Mining Very Large Databases Using Software Agents

Chengqi Zhang, Xiaowei Yan, Shichao Zhang, and Paul Kennedy
Faculty of Information Technology
University of Technology Sydney
Sydney, NSW 2007, Australia
{chengqi, xyan, zhangsc, paulk}@it.uts.edu.au

Abstract— **Some databases are simply large (e.g., with terabytes of data). A number of potential and diverse patterns in databases would sufficiently be mined so as to support various decisions in applications. This research advocates a re-recognition and re-cogitation to how to discover very large databases. It also presents a new technique mining very large databases will be developed on different hierarchy for possible different applications. To do so, a system MVLDB (Mining Very Large Databases) for such as analyzing, partitioning, re-organizing, and mining databases is built by classifying and clustering in this paper. The system uses software agents in all aspects including sampling, data clustering and data mining.**

Keywords: Data mining, KDD, software agent, information gathering.

I. Introduction

As have known, mining associations in databases is helpful to applications. However, there are still a key limitation in previous methods. It is that previous mining techniques are inadequate to meet applications due to the fact that the results mined in a huge databases are the same as that several sands in sea.

This limitation is apparent. We had done some experiments for mining large scale databases by previous algorithms. Both our experiments and public experiments in current reports show that only fewer association rules are of interest in such huge databases. Broadly speaking, this is the same as that we take several sands in sea. For example, we mined a database with 100,000 transactions (from the Synthetic Classification Data Sets in Internet "http://www.kdnuggets.com/") using the algorithm in [1]. There are only 157 association rules of interest, which minimum support is 0.0015 and minimum confidence is 0.65. In particular, only about 20,000 transactions in the database are fitted by the 157 rules of interest. This means that huge amounts of information in the database are wasted. Certainly, the use of these rules to applications is doubted. Generally, most of the data in a given database are useful to applications. They would sufficiently be used when we make decisions. But human experts are unable to tackle huge databases for decision purposes. The goal of data mining is to develop models and tools for assisting human experts to deal with very large databases. Unfortunately, the effects of previous models and tools are still unsatisfactory. This pitfall may become a heavy-fisted problem to impact the progress and applications of data mining.

On the other hand, previous mining techniques are inadequate to meet applications. For example, we may discover association rules like "*milk* ← *newspaper*" in the market basket data of a supermarket. The rules like "*milk* ← *newspaper*" may be useful to decisions of such as predicting and buying in. However, we would not place "milks" near to "newspapers" for the placement of the supermarket. This means that the mined results in market basket data don't meet some applications such as placement in the supermarket. Also, as have seen, the mined results in market basket data are only dealt with a small kind of data in the database: frequent itemsets.

To overcome this pitfall, we argue a re-recognition and re-cogitation to how to discover very large databases. And a system MVLDB (Mining Very Large Databases) for such as analyzing, partitioning, re-organizing, and mining databases is built by classifying and clustering in this paper. The system uses software agents in all aspects including sampling, data clustering and data mining.

The rest of this paper is organized as follows. We begin with introducing the architecture of MVLDB in Section *II*. In Section *III* we presents the sampling agent, which is responsible for selecting an appropriate subset of a given database. Section *IV* shows the data mining agent. Section *V* advocates a database clustering agent. And we simply conclude this paper in the last section.

II. Architecture of MVLDB

This section discusses the facilities which MVLDB agents offer the users in mining very large databases. The architecture of MVLDB is depicted by Figure 1.

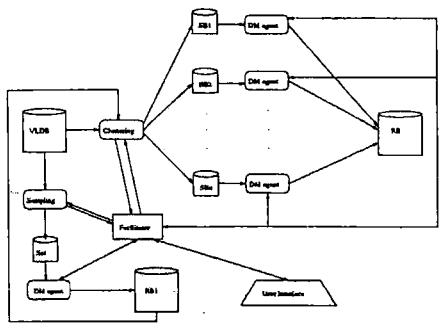

Fig. 1. The MVLDB architecture

In the figure, $VLDB$ denotes a very large database, SB_i $(1 \leq i \leq n)$ denotes a suitable set deriving from $VLDB$, Set is a subset of $VLDB$, $RB1$ denotes the base of all approximating rule, RB denotes association rule base. DM agent stands for a data mining agent. The facilitator is responsible for managing the agents and interfacing users. The sampling is an agent responsible for getting a appropriate sample from $VLDB$. The clustering is an agent responsible for analyzing and partitioning $VLDB$ according to applications. The user interface is also an agent responsible for inputting requirements to the system and outputting the mined rules to users. These agents share their information through the facilitator in the system. The functions of agents will be presented in following sections.

