4}$$

In the following example, we compute *diffInVar* corresponding to Example 1.

Example 3: we continue here with Example 1. Matrix *diffInVar* is given as follows.

$$diffInVar = \begin{bmatrix} 0 & 0.103 & 0.082 & 0.084 & 0.102 \\ 0.103 & 0 & 0.021 & 0.019 & 0.001 \\ 0.082 & 0.021 & 0 & 0.002 & 0.020 \\ 0.084 & 0.019 & 0.002 & 0 & 0.018 \\ 0.102 & 0.001 & 0.020 & 0.018 & 0 \end{bmatrix}$$

Matrix *diffInVar* is symmetric square matrix. We shall use this matrix for clustering items in multiple databases.

Intuitively, if the difference in variations between two items is close to zero then they may be put in the

same class. Before clustering the items, we define a class as follows.

Definition 3: let $I = \{i_1, i_2, \ldots, i_p\}$ be the set of items. A class formed at the level of difference in variation α is defined as follows.

$$class(I,\alpha) = \begin{cases} X: X \subseteq I, |X| \geq 2, \text{and } degOfVar(x_1, x_2) \leq \alpha, \text{for } x_1, x_2 \in X \\ X: X \subseteq I, |X| = 1 \end{cases}$$

Based on the above definition of a class, we define a clustering as follows.

Definition 4: let $I = \{i_1, i_2, \ldots, i_p\}$ be the set of items. Let $\pi(I, \alpha)$ be a clustering of items in I at the level of difference in variation α. Then, $\pi(I, \alpha) = \{X: X \in \rho(I)$, and X is a *class*$(I, \alpha)\}$, where $\rho(I)$ is the power set of I. During the clustering process we may like to impose the restriction that each item belongs to at least one class. This restriction makes a clustering complete. We define a complete clustering as follows.

Definition 5: let $I = \{i_1, i_2, \ldots, i_p\}$ be the set of items. Let $\pi(I, \alpha) = \{C_1(I, \alpha), C_2(I, \alpha), \ldots, C_m(I, \alpha)\}$, where $C_k(I, \alpha)$ is the k-th class of the cluster π, for $k = 1, 2, \ldots, m$. π is complete, if $\bigcup_{k=1}^{m} C_k(I, \alpha) = I$.

In a complete clustering, two classes may have common items. We may be interested in finding out a cluster containing mutually exclusive classes. A mutually exclusive cluster could be defined as follows.

Definition 6: let $I = \{i_1, i_2, \ldots, i_p\}$ be the set of items. Let $\pi(I, \alpha) = \{C_1(I, \alpha), C_2(I, \alpha), \ldots, C_m(I, \alpha)\}$, where $C_k(I, \alpha)$ is the k-th class of the cluster π, for $k = 1, 2, \ldots, m$. π is mutually exclusive if $C_i(I, \alpha) \cap C_j(I, \alpha) = \phi$, $i \neq j$, and $1 \leq i, j \leq m$.

We may be interested in finding out such a mutually exclusive and complete cluster. A partition of a set of items I is defined as follows.

Definition 7: let $\pi(I, \alpha)$ be a mutually exclusive and complete cluster of a set of items I at the level of difference in variation α. $\pi(I, \alpha)$ is called a non-trivial partition if $1 < |\pi| < m$.

A partition is a cluster. But a cluster is not necessarily be a partition. In the next section, we find the best non-trivial partition (if it exists) of a set of items. The items in a class are similar with respect to their variations. We are interested in the classes of a partition where the variations of items are less. The items in these classes are useful in devising strategies for the company. Thus, we define average degree of variation *adv*, of a class as follows.

Definition 8: let C be a class of partition π. Then,

$$adv(C \mid \pi) = \frac{1}{|C|} \Sigma_{x \in C} degOfVar(x).$$

5.1. Finding the Best Non-Trivial Partition

With reference to Example 1, we arrange all non-zero and distinct values of *diffInVar* in non-decreasing order for finding all the non-trivial partitions, for $1 \leq i < j \leq 5$. The arranged values of *diffInVar* are given as follows: 0.001, 0.002, 0.018, 0.019, 0.020, 0.021, 0.082, 0.084, 0.102, 0.103. We get two non-trivial partitions at α = 0.001, and 0.002. The partitions are given as follows: $\pi^{0.001} = \{\{x_1\}, \{x_2, x_5\}, \{x_3\}, \{x_4\}\}$, and $\pi^{0.002} = \{\{x_1\}, \{x_2, x_5\}, \{x_3, x_4\}\}$. We observe that at different levels of α we have different partitions. We would like to find the best partition among these partitions. The best partition is based on the principle of minimizing the intra-class variation and minimizing the inter-class similarity. Intra-class variation and inter-class similarity are defined as follows.

