

Database Classification for Multi-Database Mining

Xindong Wu^(*), Chengqi Zhang⁽⁺⁾ and Shichao Zhang⁽⁺⁾

^(*) Department of Computer Science

University of Vermont

Burlington, Vermont 05405, USA

⁽⁺⁾ Faculty of Information Technology, University of Technology, Sydney

PO Box 123, Broadway NSW 2007, Australia

xwu@cs.uvm.edu; {chengqi, zhangsc}@it.uts.edu.au

Abstract

Many large organizations have multiple databases distributed in different branches, and therefore multi-database mining is an important task for data mining. To reduce the search cost in the data from all databases, we need to identify which databases are most likely relevant to a data mining application. This is referred to as database selection. For real-world applications, database selection has to be carried out multiple times to identify relevant databases that meet different applications. In particular, a mining task may be without reference to any specific application. In this paper, we present an efficient approach for classifying multiple databases based on their similarity between each other. Our approach is application-independent.

Keywords: database selection, classification, multi-database mining.

1 Introduction

If a large company is a comprehensive organization that has different types of businesses with different meta-data structures, the databases within the company would have to be classified before the data is analyzed. For example, if an inter-state company has 25 branches including five supermarkets, seven manufacturing units, and 13 investment trusts, we cannot directly apply existing data mining techniques to the union of the 25 databases from the 25 branches. We would need to classify them into three classes according to the three types of businesses, and then mine the databases in each class.

Consider six databases D_1, D_2, \dots, D_6 as follows.

$$\begin{aligned} D_1 &= \{(A, B, C, D); (B, C); (A, B, C); (A, C)\} \\ D_2 &= \{(A, B); (A, C); (A, B, C); (B, C); (A, B, D)\} \\ D_3 &= \{(B, C, D); (A, B, C); (B, C); (A, D)\} \\ D_4 &= \{(F, G, H, I, J); (E, F, H); (F, H)\} \\ D_5 &= \{(E, F, H, J); (F, H); (F, H, J); (E, J)\} \\ D_6 &= \{(F, H, I, J); (E, H, J); (E, F, H); (E, I)\} \end{aligned}$$

where each database has several transactions, separated by semicolons (;), and each transaction contains several items, separated by commas (,).

Let $minsupp = 0.5$. We have the frequent itemsets in each database as follows.

- $D_1 : A, B, C, AB, AC, BC, ABC$
 $D_2 : A, B, C, AB$
 $D_3 : A, B, C, BC$
 $D_4 : F, H, FH$
 $D_5 : E, F, H, J, EJ, FH, FJ, FHJ$
 $D_6 : E, F, H, I, J, EH, FH, HJ.$

Now let us put the six databases into $TD = D_1 \cup D_2 \cup \dots \cup D_6$, with 24 transactions. When $minsupp = 0.5$, the frequent itemsets from each database are listed in Table 1.

Table 1 *Frequent Itemsets from Each Database in TD*

<i>Itemset</i>	<i>Frequency</i>	$\geq minsupp$
<i>A</i>	9	<i>no</i>
<i>B</i>	10	<i>no</i>
<i>C</i>	10	<i>no</i>
<i>AB</i>	6	<i>no</i>
<i>AC</i>	6	<i>no</i>
<i>BC</i>	8	<i>no</i>
<i>ABC</i>	4	<i>no</i>
<i>E</i>	6	<i>no</i>
<i>F</i>	8	<i>no</i>
<i>H</i>	9	<i>no</i>
<i>I</i>	3	<i>no</i>
<i>J</i>	6	<i>no</i>
<i>EH</i>	4	<i>no</i>
<i>EJ</i>	3	<i>no</i>
<i>FH</i>	8	<i>no</i>
<i>HJ</i>	5	<i>no</i>
<i>FHJ</i>	4	<i>no</i>

From Table 1, there is no frequent itemset in TD when $minsupp = 0.5$.

If each database is large and the number of databases is greater than six, then the number of frequent itemsets from each database may be so large that browsing them and finding interesting association rules can be rather difficult. Therefore, it is hard to identify which of the frequent itemsets from individual databases are useful.

The above six databases apparently belong to two different classes of applications. The first class of application is relevant to items A, B, C , and D , and the second class is relevant to items E, F, G, H, I , and J . If we classify the six databases into two classes as follows: $TD_1 = \{D_1, D_2, D_3\}$ and $TD_2 = \{D_4, D_5, D_6\}$, and mine TD_1 and TD_2 separately, we would get some interesting results.

Table 2 *Frequent Itemsets from Each Database in TD₁*

<i>Itemsets</i>	<i>Frequency</i>	$\geq minsupp$
<i>A</i>	9	<i>yes</i>
<i>B</i>	10	<i>yes</i>
<i>C</i>	10	<i>yes</i>
<i>AB</i>	6	<i>no</i>
<i>AC</i>	6	<i>no</i>
<i>BC</i>	8	<i>yes</i>
<i>ABC</i>	4	<i>no</i>

From Table 2, when $minsupp = 0.5$, A, B, C , and BC are frequent itemsets.

Table 3 *Frequent Itemsets from Each Database in TD₂*

<i>Itemsets</i>	<i>Frequency</i>	$\geq \text{minsupp}$
<i>E</i>	6	<i>yes</i>
<i>F</i>	8	<i>yes</i>
<i>H</i>	9	<i>yes</i>
<i>I</i>	3	<i>no</i>
<i>J</i>	6	<i>yes</i>
<i>EH</i>	4	<i>no</i>
<i>EJ</i>	3	<i>no</i>
<i>FH</i>	8	<i>yes</i>
<i>HJ</i>	5	<i>no</i>
<i>FHJ</i>	4	<i>no</i>

From Table 3, when $\text{minsupp} = 0.5$, *E*, *F*, *H*, *J* and *FH* are frequent itemsets.

The above observations motivate us to design an efficient database classification for multi-database mining. We will attack two key problems in database classification: (i) how to effectively measure the relevance of databases independent applications, and (ii) how to effectively search for the best classification.

The rest of this paper is organized as follows. In Section 2, we present an approach for classifying databases by similarity. Section 3 designs an algorithm that searches for the best classification. Section 4 evaluates the proposed approach by experiments, and we review related work in Section 5.

2 Classifying Multiple Databases

In this section, we present a classification scheme by similarity between different databases, and also construct a criterion for evaluating different classifications.

2.1 Features in Databases

To classify multiple databases, we need to check what is used to evaluate the relevance between different databases. Indexing databases by features is a common technique. Therefore, we can measure the relevance between two databases by comparing their features.

There are different types of databases, such as relational databases, transaction databases, and text databases. This paper focuses on transaction databases, and all items in a transaction database are used as features to index the database. If two databases share a significant number of common items, the two databases are relevant to each other. Classification will be based on the items in each database. If a large transaction database has been mined for association rules, we can also use the discovered rules or all items that occur in these rules to represent the database.

Selecting features in a database is not an easy task. Because databases in real-world applications are often very large, it is costly to search all features in multiple databases. In this case, we can use sampling techniques to sample each large database.

2.2 Similarity Measures

Let D_1, D_2, \dots, D_m be m databases from different branches of a large company, $Items(D_i)$ the set of items in D_i ($i = 1, 2, \dots, m$), and S_i the set of association rules from D_i ($i = 1, 2, \dots, m$). If we do not have any other information about these databases, the items from these databases can be used to measure the closeness of each pair of database objects. We denote this measure by sim_1 .

Definition 1 *The similarity sim_1 between the items of two databases D_i and D_j is defined as follows.*

$$\text{sim}_1(D_i, D_j) = \frac{|Items(D_i) \cap Items(D_j)|}{|Items(D_i) \cup Items(D_j)|}$$

where “ \cap ” denotes set intersection, “ \cup ” denotes set union, and “ $|Items(D_i) \cap Items(D_j)|$ ” is the number of elements in set $Items(D_i) \cap Items(D_j)$.

In the above definition of function sim_1 , we take the size of the intersection of a pair of database objects to measure the closeness of the two databases. A large intersection corresponds to a high degree of similarity, whereas two database objects with a small intersection are considered to be rather dissimilar.

We now illustrate the use of the above similarity by an example.

Example 1 Let $Items(D_1) = \{a_1, a_2, a_3\}$ and $Items(D_2) = \{a_2, a_3, b_1, b_2\}$ be two sets of items of two databases D_1 and D_2 , respectively. The similarity between D_1 and D_2 is as follows.

$$sim_1(D_1, D_2) = \frac{|Items(D_1) \cap Items(D_2)|}{|Items(D_1) \cup Items(D_2)|} = \frac{2}{5} = 0.4.$$

If $sim_1(D_i, D_j) = 1$, it means that $Items(D_i) = Items(D_j)$ or, D_i and D_j can completely belong to the same class under measure sim_1 . It does not mean $D_i = D_j$ when $sim_1(D_i, D_j) = 1$.

The sim_1 measure is useful to estimate the similarity between two large transaction databases because getting more information such as the amount of an item purchased by customers from the databases is often expensive.

In the meanwhile, the similarity between two databases D_i and D_j can be approximated by the intersection of their two interesting items, where, for a database D_k , the interesting items are the items that occur in the rule set S_k from D_k ($k = i, j$).