The work procedure of MVLDB is as follows. A user sends a "mining" request to MVLDB via the user interface to the facilitator. The facilitator firstly calls the sampling agent to extract a suitable subset of a database given by the user. Secondly, the facilitator calls a DM agent to mine the subset to obtain a set of approximating association rules in $RB1$. Thirdly, the DM agents pass the facilitator the mining rules. Fourthly, the facilitator calls the clustering agent to analyze and partition $VLDB$ by $RB1$ and requirements. And the clustering agent creates some data sets from $VLDB$. Fifthly, the facilitator calls DM agents to mine the clustered data sets. The DM agents pass the facilitator the mining rules. Finally, the facilitator forwards the user the mined results.

III. Sampling Agent

The sampling agent is responsible for extracting a appropriate sample from $VLDB$. It is described as follows.

In probability theory, if a situation is such that only two outcomes, often called success and failure, are possible, it is usually called a *trial*. The variable element in a trial is described by a probability distribution on a sample space of two elements, 0 representing failure and 1 success; this distribution assigning the probability $1-\theta$ to 0 and θ to 1, where $0 \leq \theta \leq 1$. Suppose we consider n independent repetitions of a given trial. The variable element in these is described by a probability distribution on a sample space of 2^n points, the typical point being $x = (x_1, x_2, \cdots, x_n)$, where each x_i is 0 or 1, and x_i represents the result of the ith trial. The appropriate probability distribution is defined by

$$p_\theta(x) = \theta^{m(x)}(1-\theta)^{n-m(x)},$$

where $m(x) = \sum_{i=1}^{n} x_i$ is the number of 1s in the results of the n trials, this being so since the trials are independent.

Given an x in this situation it seems reasonable to estimate θ by $m(x)/n$, the proportion of successes obtained. This seems in some sense to be a 'good' estimate of θ.

In this way, a database D can be taken as a trial. For any itemset A, it is 1 if the itemset A occurs in a transaction T (written as $T(A)$), else it is 0 (written as $\neg T(A)$). Suppose the probability of A occurring in the database is p and the probability of A not occurring is $q = 1 - p$. Hence, this given database can be taken as a Bernoulli trial according to the definition in [4]. In particular, we can approximate the probability p of A by central limit theorem.

A. Evaluating The Sizes of Samples

In our sampling agent, central limit theorem is applied to estimate the size of samples.

The central limit theorem is one of the most remarkable results in probability theory. Loosely put, it states that the sum of a large number of independent random variables has a distribution that is approximately normal. Hence it not only provides a simple method for computing approximating probabilities for sums of independent random variables, but it also helps explain the remarkable fact that the empirical frequencies of so many natural populations exhibit bell-shaped (that is, normal) curves. In its simplest form the central limit theorem is as follows.

Let $X_1, X_2, \cdots$ be a sequence of independent and identically distributed random variables, each having finite mean $E(X_i) = \mu$ and $Var(X_i) = \sigma^2$. Then the distribution of

$$\frac{X_1 + \cdots + X_n - n\mu}{\sigma\sqrt{n}}$$

tends to the standard normal as $n \to \infty$. That is,

$$P\{\frac{X_1 + \cdots + X_n - n\mu}{\sigma\sqrt{n}} \leq a\} \to \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{a} e^{-x^2/2} dx \quad (1)$$

$$as \quad n \to \infty.$$

This means that we can approximate probabilities of random variables by the central limit theorem. Readers are referred to [4] for other concepts and theorems.

We will set up a new mining model in next section, which applies the central limit theorem to mine approximate frequent itemsets from a sample of a given large databases. So we now estimate the sample size as follows.