Definition 9: the intra-class variation *intra-var* of a partition π at the level α is defined as follows.

$$intra\text{-}var(\pi^{\alpha}) = \Sigma_{k=1}^{|\pi|} \Sigma_{x_i, x_j \in C_k; x_i < x_j} |degOfVar(x_i) - degOfVar(x_j)|$$

Definition 10: the inter-class similarity *inter-sim* of a partition π at the level α is defined as follows.

$$inter\text{-}sim(\pi^{\alpha}) = \Sigma_{c_p, c_q \in \pi; p<q} \Sigma_{x_i \in C_p, x_j \in C_q} \text{minimum}\{degOfVar(x_i), degOfVar(x_j)\}$$

The best partition among a set of partitions is selected on the basis of goodness value of a partition. Goodness measure *goodness*, of a partition is defined as follows.

Definition 11: the goodness of a partition π at level α is defined as follows: $goodness(\pi^{\alpha}) = intra\text{-}var(\pi^{\alpha}) + inter\text{-}sim(\pi^{\alpha}) - |\pi^{\alpha}|$, where $|\pi^{\alpha}|$ is the number of classes of π.

We have subtracted $|\pi^{\alpha}|$ from the sum of intra-class variation and inter-class similarity to remove the bias of goodness value of a partition. Better partition is obtained at higher goodness value. We would like to partition the set of items in Example 1 using above goodness measure.

Example 4: with reference to Example 2, we calculate goodness value of each of the non-trivial partitions. intra-var$(\pi^{0.001})$ = 0.001, inter-sim$(\pi^{0.001})$ = 0.081, and $|\pi^{0.001}| = 4$. Thus, goodness$(\pi^{0.001})$ = -3.916. intra-var$(\pi^{0.002})$ = 0.003, inter-sim$(\pi^{0.002})$ = 0.06, and $|\pi^{0.002}| = 3$. Thus, goodness$(\pi^{0.002})$ = -2.937.

The goodness value corresponding to the partition $\pi^{0.002}$ is the maximum. Thus, the partition $\pi^{0.002}$ is the best among the non-trivial partitions. Let us return back to Example 1. There are five series of supports corresponding to five items. Based on variation among the supports in a series, we could partition the series as follows: {series 1}, {series 2, series 5}, {series 3, series 4}. Hence, we get the following partition: $\{x_1\}$, $\{x_2, x_5\}$, $\{x_3, x_4\}$. The proposed clustering technique also identifies the same partition as the best partition. Thus, it verifies the correctness of the proposed clustering technique. *adv* $(\{x_1\} \mid \pi^{0.002}) = 0.105$, *adv* $(\{x_2, x_5\} \mid \pi^{0.002}) = 0.0025$ and *adv* $(\{x_3, x_4\} \mid \pi^{0.002}) = 0.022$. We find that the average degree of variation of $\{x_2, x_5\}$ is the least among the classes of $\pi^{0.002}$. Thus, the items x_2 and x_5 are most suitable among all the items in the given databases for making strategies of the company. We design an algorithm for finding best

non-trivial partition of items in multiple databases. First we describe different data structures used in designing an algorithm for finding the best partition of items. For each item there are k supports corresponding to k different years.