Definition 2 We construct similarity sim_2 by the interesting items in S_1, S_2, \dots, S_m as follows.

$$sim_2(D_i, D_j) = \frac{|Items(S_i) \cap Items(S_j)|}{|Items(S_i) \cup Items(S_j)|}$$

where “ $Items(S_k)$ ” stands for the set of all items that occur in the association rules from S_k .

Let $S = \{S_1, S_2, \dots, S_m\}$. Then function $sim_2 : Items(S) \times Items(S) \rightarrow [0, 1]$. We take the size of the intersection of the interesting items of a pair of database objects to measure the closeness of the two databases. We illustrate the use of sim_2 by an example below.

Example 2 Let $S_1 = \{i_1 \wedge i_3 \rightarrow i_2, i_2 \rightarrow i_4, i_3 \rightarrow i_5\}$ and $S_2 = \{i_1 \wedge i_3 \rightarrow i_4, i_3 \rightarrow i_5, i_6 \rightarrow i_7\}$ be two sets of association rules from two databases D_1 and D_2 , respectively. Then $Items(S_1) = \{i_1, i_2, i_3, i_4, i_5\}$ and $Items(S_2) = \{i_1, i_3, i_4, i_5, i_6, i_7\}$ are the two sets of interesting items of D_1 and D_2 , respectively. The similarity between D_1 and D_2 on the interesting items is as follows.

$$sim_2(D_1, D_2) = \frac{|Items(S_1) \cap Items(S_2)|}{|Items(S_1) \cup Items(S_2)|} = \frac{4}{7} = 0.556.$$

In the above example, if $Items(D_1) = Items(D_2) = \{i_1, i_2, \dots, i_8\}$, then $sim_1(D_1, D_2) = 1$ although $sim_2(D_1, D_2) = 0.556$. In fact, sim_2 only approximates the degree of the similarity between these items.

We have defined two functions for measuring the similarity of a pair of database objects. The two measure functions have some properties as follows.

Property 1 $sim_k(D_i, D_j)$ ($k = 1, 2$) satisfy that

- (1) $0 \leq sim_k(D_i, D_j) \leq 1$;
- (2) $sim_k(D_i, D_j) = sim_k(D_j, D_i)$;
- (3) $sim_k(D_i, D_i) = 1$.

Proof: For (1), because

$$\begin{aligned} 0 &= |\emptyset| \leq |Items(D_i) \cap Items(D_j)| \leq |Items(D_i) \cup Items(D_j)|, \\ 0 &\leq |Items(D_i) \cap Items(D_j)| / |Items(D_i) \cap Items(D_j)| \leq 1, \end{aligned}$$

and

$$0 \leq \text{sim}_1(D_i, D_j) \leq 1.$$

Also, we have

$$0 \leq \text{sim}_2(D_i, D_j) \leq 1.$$

(2) and (3) can be obtained directly from their definitions. □

2.3 Relevance of Databases and Database Classification

Definition 3 A database D_i is α -relevant to D_j under measure sim_1 if $\text{sim}_1(D_i, D_j) > \alpha$, where $\alpha (> 0)$ is a given threshold.

Let $\alpha = 0.4$. Consider the data in Example 2. Because $\text{sim}_1(D_1, D_2) = 0.556 > \alpha = 0.4$, the database D_1 is 0.4-relevant to D_2 .

Definition 4 Let D be a set of m databases D_1, D_2, \dots, D_m . The similarity of database D_i under measure sim_1 , denoted as $D.\text{similarity}(D_i, \text{sim}_1, \alpha)$, returns the following databases:

$$D.\text{similarity}(D_i, \text{sim}_1, \alpha) = \{d \in D \mid d \text{ is } \alpha\text{-relevant to } D_i\}.$$

Consider the six databases in Section 1. Let D be the set of 6 databases D_1, D_2, \dots, D_6 . Then we have,

$$D.\text{similarity}(D_1, \text{sim}_1, 0.5) = \{D_2, D_3\}$$

$$D.\text{similarity}(D_2, \text{sim}_1, 0.5) = \{D_1, D_3\}$$

$$D.\text{similarity}(D_3, \text{sim}_1, 0.5) = \{D_1, D_2\}$$

$$D.\text{similarity}(D_4, \text{sim}_1, 0.5) = \{D_5, D_6\}$$

$$D.\text{similarity}(D_5, \text{sim}_1, 0.5) = \{D_4, D_6\}$$

$$D.\text{similarity}(D_6, \text{sim}_1, 0.5) = \{D_4, D_5\}$$

Our similarity in this example provides a good classification of the six databases independent of specific applications.

Definition 5 Let D be a set of m databases D_1, D_2, \dots, D_m . A class class^α of D under measure sim_1 is defined as

$$\text{class}^\alpha = \{d \in D \mid \forall d' \in \text{class}^\alpha (d \text{ is } \alpha\text{-relevant to } d')\}.$$

The above definition of a class specifies that any two of the database objects in the class are α -relevant to each other.

Definition 6 Let D be a set of m databases D_1, D_2, \dots, D_m . $\text{class}(D, \text{sim}, \alpha) = \{\text{class}_1^\alpha, \text{class}_2^\alpha, \dots, \text{class}_n^\alpha\}$ is a classification of D_1, D_2, \dots, D_m under measure sim_1 if

$$(1) \text{class}_1^\alpha \cup \text{class}_2^\alpha \cup \dots \cup \text{class}_n^\alpha = D;$$

$$(2) \text{ for any two databases } D_i \text{ and } D_j \text{ in } \text{class}_k, \text{sim}(D_i, D_j) \geq \alpha;$$

This definition allows a database to belong to more than one class. We now show a classification for multiple databases by an example below.

Example 3 Suppose S_1, S_2, \dots, S_{10} are the sets of association rules from 10 databases

$$D_1, D_2, \dots, D_{10},$$

respectively, where $Items(S_i) = \{a_1, a_2, a_3, a_4, b_i\}$ for $i = 1, 2, 3, 4$, $Items(S_{4+j}) = \{b_1, b_2, b_3, b_4, c_j\}$ for $j = 1, 2, 3, 4$, $Items(S_{8+k}) = \{c_1, c_2, c_3, c_4, d_k\}$ for $k = 1, 2$, and $\alpha = 0.4$. Then

$$class_1^{0.4} = \{D_1, D_2, D_3, D_4\},$$

$$class_2^{0.4} = \{D_5, D_6, D_7, D_8\},$$

$$class_3^{0.4} = \{D_9, D_{10}\}$$

are a classification of D_1, D_2, \dots, D_{10} .

2.4 Complete Classifications and Their Goodness

By Definition 6, we may get different classifications with different α values. Generally speaking, a good classification is determined by the structures and distributions of data in the given databases.

Definition 7 Let $class(D, sim_1, \alpha) = \{class_1^\alpha, class_2^\alpha, \dots, class_n^\alpha\}$ be a classification of m databases D_1, D_2, \dots, D_m under measure sim_1 . The classification is complete if

- (1) for any two classes $class_i^\alpha$ and $class_j^\alpha$, $class_i^\alpha \cap class_j^\alpha = \emptyset$ for $i \neq j$;
- (2) for any two databases D_i in $class_l^\alpha$ and D_j in $class_h^\alpha$, if $l \neq h$, $sim_1(D_i, D_j) \leq \alpha$.

Clearly, when $\alpha = 1$, we can obtain a complete classification for a set of given databases. Also, when $\alpha = 0$, we can obtain another complete classification for a set of given databases. These two classifications are called trivial classifications. In real-world applications, multiple databases from the same organization can be divided into classes (such as business types) that are non-trivial classifications, but in practice, we do not always have such non-trivial classifications on the available databases. We illustrate this argument by an example below.

Example 4 Let $Items_1 = \{1, 2, 3\}$, $Items_2 = \{2, 3, 4\}$, $Items_3 = \{3, 4, 5\}$ and $Items_4 = \{4, 5, 6\}$ be four sets of items from four data sources D_1, D_2, D_3 , and D_4 , respectively. Then,

$$sim_1(D_1, D_2) = \frac{|Items_1 \cap Items_2|}{|Items_1 \cup Items_2|} = \frac{2}{4} = 0.5,$$

$$sim_1(D_1, D_3) = \frac{|Items_1 \cap Items_3|}{|Items_1 \cup Items_3|} = \frac{1}{5} = 0.2,$$

$$sim_1(D_1, D_4) = \frac{|Items_1 \cap Items_4|}{|Items_1 \cup Items_4|} = \frac{0}{6} = 0.$$

$$sim_1(D_2, D_3) = \frac{|Items_2 \cap Items_3|}{|Items_2 \cup Items_3|} = \frac{2}{4} = 0.5,$$

$$sim_1(D_2, D_4) = \frac{|Items_2 \cap Items_4|}{|Items_2 \cup Items_4|} = \frac{1}{5} = 0.2,$$

$$sim_1(D_3, D_4) = \frac{|Items_3 \cap Items_4|}{|Items_3 \cup Items_4|} = \frac{2}{4} = 0.5$$

Therefore, there is no non-trivial complete classification for the four databases.

From the above definitions, we can get the following theorems.

Theorem 1 *If a classification $class(D, sim_1, \alpha) = \{class_1^\alpha, class_2^\alpha, \dots, class_n^\alpha\}$ of m databases D_1, D_2, \dots, D_m under measure sim_1 is complete, $sim_1(D_i, D_j) \geq \alpha$ and $sim_1(D_j, D_l) \geq \alpha$, then*

$$sim_1(D_i, D_l) \geq \alpha.$$

Proof: By Definition 7, D_i and D_j belong to a class $class_i^\alpha$ and, D_j and D_k belong to a class $class_k^\alpha$.