Theorem 1: Let D be a large database, $T_1, T_2, \cdots, T_m$ be the transactions in D, A be an itemset in D, $\eta > 0$ be the degree of asymptotic to frequent itemsets, $\xi \geq 0$ be the upper probability of $P[|Ave(X_n) - p| \leq \eta]$, where $Ave(X_n)$ is the average of A occurring in n transactions in D. Suppose records in D are matched Bernoulli trials. If n random records of D is enough for determining the approximate frequent itemsets in D according to central limit theorem, n must be as follows:

$$n \geq \frac{z^2_{(1+\xi)/2}}{4\eta^2} \quad (2)$$

where z_x is a standard normal distribution function, which can find out it from the Appendix in [4].

Proof: From the given conditions in this theorem, we take

$$P(|Ave(X_n) - p| \leq \eta) = \xi.$$

Clearly,

$$P(|Ave(X_n) - p| \leq \eta) = P(-\eta \leq (Ave(X_n) - p) \leq \eta)$$

$$= P(\frac{-\eta}{1/(2\sqrt{n})} \leq (\frac{Ave(X_n) - p)}{1/(2\sqrt{n})} \leq \frac{\eta}{1/(2\sqrt{n})})$$

$$\approx N(2\eta\sqrt{n}) - N(-2\eta\sqrt{n})$$

$$= 2N(2\eta\sqrt{n}) - 1,$$

and for this probability to equal ξ we need

$$N(2\eta\sqrt{n}) = \frac{1}{2}(1 + \xi)$$

which is satisfied by

$$2\eta\sqrt{n} = z_{(1+\xi)/2}.$$

the required value for n then is

$$n \geq \frac{z^2_{(1+\xi)/2}}{4\eta^2}.$$

□

We now illustrate the use of this theorem by an example as follows.

Example 1: Suppose a new process is available for doping silicon chips, used in electronic devices. p (unknown) is the probability that each chip produced in this way is defective. We assume that the defective chips are independent of each other. How many chips, n, must we produce and test so that the proportion of defective chips found ($Ave(X_n)$) does not differ from p by more than 0.01, with probability at least 0.99? That is, we want n such that

$$P(|Ave(X_n) - p| < 0.01) > 0.99,$$

$\eta = 0.01, \xi = 0.995, z_{0.95} = 2.57$, we have

$$n = \frac{2.57^2}{4 * 0.01^2} = 16513,$$

considerably smaller than the value $n = 27000$ that is needed by using the approximating model in Chernoff bounds [8].

B. Generating Random Data Subset

Based on Theorem 1, we can obtain a random sample from the database in two steps: (1) generate n random numbers, where n is determined by the central limit theorem; (2) choose n transactions in the database according to the random numbers. In this subsection, we present the procedure of generating random databases.

Generally, it is difficult to apply absolutely random numbers to choose tuples from given database. So, pseudo-random numbers are used for choosing tuples from the database so as to control the generated random database as a sample.

There are many methods of generating pseudo-random numbers. Here we can use one from the following pseudo-random number generators.

Suppose pseudo-random numbers is as:

$$x_0, x_1, x_2, \cdots$$

The ith pseudo-random number ($i > 0$) can be determined as

$$x_i = (ax_{i-1} + b)\ MOD\ m, \tag{3}$$

where a, b, m are constants. The sequence $x_0, x_1, x_2, \cdots$ is a sequence of integers between 0 and $m - 1$.

In the above formula, if $a = 1$, the another linear pseudo-random number generator is as:

$$x_i = (x_{i-1} + b)\ MOD\ m, \tag{4}$$

if $b = 0$, a simple linear pseudo-random number generator is as:

$$x_i = ax_{i-1}\ MOD\ m. \tag{5}$$

Apparently, the first form of linear pseudo-random number generators is the best one. It has higher stochastic degree when x_0, a, b, m are mutually prime number.

For simplicity, here we only present the algorithm of linear pseudo-random number generator given in the above (3). Generating pseudo-random numbers is as follows.