We maintain $m \times k$ supports for m items in array supports. The i^{th} row of supports stores supports corresponding to i^{th} item for k years, for i = 1, 2,…, m. Let means be a two dimensional array such that the i-th row stores means of supports corresponding to different years for i^{th} item, for i = 1, 2,…, m. Let autocovariances be a two dimensional array such that the i^{th} row stores autocovariances of supports corresponding to different lags for i^{th} item, for i = 1, 2,…, m. For year j, we compute mean value of supports for year 1 to j. Thus we get different mean values for different years. Let *stdDevMeans* be the standard deviation of these mean values. For year j, we also compute autocovariances of supports for year 1 to j at different lags. Thus we get different autocovariances for different lags corresponding to a year. Let *stdDevAutocovars* be the standard deviation of these autocovariances. The degrees of variation of different items are stored in array degInVar. Variable S is a one dimensional array containing ${}^{m}C_2$ difference in variations. adv is a one dimensional array which stores the average degree of variation for the items in each class. The algorithm is presented below.

Algorithm 1: find best non-trivial partition (if it exists) of items in multiple databases.

Procedure BestPartition (m, supports)

Inputs:
m: number of items
supports: array of supports of different items corresponding to different years

Outputs:
Best non-trivial partition (if it exists) of items in multiple databases

```
01: for i = 1 to m do
02:   compute means(i) using formula (1) at different
      years;
03:   let stdDevMeans = standard deviation of mean
      values for different years;
04:   compute autocovariance(i) using formula (2) at
      different time lags;
05:   let stdDevAutocovars = standard deviation of
      autocovariances;
06:   compute degOfVar(i) = stdDevMeans +
      stdDevAutocovar;
07: end for
08: for row = 1 to m do
09:   for col = (row + 1) to m do
10:     compute diffInVar(row, col) using formula (4);
11:   end for
12: end for
13: sort distinct elements in the upper triangle of
    diffInVar in non-decreasing order into S;
14: let k = 1; let maxGoodness = -9999; π = ϕ;
15: while (k ≤ |S|) do
16:   let curRow = 1; let curClass = 1;
17:   for i = 2 to m do
18:     classLabel(i) = 0;
19:   end for
20:   let classLabel(1) = 1;
21:   let curDiffVar = S(k);
22:   for col = curRow + 1 to m do
23:     if (diffInVar(curRow, col) ≤ curDiffVar) then
24:       if (classLabel (col) = 0) then
25:         classLabel(col) = curClass;
26:       else if (classLabel (col) ≠ curClass) then
27:         partition does not exist at this level;
28:         go to line 49;
29:       end if
30:     end if
31:   end for
32:   increased curRow by 1;
33:   if (classLabel(curRow) = 0) then
34:     increased curClass by 1;
35:     classLabel(curRow) = curClass;
36:   else curclass = classLabel(curRow);
37:   end if
38:   if (curRow ≤ m) go to line 22; end if
39:   let j = 0;
40:   while ((classLabel(j) ≠ 0) and (j < m)) do
41:     increase j by 1;
42:   end while
43:   if (j = m + 1) then
44:     if (maxGoodness < goodness value of current
        partition) then
45:       maxGoodness = goodness value of current
          partition;
46:       store current partition into π;
47:     end if
48:   end if
49:   increase k by1;
50: end while
51: return π;
end procedure
```

In this paragraph, we explain different lines of above algorithm. Algorithm 1 computes degreeOfVar for all items using lines 1-7. Matrix *diffInVar* is constructed using lines 8-12. We check the existence of partition at every value in S. We start checking partition by assigning the first item to classLabel 1. Also, clustering process is performed row by row, starting from row number 1. At the i^{th} row, all the items greater than i are classified. During this process, if a labeled item gets another label then we conclude that partition does not exist at the current level. After increasing the current row by 1 we check the class label corresponding to current row. Each row corresponds to an item in the database. If the current row is not labeled yet then we increase the class label by 1. If the goodness value of the current partition is less than the goodness value of another partition then the current partition is ignored.

Lemma 1: algorithm 1 executes in $O(m^4)$ time.

Proof: line 2 takes $O(k)$ time to compute means(i), for some i = 1, 2, …, m. Also, line 3 takes $O(k)$ time to compute standard deviation of mean values. To

compute formula 2, we require $O(k)$ time. Thus, line 4 takes $O(k^2)$ time. In line 5, we compute standard deviation of k-1 autocovariance values. Thus, line 5 takes $O(k^2)$ time. The for-loop in lines 1-7 repeat m times. Thus, the for-loop in lines 1-7 take $O(m \times k^2)$ time. For computing *diffInVar* at a given row and column, it takes O(1) time. Thus, lines 8-12 take $O(m^2)$ time. There are maximum ${}^{m-1}C_2$ elements in the upper triangle of *diffInVar*. Thus, line 13 takes $O(m^2 \times \log(m))$ time. The while-loop at line 15 repeats maximum ${}^{m-1}C_2$ times. Each of the loops at lines 17, 22, and 40 takes O(m) time. To store a partition it takes O(m) time. To compute goodness value for a particular partition, it takes $O(m^2)$ time. Thus, the lines 15-50 take $O(m^4)$ time. The time complexity of bestPartition algorithm is $O(m^4)$.