Since the classification $class(D, sim_1, \alpha)$ is complete, if $l \neq k$, then $class_i^\alpha \cap class_k^\alpha = \emptyset$. Given $D_j \in class_i^\alpha$ and $D_j \in class_k^\alpha$ we have $class_i^\alpha \cap class_k^\alpha \neq \emptyset$, and $l = k$. Therefore,

$$sim_1(D_i, D_l) \geq \alpha.$$

□

Theorem 2 *If a classification $class(D, sim_1, \alpha) = \{class_1^\alpha, class_2^\alpha, \dots, class_n^\alpha\}$ of m databases D_1, D_2, \dots, D_m under measure sim_1 is complete, then for any two databases D_i and D_j , $sim_1(D_i, D_j) \geq \alpha$ if and only if there is a $class_k$ such that $class_k$ contains both D_i and D_j .*

Proof: (1) “ \Rightarrow ”: if $sim(D_i, D_j) \geq \alpha$, then there is a $class_k$ such that $class_k$ contains both D_i and D_j .”

This can be confirmed by the definitions of similarity and complete classification.

(2) “ \Leftarrow ”: if a $class_k$ contains both D_i and D_j , then $sim_1(D_i, D_j) \geq \alpha$.”

This can be observed from the definition of complete classification.

□

To evaluate a complete classification, we define a goodness measure, *Goodness*, below. The goodness of a complete classification is estimated by the sum of distances between each pair of databases in the classes of the classification.

Definition 8 *Let $class^\alpha$ be a class of m given databases. The sum of the distances of each pair of the databases in $class^\alpha$ under measure sim_1 is defined as follows:*

$$Value(class^\alpha) = \sum_{\substack{i \neq j \\ D_i, D_j \in class^\alpha}} (1 - sim_1(D_i, D_j))$$

$1 - sim_1(D_i, D_j)$ here is the distance between two databases D_i and D_j (because traditional similarity between database objects is expressed by a distance function such that a low distance corresponds to a high degree of similarity, whereas two database objects with a large distance are considered to be rather dissimilar).

Property 2 *For a class $class^\alpha$ of given multiple databases, we have*

$$0 \leq Value(class^\alpha) \leq |class^\alpha|^2 - |class^\alpha|$$

where $|class^\alpha|$ denotes the number of elements in $class^\alpha$ under measure sim_1 .

Proof: Because $0 \leq sim_1(D_i, D_j) \leq 1$, $0 \leq 1 - sim_1(D_i, D_j) \leq 1$, and $Value(class^\alpha) \geq 0$.

There are $|class^\alpha|^2$ distance values for $class^\alpha$. The distances between D_i and D_j are 0 when $i = j$ and the other $|class^\alpha|^2 - |class^\alpha|$ distances may not be 0. Hence, the $|class^\alpha|^2 - |class^\alpha|$ distances need to be added up. Because $1 - sim_1(D_i, D_j) \leq 1$, we have,

$$Value(class^\alpha) \leq |class^\alpha|^2 - |class^\alpha|$$

□

Using all sums of distances between each pair of databases, we can define the goodness of a classification as follows.

Definition 9 Let $class(D, sim, \alpha) = \{class_1^\alpha, class_2^\alpha, \dots, class_n^\alpha\}$ be a classification of m databases D_1, D_2, \dots, D_m under measure sim_1 . The goodness of the class classification is defined as follows.

$$Goodness(class, \alpha) = \sum_{i=1}^n Value(class_i^\alpha).$$

Property 3 Let $Goodness(class, \alpha)$ be the goodness of a complete classification $class(D, sim, \alpha)$, which is $\{class_1^\alpha, class_2^\alpha, \dots, class_n^\alpha\}$ for m databases under measure sim_1 . Then

$$0 \leq Goodness(class, \alpha) \leq m^2 - m$$

Proof: Because $Value(class^\alpha) \geq 0$, $Goodness(class, \alpha) \geq 0$. According to Property 2,

$$\begin{aligned} Goodness(class, \alpha) &\leq (|class_1^\alpha|^2 - |class_1^\alpha|) + (|class_2^\alpha|^2 - |class_2^\alpha|) \\ &+ \dots + (|class_n^\alpha|^2 - |class_n^\alpha|) \\ &= |class_1^\alpha|^2 + |class_2^\alpha|^2 + \dots + |class_n^\alpha|^2 \\ &- (|class_1^\alpha| + |class_2^\alpha| + \dots + |class_n^\alpha|) \end{aligned}$$

Also, because $class(D, sim_1, \alpha)$ is a complete classification,

$$m = |class_1^\alpha| + |class_2^\alpha| + \dots + |class_n^\alpha|.$$

On the other hand,

$$m^2 \geq |class_1^\alpha|^2 + |class_2^\alpha|^2 + \dots + |class_n^\alpha|^2.$$

Therefore,

$$\begin{aligned} m^2 - m &\geq |class_1^\alpha|^2 + |class_2^\alpha|^2 + \dots + |class_n^\alpha|^2 \\ &- (|class_1^\alpha| + |class_2^\alpha| + \dots + |class_n^\alpha|) \end{aligned}$$

and,

$$Goodness(class, \alpha) \leq m^2 - m.$$

□

The number ($|class|$) of elements in a classification $class$ is also related to α . To demonstrate the relationships among $|class|$, $Goodness$, and α , we use an example below.

Example 5 Consider six databases D_1, D_2, \dots, D_6 , where $Items(D_1) = \{A, B, C, D, E\}$, $Items(D_2) = \{A, B, C\}$, $Items(D_3) = \{A, B, D\}$, $Items(D_4) = \{F, G, H, I, J\}$, $Items(D_5) = \{F, G, H\}$, and $Items(D_6) = \{F, G, I\}$.

We use similarity function sim_1 to show the changes of the number of classes and the Goodness when α is changed. The similarity between each pair of the six databases is illustrated in the following table.

Table 4 The similarity between six databases

sim_1	D_1	D_2	D_3	D_4	D_5	D_6
D_1	1	0.6	0.6	0	0	0
D_2	0.6	1	0.5	0	0	0
D_3	0.6	0.5	1	0	0	0
D_4	0	0	0	1	0.6	0.6
D_5	0	0	0	0.6	1	0.5
D_6	0	0	0	0.6	0.5	1

(1) When $\alpha = 0$, $|class| = 1$ with

$$class = \{\{D_1, D_2, \dots, D_6\}\}$$

$$Goodness(class, 0) = 2(4(1 - 0.6) + 2(1 - 0.5) + 9) = 23.2$$

(2) When $0 < \alpha \leq 0.5$, $|class| = 2$ with

$$class = \{\{D_1, D_2, D_3\}; \{D_4, D_5, D_6\}\}$$

$$Goodness(class, 0) = 4(2(1 - 0.6) + (1 - 0.5)) = 5.2$$

(3) When $0.5 < \alpha \leq 0.6$, there is no non-trivial complete classification. This interval is different from $0.6 < \alpha \leq 1$.

(4) When $0.6 < \alpha \leq 1$, $|class| = 6$ with

$$class = \{\{D_1\}; \{D_2\}; \{D_3\}; \{D_4\}; \{D_5\}; \{D_6\}\}$$

$$Goodness(class, 0) = 6(1 - 1) = 0$$

This example illustrates the fact that we can obtain a good classification when the distance between n and $Goodness$ gets the smallest value for $\alpha \in [0, 1]$. We now define a function to judge the goodness of a classification.

Definition 10 Let $class(D, sim, \alpha) = \{class_1^\alpha, class_2^\alpha, \dots, class_n^\alpha\}$ be a classification of m databases D_1, D_2, \dots, D_m under measure sim_1 . The absolute of the difference between the $Goodness$ and $|class|$, called $distance_{Goodness}^f$ for $\alpha \in [0, 1]$, is defined as follows,

$$distance_{Goodness}^f(class, \alpha) = |Goodness(class, \alpha) - |class||.$$

For any given $|class|$ and $Goodness$ with respect to a different α , we can obtain a different $distance_{Goodness}^f$ from a complete classification of the given multiple databases. We now illustrate the use of $distance_{Goodness}^f$ by an example below.

Example 6 Consider the six databases in Example 5.

(1) When $\alpha = 0$,

$$class = \{\{D_1, D_2, \dots, D_6\}\},$$

$$distance_{Goodness}^f(class, 0) = |23.2 - 1| = 22.2.$$

(2) When $0 < \alpha \leq 0.5$,

$$class = \{\{D_1, D_2, D_3\}; \{D_4, D_5, D_6\}\};$$

$$distance_{Goodness}^f(class, \alpha) = |5.2 - 2| = 3.2.$$

(3) When $0.5 < \alpha \leq 0.6$, there is no non-trivial complete classification.

(4) When $0.6 < \alpha \leq 1$,

$$class = \{\{D_1\}; \{D_2\}; \{D_3\}; \{D_4\}; \{D_5\}; \{D_6\}\},$$

$$distance_{Goodness}^f(class, \alpha) = |0 - 6| = 6.$$

$distance_{Goodness}^f$ has a polar-point for a set of given databases. In this example, $distance_{Goodness}^f = 3.2$ is the minimal value for $\alpha \in (0, 0.5]$ which corresponds to the best classification. We will search for the best classification for given multiple databases under $distance_{Goodness}^f$ in Section 3.