Procedure 1: RandomNumber

Input: a: integer constant, b: integer constant, m: real database size,
n: random database size, x_0: first pseudo-random number;

Output: X: set of pseudo-random numbers;

(1) **let** $X \leftarrow \emptyset$;
let $a \leftarrow$ a bigger prime number;
let $b \leftarrow$ a prime number is different from a;
read x_0 a prime number is different from a and b;
let $c \leftarrow x_0$;
let $X \leftarrow X \cup \{x_0\}$;
(2) **while** $|X| \neq n$ **do begin**
  let $x_i \leftarrow (a * c + b)\ MOD\ m$;
  if $x_i \notin X$ **then**
    let $X \leftarrow X \cup \{x_i\}$;
  let $c \leftarrow x_i$;
end
(3) **for** $i = 1$ **to** n do
  output random number x_i in X;
(4) **endall.**

The procedure RandomNumber generates n random numbers, where n is equal to the sample size. We can apply these numbers to select n instances as a sample. Step (1) does initialization. To generate m different random numbers, a, b, and x_0 would be three different prime number, where m is equal to the size of a given large database. Step (2) generates n random numbers and saves them into set X, which each random number is less than m according

to the operator "MOD". Step (3) outputs the generated random numbers.

The method of generating random database from real database is as: (1) generating a set X of pseudo-random numbers, where $|X| = n$; and (2) generating the random database RD from D using pseudo-random number set X. That is, for any $x_i \in X$, get $(x_i + 1)$th record of D and append it into RD. It can be implemented as follows. Generating random index database from real database is as follows.

Procedure 2: RandomDatabase
Input: D: original real database;
Output: RD: random database;
(1) **let** $RD \leftarrow \emptyset$;
call procedure RandomNumber to generate the set X of n random numbers;
(2) **for** any numbers x_i in X **do begin**
let $j \leftarrow x_i + 1$;
let $record \leftarrow$ the jth record in D;
let $RD \leftarrow RD \cup \{record\}$;
end
(3) **output** the random database RD;
(4) **endall.**

The procedure RandomDatabase generates a sample RD of a given large database D by the set of random numbers. Step (1) first assigns a initial value $\emptyset$ to the sample RD, and then generates the set X of n random numbers by calling the procedure RandomNumber. Step (2) generates the random data subset RD of D by the numbers in X. The ith transaction in RD is the x_ith record in the database D. Because the random numbers are required different from each other when they are generated, each record in the database D can be dealt with at most one time. And the records in RD are some random instances selected from the database D. Step (3) outputs the sample RD.

Note that generating random database RD of the given database D doesn't mean to establish a new database RD. It only needs to build a view RD over D.

IV. Data Mining Agent

There are many proposed methods of measuring the uncertainty of association rules. We advocate to apply Dempster-Shafer theory (D-S) [7] to capture the uncertainty of association rules in this paper.

The Dempster-Shafer theory uses a number in the range $[0, 1]$ to indicate belief in a hypothesis given a piece of evidence. This number is the degree to which the evidence supports the hypothesis.

Let Ω be a finite non-empty set and call it the frame of discernment. The impact of each distinct piece of evidence on the subsets of Ω is represented by a function called a basic probability assignment (bpa). A bpa is a generalization of the traditional probability density function; the latter assigns a number in the range $[0, 1]$ to every singleton of Ω such that the numbers sum to 1. Using 2^Ω, the enlarged domain of all subsets of Ω, a bpa denoted m assigns a number in $[0, 1]$ to every subset of Ω such that the numbers sum to 1. We present formally the needed definitions as follows.

Definition 1: A function

$$m : 2^\Omega \rightarrow [0, 1]$$

is called a mass function if it satisfies
(1) $m(\emptyset) = 0$,
(2) $\sum_{A:A\subseteq\Omega} m(A) = 1$.

A mass function is a basic probability assignment to all subsets X of Ω. The quantity $m(A)$ is called A's mass or basic probability value. This term has been given to this measure in evidence theory because it represents the exact amount of belief committed to the proposition represented by subset A of Ω. A subset A of a frame Ω is called a focal element of a mass function m over Ω if $m(A) > 0$.

