Mining Approximate Dependencies Using Partitions on Similarity-relation-based Fuzzy Databases

Shyue-liang Wang*, Jenn-Shing Tsai** and Been-Chian Chien**
*Department of Information Management
I-Shou University, Taiwan
slwang@csa500.isu.edu.tw
**Institute of Information Engineering
I-Shou University, Taiwan
{m873202m|cbc}@csa500.isu.edu.tw

ABSTRACT

In this work, we present a data mining technique for determining approximate dependencies in similarity-relation-based fuzzy databases. Similarity-relation-based fuzzy data model is most suitable for describing analogical data over discrete domains, in addition to fuzzy-set-based fuzzy data models. Approximate dependency is an extension of functional dependency such that equality of tuples is extended and replaced with the notion of equivalence class. The approximate dependency we define here can capture more real-world integrity constraints then most existing functional dependencies on fuzzy databases. A level-wise mining technique is adopted here for the search of all possible nontrivial minimal approximate dependencies based on equivalence classes of attribute values. An algorithm based on Huhtala [5] is presented here whereas other approximate types of functional dependencies introduce only conceptual viewpoints.

Keyword: data mining, approximate dependency, similarity relation, fuzzy database

1. INTRODUCTION

Data mining, also referred to as knowledge discovery on databases, is the nontrivial extraction of implicit, previously unknown, and potentially useful information from data [4]. It is the search for relationships and patterns that exist in large databases, but are "hidden" among the vast amount of data. Association rules, classification rules, discrimination rules, characteristic rules, functional dependencies, and clusterings are some of the relationships and patterns typically searched.

Mining of functional and approximate dependencies from relations has been identified as an important database analysis technique. Functional dependencies are relationships between attributes of a relation: a functional dependency states that the value of an attribute is uniquely determined by the values of some other attributes. The discovery of functional dependencies from relations has received considerable interests. An approximate dependency can be considered as a functional dependency that almost holds. It describes approximate relationships between attributes of a relation in databases. Many mechanisms for incorporating approximate relationships have been proposed previously: the fuzzy functional dependency framework of Raju and Majumdar [9], the probabilistic dependency framework of Haux and Eckert [5], and cluster dependency framework of Saharia and Barron [10], etc. However, current mining techniques of determining functional and approximate dependencies are based on crisp databases. Although various forms of approximate dependency on fuzzy databases have been proposed, their emphases are on the conceptual viewpoints, and no algorithms are given.

In this work, we present an approach for finding approximate dependencies from databases that contain similarity-relation-based fuzzy data. A similarity relation is a generalization of an equivalence relation such that reflexiveness, symmetry, and max-min transitivity is satisfied. In addition to fuzzy-set-based fuzzy databases, similarity-relation-based fuzzy data model has been proposed by Buckles and Petry [3] as an important tool for storing fuzzy information in database technology. It is also recognized as a model that is most suitable for describing analogical data over discrete domain [8]. When a similarity relation is defined on an attribute domain, its attribute values can be partitioned into equivalence classes with respect to a specified level value. Attribute values in an equivalence class are similar to each other to the degree of the given level value.

The approximate dependency in our fuzzy database is defined as a functional dependency based on equivalence classes of attribute values. If attribute A is approximately dependent on attributes in X, it means that each equivalence class in A is functionally dependent on an equivalence class in X. For each attribute with a corresponding level value, the tuples in database are partitioned into a set of equivalence classes. The use of partitions on the tuples makes the discovery of approximate dependencies easy and efficient as tuple-wise computation is reduced to equivalence-class-wise manipulation. Our algorithm is based on the level-wise algorithm that has been used in many data mining applications [6,7]. It starts from dependencies with a small left-hand side and works towards larger dependencies, until the minimal dependencies that hold are found.

The rest of our paper is organized as follows. Section 2 reviews the basic concept of partitions and approximate dependencies on fuzzy databases based on similarity relations. Section 3 describes the searching algorithm of finding the approximate dependencies based on Huhtala etc. Finally a discussion is given at the end of paper.

2. PARTITIONS AND APPROXIMATE DEPENDENCIES ON FUZZY DATABASES

Fuzzy relational databases to accommodate incomplete, imprecise or uncertain information have been studied extensively in recent years [1-3,11]. To represent fuzzy information in the data model, two basic approaches can be classified: model through similarity relations (or proximity relations) [3] and model through possibility distribution [1,2]. Similarity-relation-based fuzzy database has been recognized as a model that is most suitable for describing analogical data over discrete domains. In the following, we review the basic concept of the model.

For each attribute domain D, a similarity relation s is defined over its domain elements: $s : D \times D \rightarrow [0, 1]$. A similarity relation is a generalization of an equivalence relation in that if a, b, c $\in$ D, then s is

reflexive:	$s(a, a) = 1$
symmetric:	$s(a, b) = s(b, a)$
transitive:	$s(a, c) \geq \max[\min(s(a, b), s(b, c))]$ for all $b \in D$

A similarity-relation-based fuzzy relational database[3] is defined as a set of relations consists of tuples. Let t_i represent the i-th tuples of a relation R, it has the form $(t_{i1}, t_{i2}, \ldots, t_{im})$ where t_{ij} is defined on the domain set D_j, $1 \leq j \leq m$. Unlike the ordinary relational database, two simple but important extensions are defined. The first extension is that the tuple component t_{ij} selected from D_j is not constrained to be singleton, instead, $t_{ij} \subseteq D_j$ and $t_{ij} \neq \varnothing$. Allowing tuple component t_{ij} to be a subset of the domain D_j means that fuzzy information can be represented. If t_{ij} consists of a single element, it represents the most precise information; whereas if t_{ij} is the domain D_j itself, it corresponds to the fuzziest information. This leads to the definition [3]:

DEFINITION A fuzzy database relation R is a subset of the set cross product

$$2^{D_1} \times 2^{D_2} \times \quad \ldots \quad \times 2^{D_m},$$

where $2^{D_j} = 2^{D_j} - \varnothing$, 2^{D_j} represents the power set of D_j.