In finding stable items in multiple databases, a class having minimal average degree of variation in the best partition might not be a best class at a given degree of stability. In many applications, we may need to find stable items at a given degree of stability. In this case, it might not be a requirement that the stable items need to form a class of a non trivial partition. Thus, the question of finding a partition might not arise always. To find such a class we shall follow a different approach.

5.2. Finding a Best Class

Before finding a best class, we first define the concept of best class as follows.

Definition 12: let C be a class of items. C is called a best class at the level of difference in variation α if (i) $|x - y| \leq \alpha$, for x, y $\in$ C, (ii) *adv*(C) is the minimum among all classes of maximal size, and (iii) C has a maximal size.

In Lemma 1, we show that it might not be possible to find two classes of maximal size having the same average degree of variation.

Lemma 2: best class is unique.

Proof: let $x_1, x_2, \ldots, x_m$ be the items sorted on non-decreasing degree of variation. We conclude that item x_1 has maximum stability, and the item x_m has minimum stability. At level α, let the stabilities of items $x_1, x_2, \ldots, x_k$ be less than or equal to α, and the stabilities of items $x_{k+1}, x_{k+2}, \ldots, x_m$ be greater than α, for $1 \leq k \leq m$. The best class has least average degree of variation. Also, the difference in variation of two items in the class is less than or equal to α. Thus, $\{x_1, x_2, \ldots, x_k\}$ forms the best class. We are not concerned whether it becomes a member of a partition. *adv*($\{x_1, x_2, \ldots, x_k\}$) is the minimum, and hence best class is unique.

We might be interested in finding best class of items in multiple databases. We use array *class* to hold the best class of items. In the following, we provide an algorithm in finding best class of items in multiple databases.

Algorithm 2: find the best class of items in multiple databases induced by stability.

Procedure BestClass (m, α, supports)

Inputs:
m: number of items
α: level of degree of variation
supports: array of supports of different items corresponding to different years

Outputs:
Best class of items in multiple databases
01: perform lines 01 – 07 of Algorithm 1;
02: sort array degOfVar in non-decreasing order;
03: let class (1) = degOfVar(1); let count = 1;
let avgVar = 0;
04: for i = 2 to m do
05: class (i) = -1;
06: end for
07: for i = 2 to m do
08: if ((degOfVar(i) – degOfVar(1)) ≤ α)
09: class (i) = degOfVar(i);
10: increase count by 1;
11: avgVar = avgVar + degOfVar(i);
12: end for
13: avgVar = avgVar / count;
14: return (class, count, avgVar);
end procedure

In this paragraph, we explain different lines of above algorithm. We compute degree of variations for all items using lines 1-7 and store them in array degreeOfVar in non-decreasing order. The best class would contain the first item of degreeOfVar. The item with least degreeOfVar is assigned to class 1. An item *i* is included in the best class if (degOfVar(i) – degOfVar(1)) $\leq \alpha$. Algorithm 2 returns best class *class*, the number of items in the best class *count*, and the average degree of variation of the best class *avgVar*.

Lemma 3. Algorithm 2 executes in maximum $\{ O(m \times k^2), O(m \times \log(m)) \}$time.

Proof. Line 1 executes in $O(m \times k^2)$ time [Lemma 2], where k is the number of years. There are two for loops in algorithm 2 apart from loops placed in line 1. Each of these loops executes in O(m) time. Line 2 takes $O(m \times \log(m))$ time. Thus, the lemma follows.

6. Experiments

We have carried out several experiments to study the effectiveness of our approach. All the experiments have been implemented on a 1.6 GHz Pentium IV with 256 MB of memory, using the software visual C++ (version 6.0). We present the experimental results using two real datasets mushroom [8], and ecoli [15]. Dataset ecoli is a subset of ecoli database and it has been processed for the purpose of conducting experiments.