3 Searching for the Best Classification

To search for the best classification from the given multiple databases, we propose a two-step approach in this section. The first step designs a procedure that generates a classification for a given α . The second step develops an algorithm that searches for the best classification by $distance_{Goodness}^f$.

3.1 The First Step: Generate A Classification

For m given databases D_1, D_2, \dots, D_m , we can use a measure of similarity such as sim_1 to build a relation table of similarity between the databases as follows.

Table 5 *The relation table of similarity*

sim_1	D_1	D_2	\dots	D_m
D_1	$sim_1(D_1, D_1)$	$sim_1(D_1, D_2)$	\dots	$sim_1(D_1, D_m)$
D_2	$sim_1(D_2, D_1)$	$sim_1(D_2, D_2)$	\dots	$sim_1(D_2, D_m)$
\vdots	\vdots	\vdots	\vdots	\vdots
D_m	$sim_1(D_m, D_1)$	$sim_1(D_m, D_2)$	\dots	$sim_1(D_m, D_m)$

When α is assigned a particular value, we can generate a corresponding classification *class* for the databases according to the relation table of similarity. The procedure *GreedyClass* for generating a classification is designed in Procedure 1.

Procedure 1 *GreedyClass*

begin

Input: D_i ($1 \leq i \leq m$): databases, α : threshold value;

Output: $Class^\alpha$: set of classes consisting of α -relevant databases;

(1) **construct** the relation table of similarity by sim_1 ;

(2) **let** $Class^\alpha \leftarrow \{\}$;

(3) **for** $i := 1$ to m **do**

begin

(3.1) **let** $a \leftarrow false$;

(3.2) **for** any class c in $Class^\alpha$ **do**

begin

let $b \leftarrow true$;

if $\neg a$ **then**

for any database d in c **do**

if b and $(sim_1(D_i, d) < \alpha)$ **then**

let $b \leftarrow false$;

if b **then**

begin

let $c \leftarrow c \cup \{D_i\}$;

let $a \leftarrow true$;

end

end

(3.3) **if** $\neg a$ **then**

begin

create a new class $\{D_i\}$;

let $Class^\alpha \leftarrow Class^\alpha \cup \{D_i\}$;

end

end

(4) **output** the classification $Class^\alpha$;

end;

Procedure *GreedyClass* generates a classification $Class^\alpha$ for m given databases under similarity sim_1 when the threshold α is assigned. Steps (1) and (2) perform initialization.

Step (3) consists of three sub-steps, and places database D_i in a class. Step (3.1) designs a logical variable a and assigns it *false* to indicate that D_i does not yet belong to any class. Step (3.2) checks if D_i can be placed in an existing class c . If $sim_1(D_i, d) \geq \alpha$ for any database d in c , D_i is placed in class c , and a is assigned *true* to indicate that D_i has been placed in a class of $Class^\alpha$. Step (3.3) generates a class for D_i if D_i is not placed in any existing class of $Class^\alpha$, and the new class is appended to $Class^\alpha$.

Step (4) outputs the classification $Class^\alpha$.

We now illustrate the performance of the above procedure by Example 5.

Example 7 *The relation table of similarity between the six databases is constructed in Table 4. Let $\alpha = 0.4$.*

When $i = 1$ in the loop of Step (3), there is no class in $Class^{0.4}$. So, class $\{D_1\}$ is generated and appended to $Class^{0.4}$. When $i = 2$, it checks if D_2 belongs to any class in $Class^{0.4}$. Because $sim_1(D_2, D_1) > 0.4$ for the only class with the only element D_1 , D_2 is placed in the same class. When $i = 3$, it checks if D_3 belongs to any existing class in $Class^{0.4}$. Because $sim_1(D_3, D_1) > 0.4$ and $sim_1(D_3, D_2) > 0.4$, D_3 is also placed in the same class.

When $i = 4$, because $sim_1(D_4, D_1) < 0.4$, D_4 cannot be placed in the class. A new class $\{D_4\}$ is generated and appended to $Class^{0.4}$.

When $i = 5$, because $sim_1(D_5, D_1) < 0.4$, D_5 cannot be placed in the class, and because $sim_1(D_5, D_4) > 0.4$, D_5 is placed in the second class. Similarly, when $i = 6$, D_6 is placed in the second class, and the loop in Step (3) is stopped.

$$Class^{0.4} = \{\{D_1, D_2, D_3\}; \{D_4, D_5, D_6\}\}$$

is output in Step (4).

The above procedure has generated a complete classification $Class^{0.4}$ for $\alpha = 0.4$. Similarly, when $\alpha = 0.52$, the procedure will generate a classification

$$Class^{0.52} = \{\{D_1, D_2\}; \{D_3\}; \{D_4, D_5\}; \{D_6\}\},$$

which is not a complete classification.

Theorem 3 *Procedure GreedyClass works correctly.*

Proof: From Steps (3) and (4), a classification of databases is generated and output for a given α . We need to show that all given databases are contained by classes of the classification $Class^\alpha$, and each database belongs to only one of the classes.

For any unclassified database D_i ($1 \leq i \leq m$), if D_i is α -relevant to all databases in a class, it is added to the class in Step (3.2); or else a new class is generated for D_i in Step (3.3). This means that all given databases are contained by classes of the classification $Class^\alpha$.

On the other hand, in the loop in Step (3), each database is handled once and the database is classified into either an existing class (in Step (3.2)), or a new class (in Step (3.3)). This means that each database belongs to only one of the classes.

□

In Procedure *GreedyClass*, m databases are required to be input for simplicity. In fact, it only requires the sets of features (items) of the databases. (For large databases, we can use the sampling techniques in [?] to obtain items.) Hence, we need

$$|Items(D_1)| + |Items(D_2)| + \cdots + |Items(D_m)|$$

units to save all items, which is less than, or equal to, n , where n is the maximum among $|Items(D_1)|$, $|Items(D_2)|$, \dots , $|Items(D_m)|$. For Step (1) of *GreedyClass*, we need m^2 units to save the similarities. Consequently, the space complexity of *GreedyClass* is $O(n^2 + m^2) = O(n^2)$ for $n \geq m$ ($O(m^2)$ if $n < m$).

Obviously, the time complexity of *GreedyClass* is dominated by the loop in Steps (1) and (3). Therefore, we have the following theorem.

Theorem 4 *The time complexity of GreedyClass is $O(n^2m^2 + m^4)$, where n is the maximum among $|Items(D_1)|$, $|Items(D_2)|$, \dots , $|Items(D_m)|$, and m is the number of given databases.*

Proof: For Step (1), there are $m^2/2$ similarities that need to be computed according to sim_1 and Property 3.1. Each similarity needs, at most, n^2 comparisons. Therefore, the time complexity of Step (1) is $O(n^2m^2)$.

For Step (3), there are m databases that need to be classified. For each database, Step (3.2) dictates the complexity by a twice-loop. No matter how the databases are distributed, at most $i - 1$ comparisons are needed to classify the i th database, although Step (3.2) is a twice-loop. Each comparison is used to access the $m \times m$ relation table of similarity. Hence, the time complexity of Step (3) is $O(m^4)$.

Accordingly, the time complexity of *GreedyClass* is $O(n^2m^2 + m^4)$. □

Note that Step (1) is included in *GreedyClass* for the purpose of description. Actually, Step (1) is taken as a procedure in applications, and is only called once for generating the relation table of similarity. This means that *the time complexity of GreedyClass is really $O(m^4)$* , by taking away Step (1).

As we have seen, database classification is time-consuming. It is obvious that, when m (the number of databases) is very large, the time complexity becomes severe. To face this problem, there are many mature techniques available in Information Retrieval and Pattern Analysis. A commonly used method is firstly to divide a large set of databases into several smaller sets and then merge classes from the smaller sets. Thus, *GreedyClass* can be optimized by resizing in Information Retrieval and Pattern Analysis when m is large.

3.2 The Second Step: Search for the Best Classification

$|class| = n = f(\alpha)$ is an increasing function for complete classifications. That is, for $\alpha_1 < \alpha_2$ in $[0, 1]$, we have $f(\alpha_1) \leq f(\alpha_2)$. $Goodness(class, \alpha)$ is a decreasing function for complete classifications. That is, for $\alpha_1 < \alpha_2$ in $[0, 1]$, we have,

$$Goodness(class, \alpha_1) \geq Goodness(class, \alpha_2).$$

However, the absolute of the difference between the *Goodness* and $|class|$

$$distance_{Goodness}^f(class, \alpha) = |Goodness(class, \alpha) - f(\alpha)|$$

is a function for complete classifications, and is with an unique polar-point: minimum $distance_{Goodness}^f(class, \alpha_0)$. That is, for $\alpha_1 \in [0, 1]$, we have,

$$distance_{Goodness}^f(class, \alpha_0) \leq distance_{Goodness}^f(class, \alpha_1).$$

Also, $distance_{Goodness}^f$ is a jumping function. There is a neighborhood of α_0 : $[a, b] \subset [0, 1]$ such that

$$distance_{Goodness}^f(class, \alpha_1) = distance_{Goodness}^f(class, \alpha_0)$$

for $\forall \alpha_1 \in [a, b]$.