Definition 2: A function

$$Bel : 2^\Omega \rightarrow [0, 1]$$

is called a belief function if it satisfies
(1) $Bel(\emptyset) = 0$,
(2) $Bel(\Omega) = 1$,
(3) for any collection $A_1, A_2, \cdots, A_n (n \geq 1)$ of subsets of Ω,

$$Bel(A_1 \cup A_2 \cup \cdots \cup A_n) \geq \sum_{I\subseteq\{1,2,\cdots,n\}, I\neq\emptyset} (-1)^{|I|+1} Bel(\cap_{i\in I} A_i).$$

A belief function assigns a measure of our total belief to each subset of Ω.

Definition 3: If m is a mass function on Ω, then the function Bel defined by

$$Bel(A) = \sum_{B:B\subseteq A} m(B) \text{ for all } A \subseteq \Omega$$

is a belief function and

$$m(A) = \sum_{B\subseteq A} (-1)^{|A-B|} Bel(B) \text{ for all } A \subseteq \Omega.$$

$Bel(A)$ represents the degree to which the actual evidence supports A, i.e. it measures the credibility of A. We are also able to define the degree to which the evidence fails to refute A, i.e. the degree to which A remains plausible.

Given a belief function Bel, the function

$$Pl(A) = \sum_{B:B\cap A\neq\emptyset} m(B) = 1 - Bel(\overline{A}).$$

is called a plausibility function. The function Pl is in one-to-one correspondence with the belief function induced by the same basic belief mass. It is simply another way of presenting the same information and could be avoided, except that it provides a convenient alternate representation of our beliefs.

Now we can define conditional mass distribution. For $E \subseteq \Omega$, if $Bel(E) \neq 0$, then we define for all $A \subseteq E$

$$m(A|E) = \frac{m(A)}{Bel(E)},$$

and $m(A|E) = 0$, otherwise.

Thus we obtain

$$Bel(A|E) = \frac{Bel(A \cap E)}{Bel(E)},$$

and

$$Pl(A|E) = \frac{Pl(A \cup \overline{E}) - Pl(\overline{E})}{1 - Pl(\overline{E})}.$$

For $\forall A, B \subseteq \Omega$, let m_A be a mass function such that

$$m_A(A) = 1, \qquad m_A(elsewhere) = 0.$$

Then
(1) A mass function m and m_A are combinable if and only if

$$Bel(\overline{A}) < 1.$$

(2) If m and m_A are combinable (combining mass operators $\oplus$, Bel_A and Pl_A were defined in [7]), denote

$$Bel(B|A) = (Bel \oplus Bel_A)(B), Pl(B|A) = (Pl \oplus Pl_A)(B).$$

Then

$$Bel(B|A) = \frac{Bel(B \cup \overline{A}) - Bel(\overline{A})}{1 - Bel(\overline{A})}$$

and

$$Pl(B|A) = \frac{Pl(B \cap A)}{Pl(A)}.$$

This model can also be applied to measure the uncertainties of association rules. Let A be an itemset, we denote the occurring set of A with A_o. Then an association rule is a relationship of the form $A \rightarrow B$, where A and B are sets of items and $A \cup B = \emptyset$. Each association rule has a support factor $g(A \cup B) = Bel(A_o \cap B_o)$, a confidence factor $f(A, B) = Bel(B_o|A_o)$ and a plausibility factor $Pl(A, B) = Pl(B_o|A_o)$.

In the same reasons, for an association rule $A \rightarrow B$, both support and confidence must be greater than or equal to some user specified minimum support (*minsupp*) and minimum confidence (*minconf*) thresholds respectively.

We now demonstrate how to apply this model to measure association rules with a database. For simplicity, we assume $Bel(X) + Bel(\overline{X}) = 1$ for all $X \subseteq \Omega$. In this case, $Bel = Pl$.