Table 1. Dataset characteristics.

Dataset	NT	ALT	AFI	NI
Mushroom	8124	24.00000	1624.80000	120
Ecoli	336	7.00000	25.83517	91
Random-68	3000	5.46033	280.98529	68

Let DB, NT, ALT, AFI, and NI denote database, the number of transactions, average length of a transaction, average frequency of an item, and number of items respectively. We present some characteristics of these datasets in Table 1. Each dataset has been divided into 10 databases, called input databases, for the purpose of conducting experiments. The input databases obtained from mushroom and ecoli are named as M_i, and E_i, for i = 0, 1, …, 9. We present some characteristics of the input databases in Table 2.

In Table 3, we present top stable items in multiple databases. In Table 4, we present five best classes and their average degree of variation for a given value of α for each database.

The best partition of items in mushroom dataset is obtained at level 0.24760. The amount of intra variation, inter similarity, and goodness value are 413.59719, 377.58308, and 789.18027 respectively. The best partition contains two classes. The best class is given as follows: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119 }.

It has average degree of variation 0.05682. The best partition of items in ecoli dataset is obtained at level 0.05632. The amount of intra variation, inter similarity, and goodness value are 44.17961, 185.60862, and 227.78823 respectively. The best partition contains two classes. The best class is given as follows: {0, 1, 3, 4, 5, 6, 7, 8, 10, 11, 12, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 43, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 94, 99, 100}. It has average degree of variation 0.01829.

The best partition of items in random-68 dataset is obtained at level 0.013670. The amount of intra variation, inter similarity, and goodness value are 4.627779, 18.608446, and 21.236225 respectively. The best partition contains two classes. The best class is given as follows: {1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68}. It has average degree of variation 0.006382.

We have studied execution time with respect to number of data sources. We observe in Figures 2, 3 and 4 that the execution time increases as the number of data sources increases.

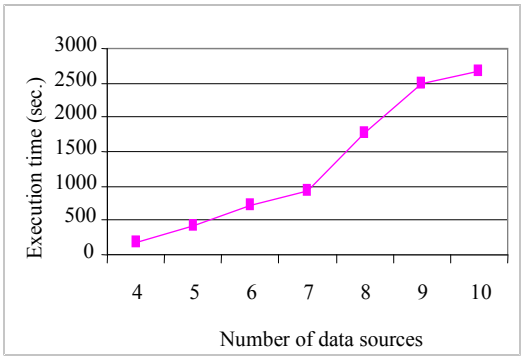


Figure 2. Execution time vs. number of data sources obtained from mushroom.

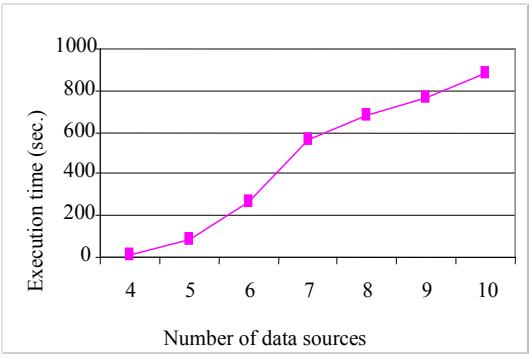


Figure 3. Execution time vs. number of data sources obtained from ecoli.

Table 2. Time database characteristics.

DB	NT	ALT	AFI	NI	DB	NT	ALT	AFI	NI
M_0	812	24.00000	295.27272	66	M_5	812	24.00000	221.45454	88
M_1	812	24.00000	286.58823	68	M_6	812	24.00000	216.53333	90
M_2	812	24.00000	249.84615	78	M_7	812	24.00000	191.05882	102
M_3	812	24.00000	282.43478	69	M_8	812	24.00000	229.27058	85
M_4	812	24.00000	259.84000	75	M_9	816	24.00000	227.72093	86
E_0	33	7.00000	4.62000	50	E_5	33	7.00000	3.91525	59
E_1	33	7.00000	5.13333	45	E_6	33	7.00000	3.50000	66
E_2	33	7.00000	5.50000	42	E_7	33	7.00000	3.91525	59
E_3	33	7.00000	4.81250	48	E_8	33	7.00000	3.39706	68
E_4	33	7.00000	3.39706	68	E_9	39	7.00000	4.55000	60
R_0	300	5.59000	28.67647	68	R_5	300	5.14000	26.67647	68
R_1	300	5.41667	28.00000	68	R_6	300	5.51000	28.35294	68
R_2	300	5.36000	27.64706	68	R_7	300	5.49667	28.33823	68
R_3	300	5.54333	28.45588	68	R_8	300	5.53667	28.47059	68
R_4	300	5.53333	28.38235	68	R_9	300	5.47667	28.23530	68