Accordingly, $distance_{Goodness}^f$ is a decreasing function for α in $[0, a]$. That is, for $\alpha_1 < \alpha_2$ in $[0, a]$, we have,

$$distance_{Goodness}^f(class, \alpha_1) \geq distance_{Goodness}^f(class, \alpha_2)$$

and $distance_{Goodness}^f$ is an increasing function in $[b, 1]$. That is, for $\alpha_1 < \alpha_2$ in $[b, 1]$, we have,

$$distance_{Goodness}^f(class, \alpha_1) \leq distance_{Goodness}^f(class, \alpha_2).$$

By the above properties of $distance_{Goodness}^f(class, \alpha)$ for complete classifications, we design an algorithm below to search for the best classification from m given databases.

Algorithm 1 *BestClassification*

begin

Input: D_i ($1 \leq i \leq m$): databases; λ : the size of step of α ;

Output: *Class*: set of classes consisting of α -relevant databases;

- (1) **let** $\alpha_1 \leftarrow 1$; $t \leftarrow 0$;
- (2) **call** procedure *GreedyClass* for α_1 ;
let $x_1 \leftarrow temp \leftarrow distance_{Goodness}^f(Class, \alpha_1)$;
- (3) **let** $\alpha_2 \leftarrow \alpha_1 - \lambda$;
- (4) **call** the procedure *GreedyClass* for α_2 ;
let $x_2 \leftarrow distance_{Goodness}^f(Class, \alpha_2)$;
if $Class^{\alpha_2}$ is not a complete classification **then**
begin
let $\alpha_2 \leftarrow$ a proper value;
go to Step (4);
end
- (5) **if** $x_1 \geq x_2$ **then**
begin
let $\alpha_1 \leftarrow \alpha_2$;
let $x_1 \leftarrow x_2$;
let $\alpha_2 \leftarrow \alpha_1 - \lambda$;
go to Step (4);
end
- (6) **else**
begin
 - (6.1) **let** $\alpha_3 \leftarrow (\alpha_1 + \alpha_2)/2$;
 - (6.2) **call** the procedure *GreedyClass* for α_3 ;
if $Class^{\alpha_3}$ is a complete classification **then**
let $x_3 \leftarrow distance_{Goodness}^f(Class, \alpha_3)$;
else if $Class^{\alpha_2}$ is not a complete classification **then**
begin
let $\alpha_3 \leftarrow$ a proper value in (α_2, α_1) ;
go to Step (6.2);
end
 - (6.3) **if** $(x_1 > x_3)$ and $(x_2 > x_3)$ **then**
begin
let $\alpha_1 \leftarrow \alpha_3$;
let $x_1 \leftarrow x_3$;
go to Step (5.1);
end
 - (6.4) **else if** $(x_1 \leq x_3)$ and $(x_2 \geq x_3)$ **then**
begin
let $\alpha_2 \leftarrow \alpha_3$;
let $x_2 \leftarrow x_3$;
go to Step (6.1);
end

```

(6.5) else if ( $x_1 = x_3$ ) and ( $x_2 = x_3$ ) then
      begin
        let  $class \leftarrow Class^{\alpha_3}$ ;
        go to Step (7);
      end
(6.6) else
      begin
        let  $class \leftarrow x_1$ ;
        go to Step (7);
      end
end
(7) output the classification  $class$ ;

end;

```

Algorithm *BestClassification* searches for the best classification from possible classifications of m given databases. Step (1) performs initialization. The search begins with a call to the procedure *GreedyClass* for $\alpha_1 = 1$ in Step (2). Step (3) selects the next threshold α_2 by α_1 and the step size λ .

Step (4) calls the procedure *GreedyClass* for α_2 . If the classification $Class^{\alpha_2}$ is not a complete classification, we need to select a proper value in $[0, 1]$ for α_2 and repeat Step (4). The “proper value” means to assign a new value to α_2 .

If $Class^{\alpha_2}$ is a complete classification, we compare

$$x_1 = distance_{Goodness}^f(Class, \alpha_1)$$

and

$$x_2 = distance_{Goodness}^f(Class, \alpha_2)$$

in Step (5). If $x_1 \geq x_2$, we take α_2 as α_1 and go to Step (4).

If $x_1 < x_2$, it means that the best classification can be found in $[\alpha_1, \alpha_2)$, and the search in $[\alpha_1, \alpha_2)$ is done in Step (6). For example, the middle point $(\alpha_1 + \alpha_2)/2$ can be taken as a new threshold α_3 to check which half of $[\alpha_1, \alpha_2)$ contains the best classification.

Step (7) outputs the best classification $class$.

Example 8 We now illustrate the performance of the above algorithm with Example 5. Let $\lambda = 0.05$.

Firstly, $\alpha_1 = 1$ and $x_1 = distance_{Goodness}^f(Class, 1) = 6$. Because $distance_{Goodness}^f(Class, \alpha) = 6$ for the points: 0.95, 0.8, \dots , 0.6, we get $\alpha_1 = 0.6$ and $x_1 = distance_{Goodness}^f(Class, 0.6) = 6$ after repeating Step (4) and (5).

When $\alpha_2 = \alpha_1 - 0.05 = 0.55$, $Class^{0.55}$ is not a complete classification in Step (4). We need to select a proper value for α_2 . For example, $\alpha_2 = \alpha_2 - 0.05 = 0.5$, and we get $x_2 = distance_{Goodness}^f(Class, 0.5) = 3.2$. Because $x_1 > x_2$, we get $\alpha_1 = 0.5$ and $x_1 = distance_{Goodness}^f(Class, 0.5) = 3.2$. Steps (4) and (5) are repeated until $\alpha_1 = 0.05$ and $x_1 = distance_{Goodness}^f(Class, 0.05) = 3.2$.

Now, $\alpha_2 = \alpha_1 - 0.05 = 0$ and $x_2 = distance_{Goodness}^f(Class, 0) = 22.2$. Because $x_1 < x_2$, we find that the best classification can be found when α is in $(0, 0.05]$. We take $\alpha_3 = (\alpha_1 + \alpha_2)/2 = 0.025$ and $x_3 = distance_{Goodness}^f(Class, 0.025) = 3.2$.

Because the condition of Step (6.4) is satisfied, $\alpha_2 = \alpha_3 = 0.025$ and $x_2 = distance_{Goodness}^f(Class, 0.025) = 3.2$. This means that we find that the best classification can be found when α is in $(0.025, 0.05]$. We take $\alpha_3 = (\alpha_1 + \alpha_2)/2 = 0.0375$ and $x_3 = distance_{Goodness}^f(Class, 0.0375) = 3.2$.

Because the condition of Step (6.5) is satisfied,

$$Class^{0.0375} \{ \{D_1, D_2, D_3\}; \{D_4, D_5, D_6\} \}$$

is output as the best classification in Step (7).

We have a theorem for the *BestClassification* algorithm as follows.

Theorem 5 *Algorithm BestClassification works correctly.*

Proof: Steps (6) and (7) search for the best classification. We need to show that if there is a best classification for the given databases, the best classification can be found, and if there is no such a best classification for these databases, a trivial best classification is reported.

For m given databases, there are at most $m^2/2$ different similarities according to sim_1 and Property 3.1. The similarities can divide $[0, 1]$ into $m^2/2 - 1$ subsets. For each subset, $distance_{Goodness}^f(class, \alpha)$ is a constant. This means that $distance_{Goodness}^f(class, \alpha)$ can get at most $m^2/2 - 1$ different values for $\alpha \in [0, 1]$ and there are at most $m^2/2 - 1$ different classifications for these databases. In particular, a classification for the databases is best when $distance_{Goodness}^f(class, \alpha)$ gets the minimal value among the $m^2/2 - 1$ different values. Therefore, if it is in existence, the polar-point of $distance_{Goodness}^f(class, \alpha)$ can be obtained in Step (6) by, at most, $Max\{m^2/2 - 1, Int(1/\lambda)\}$ comparisons, where $Int(1/\lambda)$ is an integer function and λ is the size of the step of searching. This means that if there is a best classification for the databases, the classification can be found by *BestClassification*.

If there is no non-trivial best classification, the value of $class$ is not changed, and $class = Class^1$ is output in Step (7). This means that the trivial best classification $Class^1$ is reported by *BestClassification*. (Note that $Class^1 \leq Class^0$.)

□

In the algorithm *BestClassification*, the procedure *GreedyClass* is called in. The space complexity of *BestClassification* is approximately equal to the space complexity of *GreedyClass*.

The time complexity of *BestClassification* is dominated by the loop from Step (3) to Step (6). The body of the loop is performed, at most, $Int(1/\lambda)$ times. The procedure *GreedyClass* is called, at most, $Int(1/\lambda)$ times during looping. So, the time complexity of *BestClassification* is $O(hm^4)$, where $h = Int(1/\lambda)$ and $O(m^4)$ is the time complexity of *GreedyClass*, by taking away the step for generating the relation table of similarity.

4 Experiments

To study the effectiveness of our proposed approach, we have performed several experiments. The *BestClassification* algorithm in Section 4 is implemented on Dell using Java.