Example 2: A transaction database TD with 10 transactions in Table 1 is obtained from a grocery store. Let $A = bread$, $B = coffee$, $C = tea$, $D = sugar$, $E = beer$, $F = butter$. Assume $minsupp = 0.3$. The mass function m is listed in Table 2. For itemset $B \cup D$, the occurring set B_o of B is $\{1, 2, 3, 4, 6, 9, 10\}$, the occurring set D_o of D is $\{1, 2, 3, 4, 6, 10\}$ and the occurring set $(B \cup D)_o$ of $B \cup D$ is $\{1, 2, 3, 4, 6, 10\}$, then

$$Bel(B) = \sum_{X: X \subseteq B_o} m(B_o) = 0.75$$

$$Pl(B) = \sum_{X: X \cap B_o \neq \emptyset} m(B_o) = 0.75$$

$$Bel(D) = \sum_{X: X \subseteq D_o} m(D_o) = 0.6525$$

$$Pl(D) = \sum_{X: X \cap D_o \neq \emptyset} m(D_o) = 0.75$$

$$Bel(B \cup D) = \sum_{X: X \subseteq (B \cup D)_o} m((B \cup D)_o) = 0.75$$

$$Pl(B \cup D) = \sum_{X: X \cap (B \cup D)_o \neq \emptyset} m((B \cup D)_o) = 0.75$$

so,

$$Bel(D \cup \overline{B}) = 0.90625$$

$$Bel(\overline{B}) = 0.25$$

then

$$Bel(D|B) = \frac{Bel(D \cup \overline{B}) - Bel(\overline{B})}{1 - Bel(\overline{B})}$$

$$= \frac{0.90625 - 0.25}{1 - 0.25} = 0.875$$

hence, the support and confidence are as follows:

$supp(B \cup D) = 0.75 = 75\%, conf(B, D) = 0.875 = 87.5\%$
The other itemsets is listed in the following Table 4 and 5.

Table 1: Transaction database TD

Transaction ID	Items
T_1	A, B, D
T_2	A, B, C, D
T_3	B, D
T_4	B, C, D, E
T_5	A, C, E
T_6	B, D, F
T_7	A, E, F
T_8	C, F
T_9	B, C, F
T_{10}	A, B, C, D, F

Table 2: A mass function m

event sets	$m(X)$
$\{T_1\}$	0.09375
$\{T_2\}$	0.125
$\{T_3\}$	0.0625
$\{T_4\}$	0.125
$\{T_5\}$	0.09375
$\{T_6\}$	0.09375
$\{T_7\}$	0.09375
$\{T_8\}$	0.0625
$\{T_9\}$	0.09375
$\{T_{10}\}$	0.15625

where $\Omega = \{T_1, T_2, \cdots, T_{10}\}$ (the simple form of Ω is $\Omega = \{1, 2, \cdots, 10\}$), $m(\emptyset) = 0$, $m(\{T_i\}) = |T_i| / \sum |T_j|$, and $m(Y) = 0$ for $|Y| > 1$ and $Y \subseteq \Omega$.

Table 3: The occurring set of single item

Item	occurring set X_o	$Bel(X)$
A	$\{1,2,5,7,10\}$	0.5625
B	$\{1,2,3,4,6,9,10\}$	0.75
C	$\{2,4,5,8,9,10\}$	0.65625
D	$\{1,2,3,4,6,10\}$	0.65625
E	$\{4,5,7\}$	0.3125
F	$\{6,7,8,9,10\}$	0.5

For simplicity, the number of the occurring set of the itemset is greater than or equal to 2.

Table 4: The occurring set of some itemset supports

Item	occurring set X_o	$Bel(X)$
A, B	$\{1,2,10\}$	0.375
A, C	$\{2,5,10\}$	0.375
A, D	$\{1,2,10\}$	0.375
B, C	$\{2,4,9,10\}$	0.5
B, D	$\{1,2,3,4,6,10\}$	0.65625
B, F	$\{6,9,10\}$	0.34375
C, D	$\{2,4,10\}$	0.40625
C, F	$\{8,9,10\}$	0.3125
A, B, D	$\{1,2,10\}$	0.375
B, C, D	$\{2,4,10\}$	0.40625