Table 3. Top 10 stable item in multiple database.

Mushroom		Ecoli		Random-68	
Item	**degOfVar**	**Item**	**degOfVar**	**Item**	**degOfVar**
85	0.00000	1	0.00129	42	0.00121
8	0.00018	99	0.00181	41	0.00182
12	0.00027	91	0.00225	37	0.00231
75	0.00030	4	0.00252	67	0.00272
89	0.00030	94	0.00363	11	0.00281
62	0.00090	15	0.00385	45	0.00287
22	0.00104	12	0.00389	18	0.00301
20	0.00107	19	0.00402	56	0.00310
82	0.00124	3	0.00404	3	0.00310
33	0.00141	10	0.00443	28	0.00312

Table 4. Five best classes in multiple databases.

Mushroom			Ecoli			Random-68		
α	**Items**	**adv**	**α**	**Items**	**adv**	**α**	**Items**	**adv**
0.00025	{85, 8}	0.00009	0.0020	{1, 99, 91, 4}	0.00196	0.0010	{42,41}	0.00152
0.0003	{85, 8, 12, 75, 89}	0.00021	0.0025	{1, 99, 91, 4, 94}	0.00230	0.0015	{42,41,37}	0.00178
0.0010	{85, 8, 12, 75, 89, 62}	0.00033	0.0030	{1, 99, 91, 4, 94, 15,12,19, 3}	0.00303	0.0017	{42,41,37,67,11,45}	0.00229
0.0012	{85, 8, 12, 75, 89, 62, 22, 20}	0.00051	0.0035	{1, 99, 91, 4, 94, 15,12,19, 3,10,6}	0.00331	0.0020	{42,41,37,67,11,45,18,56,3, 28}	0.00261
0.0014	{85, 8, 12, 75, 89, 62, 22,20, 82}	0.00059	0.0050	{1, 99, 91, 4, 94, 15,12,19, 3,10,6,18}	0.00354	0.0022	{42,41,37,67,11,45,18,56,3,28,7, 53}	0.00271

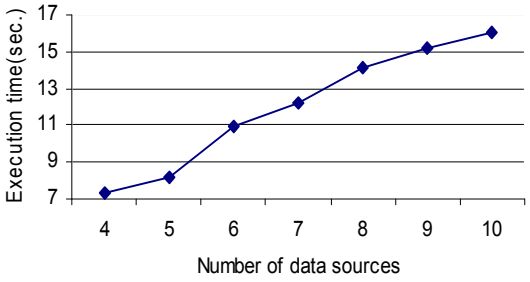


Figure 4. Execution time vs. number of data sources obtained from random-68 data.

7. Conclusion

Stable items are useful for modeling various strategies of an organization. Thus, it is necessary to identify stable items. We propose the notion of degree of stability of an item. We design an algorithm for clustering items in multiple databases based on degree of stability. The proposed technique is useful and effective.

References

[1] Adhikari A. and Rao R., "Enhancing Quality of Knowledge Synthesized from Multi-Database Mining," *Computer Journal of Pattern Recognition Letters*, vol. 28, no. 16, pp. 2312-2324, 2007.

[2] Adhikari A. and Rao R., "Synthesizing Heavy Association Rules from Different Real Data Sources," *Computer Journal of Pattern Recognition Letters*, vol. 29, no. 1, pp. 59-71, 2008.

[3] Agrawal R., Imielinski T., and Swami A., "Mining Association Rules between Sets of Items in Large Databases," *in Proceedings ACM SIGMOD Conference Management of Data*, Canada, pp. 207-216, 1993.