4.1 Evaluation of Database Classification Measures

To obtain multiple possibly-relevant databases, we adopted some techniques in [7, 8, 20]. We vertically partition a database into a number of subsets, each of which contains a certain number of attributes. It is hoped that some databases obtained this way are relevant to each other in classes. In our experiments, multiple databases are generated from the Synthetic Classification Data Sets on the Internet (<http://www.kdnuggets.com/>). We report our results on four databases from the above website. The main properties of the four databases are the following. There are $|R| = 1000$ attributes, and the average number T of attributes per row is 5, 6, 7, and 8 respectively. The number $|r|$ of rows is approximately 3000. The average size I of maximal frequent itemsets is 4. The databases are vertically partitioned into 3, 4, 5, and 6 subsets, respectively. Table 6 summarizes these parameters.

Table 6 *Synthetic data set characteristics*

<i>Data set name</i>	$ R $	T	$ r $	<i>subset</i>
<i>T5.I4.D100K</i>	1000	5	2969	<i>T5S1, T5S2, T5S3</i>
<i>T6.I4.D100K</i>	1000	6	2983	<i>T6S1, ..., T6S4</i>
<i>T7.I4.D100K</i>	1000	7	3024	<i>T7S1, ..., T7S5</i>
<i>T8.I4.D100K</i>	1000	8	3101	<i>T8S1, ..., T8S6</i>

For a given set D of m databases, the number n (or $|class|$) of elements (classes) in $class(D, sim_1, \alpha)$ is relative to α . It can be described by point-pairs in a function $n = f(\alpha)$, where, $1 \leq n \leq m$, $\alpha \in [0, 1]$. The relationship between the number of classes and α for complete classifications is illustrated in Figure 1 as follows.

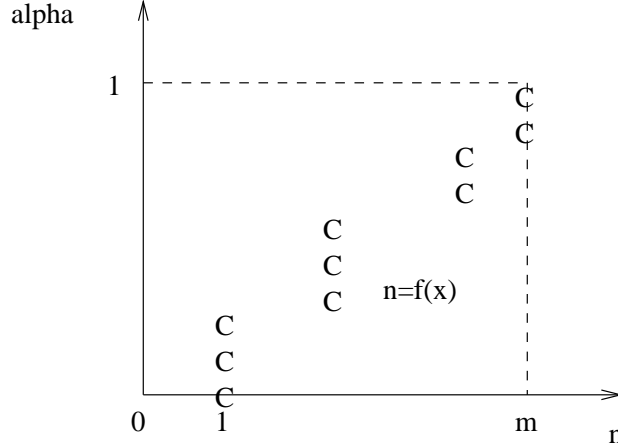


Figure 1: The relationship between $|class|$ and α for complete classifications

In Figure 1, we use a sequence of “C”s to replace a line segment for describing $|class|$ (the number of elements in the *class*) when $a \leq \alpha \leq b$. For the two trivial complete classifications, $n = 1$ when $\alpha = 0$; n is generally asymptotic to m when $\alpha \approx 1$. The relation $n = f(\alpha)$ is a jumping function in the rectangle enclosed by the n axis and three lines, $n = 1$, $\alpha = 1$, and $n = m$.

$f(\alpha)$ is a jumping function because the number of the elements in a *class* is an integer.

On the other hand, for a different α , we have obtained a different *Goodness* from complete classifications for the given databases. The relationship between *Goodness* and α for complete classifications is sketched in Figure 2.

In Figure 2, we use a sequence of “X”s to replace a line segment for describing the value of *Goodness* when α belongs to a small interval in $[0, 1]$. For the two trivial complete classifications, *Goodness* = 0 when $\alpha = 1$, and *Goodness* gets the maximum value when $\alpha = 0$. *Goodness* is a jumping function in the rectangle enclosed by the n axis, the *alpha* axis, and two lines, *alpha* = 1 and $n = m^2 - m$.

For the databases in Table 6, we applied sim_1 to measure the similarity as follows.

- (1) $sim_1(T5S1, T5S2) = 0.76$, $sim_1(T5S1, T5S3) = 0.831$, $sim_1(T5S1, TiSj) = 0$ for other datasets; $sim_1(T5S2, T5S3) = 0.785$, $sim_1(T5S2, TiSj) = 0$ for other datasets; and $sim_1(T5S3, TiSj) = 0$ for other datasets ($i \neq 5$).
- (2) $sim_1(T6S1, T6S2) = 0.804$, $sim_1(T6S1, T6S3) = 0.783$, $sim_1(T6S1, T6S4) = 0.83$, $sim_1(T6S1, TiSj) = 0$ for other datasets; $sim_1(T6S2, T6S3) = 0.773$, $sim_1(T6S2, T6S4) = 0.693$, $sim_1(T6S2, TiSj) = 0$ for other datasets; $sim_1(T6S3, T6S4) = 0.715$, $sim_1(T6S3, TiSj) = 0$ for other datasets; and $sim_1(T6S4, TiSj) = 0$ for other datasets ($i \neq 6$).
- (3) $sim_1(T7S1, T7S2) = 0.701$, $sim_1(T7S1, T7S3) = 0.73$, $sim_1(T7S1, T7S4) = 0.743$, $sim_1(T7S1, T7S5) = 0.63$, $sim_1(T7S1, TiSj) = 0$ for other datasets; $sim_1(T7S2, T7S3) = 0.711$, $sim_1(T7S2, T7S4)$

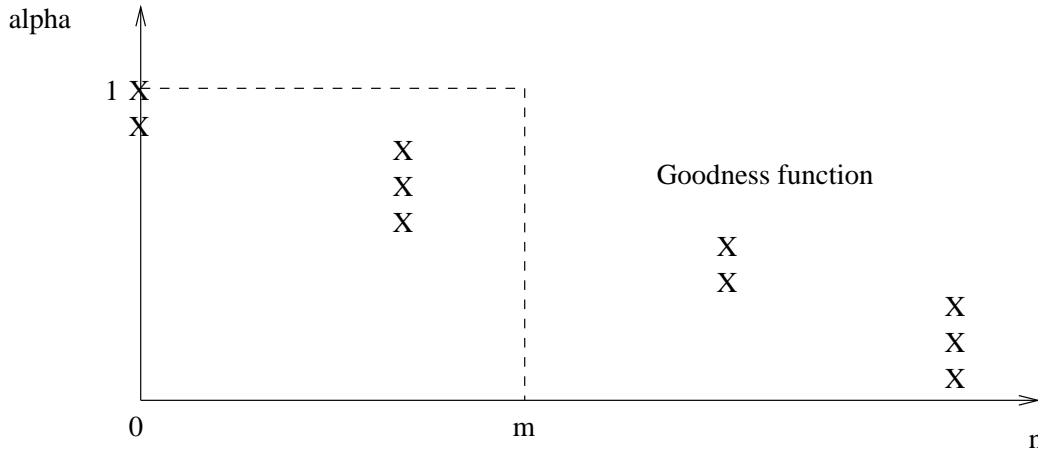


Figure 2: The relationship between the *Goodness* and α for complete classifications

$= 0.652$, $sim_1(T7S2, T7S5) = 0.68$, $sim_1(T7S2, TiSj) = 0$ for other datasets; $sim_1(T7S3, T7S4) = 0.67$, $sim_1(T7S3, T7S5) = 0.71$, $sim_1(T7S3, TiSj) = 0$ for other datasets; $sim_1(T7S4, T7S5) = 0.75$, $sim_1(T7S4, TiSj) = 0$ for other datasets; and $sim_1(T7S5, TiSj) = 0$ for other datasets ($i \neq 7$).

- (4) $sim_1(T8S1, T8S2) = 0.661$, $sim_1(T8S1, T8S3) = 0.673$, $sim_1(T8S1, T8S4) = 0.802$, $sim_1(T8S1, T8S5) = 0.672$, $sim_1(T8S1, T8S6) = 0.661$, $sim_1(T8S1, TiSj) = 0$ for other datasets; $sim_1(T8S2, T8S3) = 0.721$, $sim_1(T8S2, T8S4) = 0.706$, $sim_1(T8S2, T8S5) = 0.724$, $sim_1(T8S2, T8S6) = 0.761$, $sim_1(T8S2, TiSj) = 0$ for other datasets; $sim_1(T8S3, T8S4) = 0.715$, $sim_1(T8S3, T8S5) = 0.63$, $sim_1(T8S3, T8S6) = 0.71$, $sim_1(T8S3, TiSj) = 0$ for other datasets; $sim_1(T8S4, T8S5) = 0.686$, $sim_1(T8S4, T8S6) = 0.651$, $sim_1(T8S4, TiSj) = 0$ for other datasets; $sim_1(T8S5, T8S6) = 0.712$, $sim_1(T8S5, TiSj) = 0$ for other datasets; and $sim_1(T8S6, TiSj) = 0$ for other datasets ($i \neq 8$).

For Table 6, the relationship between $|class|$ and α and, the relationship between the *Goodness* and α for complete classifications are illustrated in Figure 3.

In Figure 3, when α belongs to a small interval in $[0, 1]$, we use a sequence of “C”s to replace a line segment for describing the number of the elements in *class* and, a sequence of “X”s to replace a line segment for describing the value of *Goodness*. For the two trivial complete classifications, $n = 18$ and *Goodness* = 0 when $\alpha = 1$; and $n = 1$ and *Goodness* gets the maximum value 257.128 when $\alpha = 0$. In particular, there is no non-trivial complete classification when $\alpha \in (0.63, 0.831]$.