Table 5: Some association rules and confidences

Left X	Right Y	$Bel(Y \cup \overline{X})$	$Bel(\overline{X})$	$Bel(Y\|X)$	$Pl(Y\|X)$
A	B	0.8125	0.4375	0.667	0.667
A	C	0.8125	0.4375	0.667	0.667
A	D	0.8125	0.4375	0.667	0.667
B	C	0.75	0.25	0.667	0.667
B	D	0.90625	0.25	0.875	0.875
B	F	0.59375	0.25	0.458	0.458
C	D	0.75	0.34375	0.619	0.619
C	F	0.65625	0.34375	0.476	0.476
A, B	D	1	0.625	1	1
B, C	D	0.90625	0.5	0.8125	0.8125

Let $minsupp = 0.3$ and $minconf = 65\% = 0.65$, then $A \wedge B \rightarrow D$, $B \rightarrow D$ and $B \wedge C \rightarrow D$ can be extracted as rules. If $minsupp = 0.5$ and $minconf = 65\% = 0.65$, then only $B \rightarrow D$ can be discovered as an association rule.

V. Databases Clustering

Data analysis underlies many computing applications, either in a design phase or as part of their on-line operations. Data analysis procedures can be dichotomized as either exploratory or confirmatory, based on the availability of appropriate models for the data source, but a key element in both types of procedures (where for hypothesis formation or decision-making) is the grouping, or classification of measurements based on either (i) goodness-of-fit to a postulated model, or (ii) natural groupings (clustering) revealed through analysis.

Clustering is useful in several exploratory pattern-analysis, grouping, decision-making, and machine-learning situations, including data mining, document retrieval, image segmentation, and pattern classification. However, in many such problems, there is little prior information (e.g., statistical models) available about the data, and the decision-maker must make as few assumptions about the data as possible. It is under these restrictions that clustering methodology is particularly appropriate for the exploration of interrelationships among the data points to make an assessment (perhaps preliminary) of their structure.

To partition databases, we use clustering in this paper. For simplicity, we select a simple method of clustering. Certainly, to meet diverse applications, some sophisticated models of clustering would be constructed. This problem will be introduced in our other papers soon.

A. Similarity Measure

Since similarity is fundamental to the definition of a cluster, a measure of the similarity between two patterns drawn from the same feature space is essential to most clustering procedures. Because of the variety of feature types and scales, the distance measure (or measures) must be chosen carefully. It is most common to calculate the dissimilarity between two patterns using a distance measure defined on the feature space. We will focus on a well-known distance measure as follows.

Let $I = \{i_1, i_2, \cdots, i_N\}$ be a set of N distinct literal called *items*. D is a set of variable length transactions over I. Each transaction contains a set of items $i_1, i_2, \cdots, i_k \in I$. A transaction has an associated unique identifier called TID. In general, a set of items is called an *itemset*. The number of items in an itemset is the *length* (or the *size*) of an itemset.

For a association rule t: $A \rightarrow B$ of D, the content of t is a set of all items in A or B of t, denoted by $R(t)$. Or

$$R(t) = \{i | i \in A \vee i \in B\}.$$

We define the distance between association rules t_1 and t_2 based on their contents:

$$m_R(t_1, t_2) = \frac{|R(t_1) \cap R(t_2)|}{|R(t_1) \cup R(t_2)|}.$$

Certainly, the larger $m_R(t_1, t_2)$ is, the smaller the distance between association rules t_1 and t_2 is.

Example 3: Let t_1, t_2 and t_3 be three association rules, and $R(t_1) = \{a_1, a_2, b_2, c_1\}$, $R(t_2) = \{a_2, b_1, b_2, c_1\}$, $R(t_3) = \{a_1, a_2, b_2, c_1, c_2\}$. Then

$$m_R(t_1, t_2) = \frac{|R(t_1) \cap R(t_2)|}{|R(t_1) \cup R(t_2)|} = \frac{3}{5} = 0.6.$$

$$m_R(t_1, t_3) = \frac{|R(t_1) \cap R(t_3)|}{|R(t_1) \cup R(t_3)|} = \frac{4}{5} = 0.8.$$

$$m_R(t_2, t_3) = \frac{|R(t_2) \cap R(t_3)|}{|R(t_2) \cup R(t_3)|} = \frac{3}{6} = 0.5.$$

We can apply this measure to partition some databases for applications.