[4] Agrawal R. and Srikant R., "Fast Algorithms for Mining Association Rules," *in Proceedings of 20th Very Large Databases (VLDB) Conference,* Santiago, pp. 487-499, 1994.

[5] Bluman G., *Elementary Statistics*: *A Step by Step Approach*, McGraw Hill, 2006.

[6] Brockwell J. and Richard D., *Introduction to Time Series and Forecasting*, Springer, 2002.

[7] Estivill V. and Yang J., "Fast and Robust General Purpose Clustering Algorithms," *Computer Journal of Data Mining and Knowledge Discovery*, vol. 8, no. 2, pp. 127-150, 2004.

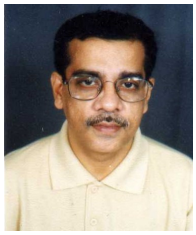
[8] Frequent Itemset Mining Dataset Repository, http://fimi.cs.helsinki.fi/data, 2004.

[9] Han J., Pei J., and Yiwen Y., "Mining Frequent Patterns without Candidate Generation," *in Proceedings ACM SIGMOD Conference Management of Data*, New York, pp. 1-12, 2000.

[10] Jain K., Murty N., and Flynn J., "Data Clustering: A Review," *Computer Journal of ACM Computing Surveys*, vol. 31, no. 3, pp. 264-323, 1999.
[11] Leonard M. and Wolfe B., "Mining Transactional and Time Series Data," *SUGI 30 Proceedings,* UK, pp. 080-30, 2005.
[12] Liu B., Ma Y., and Lee R., "Analyzing the Interestingness of Association Rules from the Temporal Dimension," *IEEE International Conference on Data Mining,* Silicon Valley, 377-384, 2001.
[13] Liu L., *Elements of Discrete Mathematics*, McGraw-Hill, 1985.
[14] Tan N., Kumar V., and Srivastava J., "Selecting the Right Interestingness Measure for Association Patterns," *in Proceedings of SIGKDD Conference*, Canada, pp. 32-41, 2002.
[15] UCI ML Repository Content Summary, http:// www.ics.uci.edu/~mlearn /MLSummary.html, 2002.
[16] Wu X., Zhang C., and Zhang S., "Database Classification for Multi Database Mining," *Computer Journal of Information Systems,* vol. 30, no. 1, pp.71-88, 2005.
[17] Wu X. and Zhang S., "Synthesizing High-Frequency Rules from Different Data Sources," *Computer Journal of IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no .2, pp. 353-367, 2003.
[18] Yang K. and Shahabi C., "On the Stationarity of Multivariate Time Series for Correlation-Based Data," *in Proceedings of ICDM*, Italy, pp. 805-808, 2005.
[19] Zhang T., Ramakrishnan R., and Livny M., "BIRCH: A New Data Clustering Algorithm and Its Applications," *Computer Journal of Data Mining and Knowledge Discovery*, vol. 1, no. 2, pp. 141-182, 1997.
[20] Zhang S., Wu X., and Zhang C., "Multi Database Mining," *Computer Journal of IEEE Computational Intelligence Bulletin*, vol. 2, no. 1, pp. 5-13, 2003.
[21] Zhang S., Zhang C., and Wu X., *Knowledge Discovery in Multiple Databases*, Springer, 2004.

Jhimli Adhikari received Master of computer application from Jadavpur University, Kolkata. At present, she is a lecturer in the Department of Computer Science, Narayan Zantye College, India. Currently, she is part-time PhD student in the Department of Computer Science and Technology, Goa University, India.

Pralhad Rao received Master of science and PhD degrees, both in mathematics, from Karnataka University and Indian Institute of Technology, Mumbai, respectively. At present, he is a professor in the Department of Computer Science and Technology, Goa University, India. His areas of interest include graph theory, knowledge discovery and data mining and data warehousing. He has twenty seven published papers.



Animesh Adhikari received Master of technology in computer science and PhD in computer science and technology degrees from Indian Statistical Institute and Goa University, respectively. At present, he is a lecturer in the Department of Computer Science, S P Chowgule College, India. His areas of interest include data mining and knowledge discovery, decision support systems, database systems and artificial intelligence. He has twelve papers published in different international journals and conferences.