This example illustrates the fact that we can obtain the best classification when the distance between n and *Goodness* gets the smallest value for $\alpha \in [0, 1]$.

Also, the relationship between $distance_{Goodness}^f$ and α for complete classifications is sketched in Figure 4.

In Figure 4, we use a sequence of “d”s to replace a line segment for describing the value of $distance_{Goodness}^f$ when α belongs to a small interval in $[0, 1]$. For the two trivial complete classifications, *Goodness* = 0 when $\alpha = 1$; and *Goodness* gets the maximum value when $\alpha = 0$. *Goodness* is a jumping function in the rectangle enclosed by the n axis, the α axis, and two lines, $\alpha = 1$ and $n = m^2 - m$.

For Table 6, the absolute of the difference between $|class|$ and *Goodness* by α is illustrated in Figure 5.

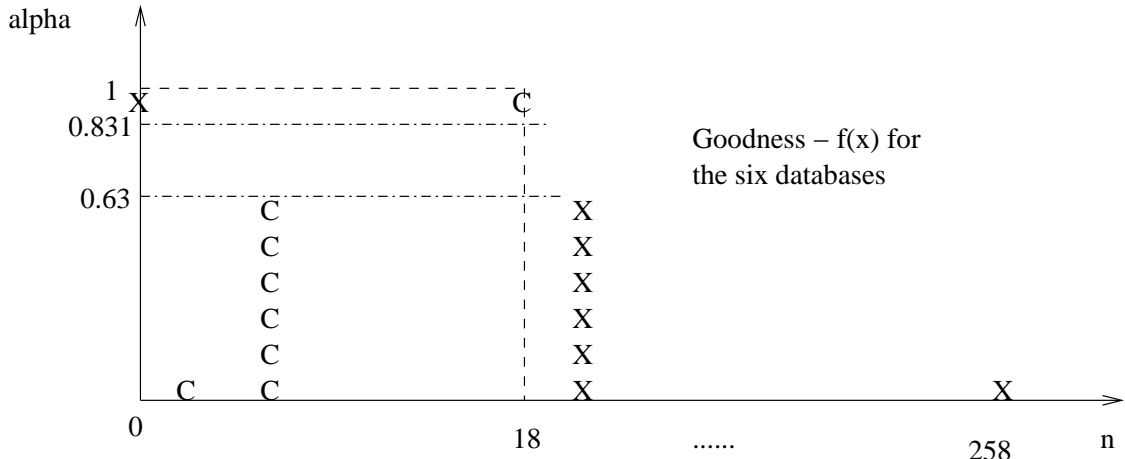


Figure 3: $|class|$, $Goodness$ and α for complete classifications

In Figure 5, we use a sequence of “d”s to replace a line segment for describing the value of $distance_{Goodness}^f$ when α belongs to a small interval in $[0, 1]$ for datasets in Table 6.

4.2 Effectiveness of Database Classification

To assess the effectiveness of our proposed approach, we report two sets of experiments using both synthetic and real-world databases.

Firstly, we generate four databases: DB_1 , DB_2 , DB_3 and DB_4 , from the Synthetic Classification Data Sets. There are $|R| = 1000$ attributes, and the average number T of attributes per row is 5. The number $|r|$ of rows is approximately 10000. The results are listed in Tables 7 and 8.

Table 7 *The effectiveness of BestClassification with Sim₁*

	DB_1	DB_2	DB_3	DB_4
DB_1	1	0.538	0.429	0
DB_2	0.538	1	0.667	0.176
DB_3	0.429	0.667	1	0.250
DB_4	0	0.176	0.250	1

In Table 7, when $\alpha = 0.429$, the best classification is $\{\{DB_1, DB_2, DB_3\}, \{DB_4\}\}$.

Table 8 *The effectiveness of BestClassification with Sim₂*

	DB_1	DB_2	DB_3	DB_4
DB_1	1	0.221	0.287	0
DB_2	0.221	1	0.157	0.089
DB_3	0.287	0.157	1	0.006
DB_4	0	0.089	0.006	1

In Table 8, $minsupp = 0.005$ for identifying itemsets of interest. When $\alpha = 0.287$, the best classification is $\{\{DB_1, DB_3\}, \{DB_2\}, \{DB_4\}\}$.

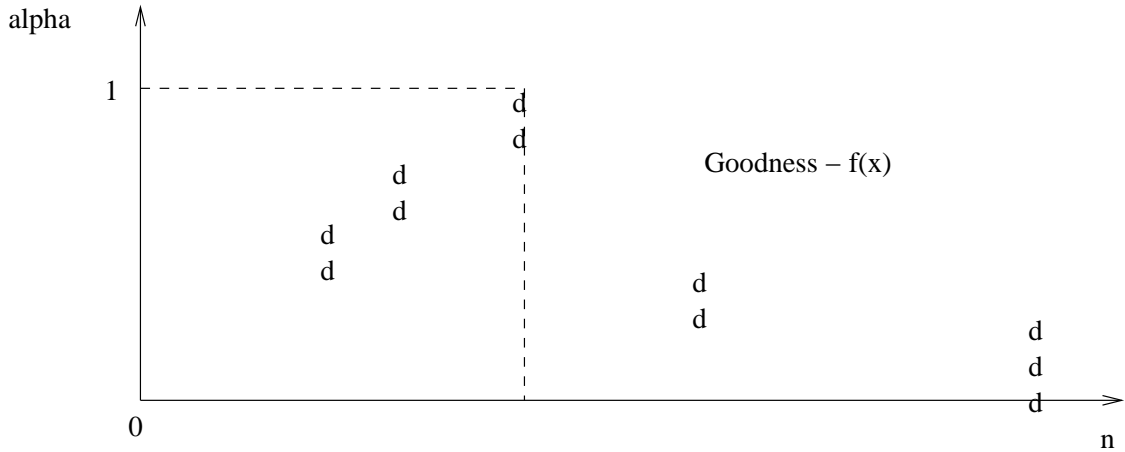


Figure 4: The difference of $|class|$ and $Goodness$ for complete classifications

Secondly, we generate six databases: $DB_1, DB_2, DB_3, DB_4, DB_5$ and DB_6 , from the Synthetic Classification Data Sets. There are $|R| = 1000$ attributes, and the average number T of attributes per row is 6. The number $|r|$ of rows is approximately 10000. The results are listed in Tables 9 and 10.

Table 9 *The effectiveness of BestClassification with Sim_1*

	DB_1	DB_2	DB_3	DB_4	DB_5	DB_6
DB_1	1	0.667	0.429	0.333	0	0.111
DB_2	0.667	1	0.667	0.333	0	0
DB_3	0.429	0.667	1	0.538	0	0
DB_4	0.333	0.333	0.538	1	0	0
DB_5	0	0	0	0	1	0.333
DB_6	0.111	0	0	0	0.333	1

In Table 9, when $\alpha = 0.333$, the best classification is $\{\{DB_1, DB_2, DB_3, DB_4\}, \{DB_5, DB_6\}\}$.

Table 10 *The effectiveness of BestClassification with Sim_2*

	DB_1	DB_2	DB_3	DB_4	DB_5	DB_6
DB_1	1	0.585	0.370	0.299	0	0.102
DB_2	0.585	1	0.569	0.297	0	0
DB_3	0.370	0.569	1	0.462	0	0
DB_4	0.299	0.297	0.462	1	0	0
DB_5	0	0	0	0	1	0.296
DB_6	0.102	0	0	0	0.296	1

In Table 10, $minsupp = 0.005$ for identifying itemsets of interest. When $\alpha = 0.296$, the best classification is $\{\{DB_1, DB_2, DB_3, DB_4\}, \{DB_5, DB_6\}\}$.

Finally, we generate six databases from real-world data sets at <http://bbs.nju.edu.cn/vd134894/main.html> <ftp://pami.sjtu.edu.cn/>

Two of them are DB_1 and DB_2 generated from Zoo, and four of them are DB_3, DB_4, DB_5 and DB_6 generated from Mushroom. The results are listed in Tables 11 and 12.

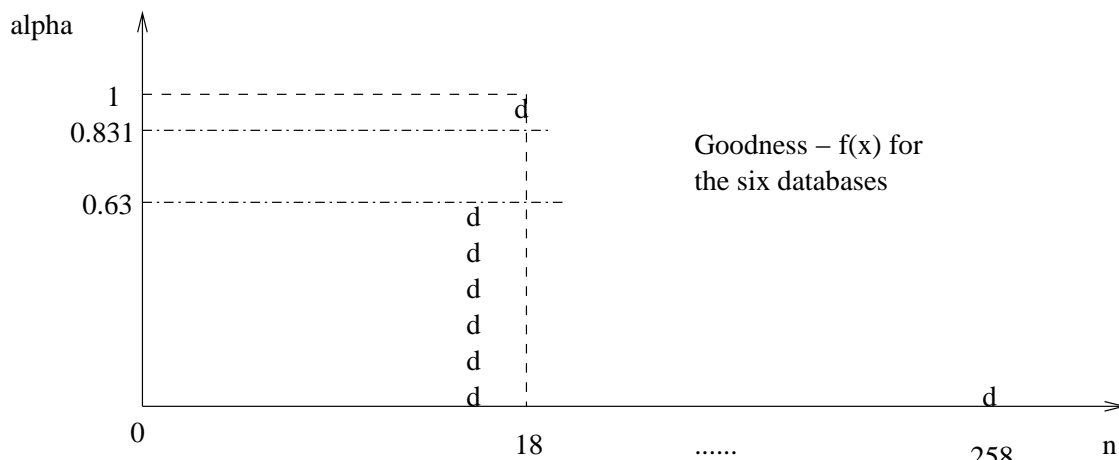


Figure 5: The difference of $|class|$ and *Goodness* for complete classifications

Table 11 *The effectiveness of BestClassification with Sim₁*

	DB_1	DB_2	DB_3	DB_4	DB_5	DB_6
DB_1	1	0.952	0	0	0	0
DB_2	0.952	1	0	0	0	0
DB_3	0	0	1	0.813	0.584	0.468
DB_4	0	0	0.813	1	0.727	0.544
DB_5	0	0	0.584	0.727	1	0.652
DB_6	0	0	0.486	0.544	0.652	1

In Table 11, when $\alpha = 0.486$, the best classification is $\{\{DB_1, DB_2\}, \{DB_3, DB_4, DB_5, DB_6\}\}$.