B. Data Wrapping And Partitioning Database

As have seen, previous mining techniques are inadequate to meet applications due to the fact that the results mined in a huge databases are the same as that several sands in sea. So we advocates a re-recognition and re-cogitation to how to discover very large databases. To do so, databases would be partitioned according different applications. As an attempt, we propose to firstly get a subset of a given database by sampling. And this subset is mined. Secondly, the mined results are clustered. Finally, the clustered results and requirements are applied to partition the database.

To partition a very large database, we can use the approximating rules to classify the data in the database. This is performed as (1) clustering the approximating rules; (2) classifying the data in the database according to the classes of the approximating rules; and (3) taking each data class as a subset of the database. Then we can mining each subset according to applications.

However, there may not be any patterns in data. For example, suppose that some transactions (market baskets) have been obtained from a supermarket. This involves spelling out the attribute value (item's name) for each transaction, separated by commas, and purchased by a customer. Three of the transactions are

Skim milk, Sunshine bread, GIS sugar.
Pauls milk, Colces bread, Sunshine biscuit.
Yeung milk, B&G bread, Sunshine chocolate.

The first customer bought some Skim milk, Sunshine bread, and GIS sugar; the second customer bought some Pauls milk, Colces bread, and Sunshine biscuit; and the third customer bought some Yeung milk, B&G bread, and Sunshine chocolate. The potential patterns in the three transactions are not clear. Sometimes we need to predict the possible amount of purchasing a new product by association rules. And we sometimes need to place products by association rules. For different applications, we must offer different association rules. Unfortunately, previous mining algorithms are inadequate to the real-world applications. In our opinion, we would develop some new techniques to re-recognize and re-cogitate the data in large databases. In our database clustering agents, this idea is considered. We illustrate it using the above transactions. For the above three transaction, if we transform them as

milk, bread, sugar.
milk, bread, biscuit.
milk, bread, chocolate.

You can be easy to get "if a customer buys milk, he/she would buy bread." And if we transform them as

Skim, Sunshine, GIS.
Pauls, Colces, Sunshine.
Yeung, B&G, Sunshine.

You can also be easy to get "customers like to buy the products of Sunshine".

As you have seen, after databases are clustered, some potential patterns become controllable.

VI. Conclusions

As have seen, to discover association rules in large scale databases has received much attention recently [1], [3], [8], [9]. But previous mining techniques are inadequate to meet the requirement of many real-world applications. For this reason, the proposed approach was focused on the above problem. We have proposed basic techniques for mining association rules in very large databases based on software agents. Also, a new method was proposed to discover more useful patterns enough to support applications by clustering, which most of data are covered/fitted by these patterns. The main contributions in this paper are as follows.

- The architecture of mining very large database based on agents is built.
- Proposed to estimate the possible association rules by sampling.
- Advocated to re-recognize and re-cogitate the data in large databases. And the data in databases would be clustered before databases are discovered.

References

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

[2] Bond, A. and Gasser, L. (eds), Readings in Distributed artificial Intelligence, Morgan Kaufman Publishers, 1988.

[3] S. Brin, R. Motwani and C. Silverstein, Beyond market baskets: Generalizing association rules to correlations. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1997: 265-276.

[4] R. Durrett, Probability: Theory and Examples, *Duxbury Press*, 1996.

[5] H. Kargupta, B. Park, D. Hershberger, and E. Johnson, Collective Data Mining: A New Perspective Toward Distributed Data Mining. *Advances in Distributed and Parallel Knowledge Discovery*, Eds: Hillol Kargupta and Philip Chan. MIT/AAAI Press, 1999.

[6] A. Prodromidis, P. Chan, and S. Stolfo, Meta-learning in distributed data mining systems: Issues and approaches, In *Advances in Distributed and Parallel Knowledge Discovery*, H. Kargupta and P. Chan (editors), AAAI/MIT Press, 2000.

[7] G. Shafer, A mathematical theory of evidence. Princeton University Press, Princeton, 1976.

[8] R. Srikant and R. Agrawal, Mining generalized association rules. *Future Generation Computer Systems*, Vol. 13, 1997: 161-180.

[9] Shichao Zhang and Xindong Wu, Large Scale Data Mining Based on Data Partitioning, Applied Artificial Intelligence, forthcoming.