Table 12 *The effectiveness of BestClassification with Sim₂*

	DB_1	DB_2	DB_3	DB_4	DB_5	DB_6
DB_1	1	0.895	0	0	0	0
DB_2	0.895	1	0	0	0	0
DB_3	0	0	1	0.493	0.486	0.384
DB_4	0	0	0.493	1	0.613	0.492
DB_5	0	0	0.486	0.613	1	0.603
DB_6	0	0	0.384	0.492	0.603	1

In Table 12, $minsupp = 0.1$ for identifying itemsets of interest. When $\alpha = 0.613$, the best classification is $\{\{DB_1, DB_2\}, \{DB_3\}, \{DB_4, DB_5\}, \{DB_6\}\}$. This classification has shown that interesting itemsets in Mushroom are irregularly distributed.

The above experiments have confirmed that DB_1 and DB_2 in Zoo are irrelevant to DB_3 , DB_4 , DB_5 and DB_6 in Mushroom.

5 Related Work

To mine multiple databases, the first method (mono-database mining technique) is to put all the data together from multiple databases to create a huge mono-dataset. There are various problems with this approach [21].

In order to confront the size of datasets, Liu, Lu and Yao have proposed an alternative multi-database mining technique that selects relevant databases and searches only the set of all relevant databases [7, 8].

Their work has focused on the identification of databases that are most relevant to an application. A relevance measure was thus proposed to identify relevant databases for mining with an objective to find patterns or regularity within certain attributes. This can overcome the drawbacks that are the result of forcedly joining all databases into a single huge database upon which existing data mining techniques or tools are applied. This approach is effective in reducing search costs for a given application.

Identifying relevant databases in [7, 8] is referred to as database selection. In real-world applications, database selection needs, however, multiple times to identify relevant databases to meet different applications. In particular, the users may need to mine their multiple databases without specifying any application, and in this case, the database selection approach does not work. The database selection approach is application-dependent.

While data mining techniques have been successfully used in many diverse applications, multi-database mining has only been recently recognized as an important research topic in the data mining community. Yao and Liu have proposed a means of searching for interesting knowledge in multiple databases according to a user query. The process involves selecting all interesting information from many databases by retrieval. Mining only works on the selected data [20].

Zhong et al have proposed a way of mining peculiarity patterns from multiple statistical and transaction databases based on previous work [22]. A peculiarity pattern is discovered from the peculiar data by searching the relevance among the peculiar data. Roughly speaking, a data item is peculiar if it represents a peculiar case described by a relatively small number of objects and is very different from other objects in a data set.

A related research effort is distributed data mining (DDM) that deals with different possibilities of data distribution. A famous effort is hierarchical meta-learning [9] which has a similar goal of efficiently processing large amounts of data. Meta-learning starts with a distributed database or a set of data subsets of an original database, concurrently runs a learning algorithm (or different learning algorithms) on each of the subsets, and combines the predictions from classifiers learned from these subsets by recursively learning 'combiner' and 'arbiter' models in a bottom-up tree manner [9]. The focus of meta-learning is to combine the predictions of learned models from the partitioned data subsets in a parallel and distributed environment.

There are also other related research efforts. Wu and Zhang have advocated an approach for identifying patterns in multi-database by weighting [19]. Ribeiro, Kaufman and Kerschberg have described a way of extending the INLEN system for multi-database mining by incorporating primary and foreign keys, as well as developing and processing knowledge segments [11]. Wrobel has extended the concept of foreign keys to include foreign links, since multi-database mining also involves accessing non-key attributes [17]. Aronis et al. introduced a system called WoRLD that uses spreading activation to enable inductive learning from multiple tables in multiple databases spread across the network [1]. Kargupta et al. have built a collective mining technique for distributed data [6]. Grossman et al. have built a system, known as Papyrus, for distributed data mining [4, 14]. Existing parallel mining techniques can also be used to deal with multiple databases [2, 3, 9, 10, 13].

The above efforts have provided good insights into multi-database mining. However, they are inadequate for the 2-step database classification situation we have addressed in this paper.

6 Conclusions

The development of effective database classification techniques is one of the key issues in the design of multi-database mining systems. The classification strategy developed in this paper provides a way for designing multi-database mining systems that are application independent, and improves the performance of multi-database mining systems because of the reduction of the search cost. Our experiments have shown that our proposed approach is effective and promising.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments on an earlier version of this paper.

References

- [1] J. Aronis et al, The WoRLD: Knowledge discovery from multiple distributed databases. *Proceedings of 10th International Florida AI Research Symposium*, 1997: 337-341.
- [2] J. Chattratichat, etc., Large scale data mining: challenges and responses. *The Third International Conference on Knowledge Discovery & Data Mining*, 1997: 143-146.
- [3] D. Cheung, V. Ng, A. Fu and Y. Fu, Efficient Mining of Association Rules in Distributed Databases, *IEEE Trans. on Knowledge and Data Engg.* Vol. 8 6(1996): 911-922.
- [4] R. Grossman, S. Bailey, A. Ramu, B. Malhi and A. Turinsky, The Preliminary Design of Papyrus: A System for High Performance, Distributed Data Mining over Clusters. In: *Advances in Distributed and Parallel Knowledge Discovery*, AAAI Press/The MIT Press, 2000: 259-275.
- [5] J. Han, J. Pei, and Y. Yin, Mining frequent patterns without candidate generation. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2000: 1-12.
- [6] H. Kargupta, W. Huang, K. Sivakumar, B. Park, and S. Wang, Collective Principal Component Analysis from Distributed, Heterogeneous Data. In: *Principles of Data Mining and Knowledge Discovery*, 2000: 452-457.
- [7] H. Liu, H. Lu, and J. Yao, Identifying Relevant Databases for Multidatabase Mining. In: *Proceedings of the 2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining*, April 15-18, 1998: 210-221.
- [8] H. Liu, H. Lu, and J. Yao, Toward Multidatabase Mining: Identifying Relevant Databases. *IEEE Trans. on Knowledge and Data Engg.* Vol. 13 4(2001): 541-553
- [9] A. Prodromidis, and S. Stolfo. Pruning meta-classifiers in a distributed data mining system. In: *Proc. of the First National Conference on New Information Technologies*, 1998: 151-160.
- [10] 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.
- [11] J. Ribeiro, K. Kaufman, and L. Kerschberg, Knowledge discovery from multiple databases. In: *Proceedings of KDD95*. 1995: 240-245.
- [12] V. Seshadri, S. Weiss and R. Sasisekharan, Feature extraction for massive data mining. In: *Proceedings of The First International Conference on Knowledge Discovery & Data Mining*, 1995: 258-262.
- [13] T. Shintani and M. Kitsuregawa, Parallel mining algorithms for generalized association rules with classification hierarchy. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1998: 25-36.
- [14] K. Turinsky and R. Grossman, A framework for finding distributed data mining strategies that are intermediate between centralized strategies and in-place strategies. In: *Proceedings of Workshop on Distributed and Parallel Knowledge Discovery at KDD-2000*, 2000: 1-7.
- [15] Geoffrey I. Webb, Efficient search for association rules. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* , 2000: 99-107.
- [16] D.H. Wolpert, Stacked Generalization. *Neural Networks*, 5(1992): 241-259.
- [17] S. Wrobel, An algorithm for multi-relational discovery of subgroups. In: J. Komorowski and J. Zytkow (eds.) *Principles of Data Mining and Knowledge Discovery*, 1997: 367-375.
- [18] X. Wu and W. Lo, Multi-Layer Incremental Induction. In: *Proceedings of the 5th Pacific Rim International Conference on Artificial Intelligence*, Singapore, 1998, 24-32.
- [19] Xindong Wu and Shichao Zhang, Synthesizing High-Frequency Rules from Different Data Sources. *IEEE Transactions on Knowledge and Data Engineering*, Vol 15 (2003), No. 2, 353-367.

- [20] J. Yao and H. Liu, Searching Multiple Databases for Interesting Complexes. In: *Proceedings of the 1st Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 1997: 198-210.
- [21] Shichao Zhang, Xindong Wu and Chengqi Zhang, Multi-Database Mining, *The IEEE Computational Intelligence Bulletin*, Vol 2, 2003.
- [22] N. Zhong, Y. Yao, and S. Ohsuga, Peculiarity oriented multi-database mining. In: *Proceedings of PKDD'99*, 1999: 136-146.