The second extension of the fuzzy relational database is that, for each domain set D_j of a relation, a similarity relation S_j is defined over the set elements: $S_j : D_j \times D_j \rightarrow [0, 1]$.

The similarity relation introduces different degrees of similarity to the elements in each domain and this is another mechanism for the representation of "fuzziness" in this fuzzy relational data model.

A simple illustration of the fuzzy database relation is shown in the Table 1 below representing the JOB, EXPERIENCE and SALARY of eight EMPLOYEES. The similarity relations for domain attributes JOB, EXPERIENCE and SALARY are shown in Fig. 1.

EMP #	JOB	EXP	SALARY
1	Salesman	3	37K
2	Design Engineer	10	40K
3	System Engineer	5	45K
4	Software Engineer	5	45K
5	Accountant	12	47K
6	Accountant	5	50
7	Secretary	10	53
8	Secretary	15	55K

Table 1 A Fuzzy Database Relation

	Sec	Sw Eng	Acct	Sys Eng	Sales-man	Dn Eng
Sec	1	0.6	0.7	0.6	0.5	0.6
SW Eng	0.6	1	0.6	0.8	0.5	0.8
Acct	0.7	0.6	1	0.6	0.5	0.6
Sys Eng	0.6	0.8	0.6	1	0.5	0.8
Salesman	0.5	0.5	0.5	0.5	1	0.5
Dn Eng	0.6	0.8	0.6	0.8	0.5	1

JOB={Sec, Sw Eng, Acct, Sys Eng, Salesman, Dn Eng}

	3	5	10	12	15
3	1	0.9	0.7	0.7	0.5
5	0.9	1	0.7	0.7	0.5
10	0.7	0.7	1	0.9	0.7
12	0.7	0.7	0.9	1	0.7
15	0.5	0.5	0.7	0.7	1

EXPERIENCE

	37	40	45	47	50	53	55
37	1	0.9	0.7	0.7	0.5	0.5	0.5
40	0.9	1	0.7	0.7	0.5	0.5	0.5
45	.7	0.7	1	0.9	0.5	0.5	0.5
47	0.7	0.7	0.9	1	0.5	0.5	0.5
50	0.5	0.5	0.5	0.5	1	0.9	0.9
53	0.5	0.5	0.5	0.5	0.9	1	0.9
55	0.5	0.5	0.5	0.5	0.9	0.9	1

SALARY

Fig. 1 Similarity relations for attribute domains

An approximate dependency over a relation schema r is expressed as X → A, where $X \subseteq r$, and $A \in r$. Informally, an approximate dependency X → A holds if all tuples that agree on X approximately also agree on A approximately. Formally, the dependency holds or is valid in a given relation R over r if for all pairs of tuples t_i and $t_l \in R$ we have: if $[t_i]_{D_j}^{\alpha_j} = [t_l]_{D_j}^{\alpha_j}$ for all $D_j \in X$, then $[t_i]_A^{\alpha_A} = [t_l]_A^{\alpha_A}$, where $[t_i]_{D_j}^{\alpha_j}$ represents the equivalence class of tuple t_i with respect to an attribute D_j with level value α_j. We explain the notations as follows.

Two tuples t_i and t_l are equivalent with respect to an attribute D_j for a given level value α_j if t_{ij} and t_{lj} belong to the same equivalence class of D_j. The equivalence classes of D_j are determined by the level value α_j and defined by the similarity relation. For example, for attribute JOB with level value 0.8, the equivalence classes are {Secretary}, {Accountant}, {Software Engineering, System Engineering, Design Engineering}, and {Salesman}. It means that Software Engineering, System Engineering and Design Engineering are similar to each other with value higher or equal to 0.8 and they are in the same equivalence class of attribute JOB. We denote it by Software Engineering $\approx_\alpha$ System Engineering (or System Engineering $\approx_\alpha$ Design Engineering) where $\alpha = 0.8$. Tuples 2 , 3 and 4 in Table 1 therefore belong to the same equivalence class in terms of attribute JOB. In general, an attribute D_j partitions the tuples of a relation into a set of equivalence classes. We denote the equivalence class of a tuple $t_i \in R$ with respect to an attribute D_j with level value α_j by $[t_i]_{D_j}^{\alpha_j}$, i.e., $[t_i]_{D_j}^{\alpha_j} = \{t_l \in R \mid t_{lj} \approx_{\alpha_j} t_{ij}\}$. The set $\pi_{D_j}^{\alpha_j} = \{[t_i]_{D_j}^{\alpha_j} \mid t_i \in R\}$ of equivalence classes is a partition of R under D_j with level value α_j. That is, $\pi_{D_j}^{\alpha_j}$ is a collection of disjoint sets (equivalence classes) of tuples, such that each set has values belonging to an equivalence class in D_j, and the union of the sets equals the relation R. The rank $|\pi|$ of a partition is the number of equivalence classes in π'.

Example 1 Consider the relation in Table 1. Attribute JOB with attribute value 0.8 has equivalence classes {Secretary}, {Accountant}, {Software Engineering, System Engineering, Design Engineering}, and {Salesman}. For {salesman}, it forms an equivalence class of tuple {1}. For {Software Engineering, System Engineering, Design Engineering}, it forms an equivalence class of tuple {2,3,4}. The partition with respect to JOB is $\pi_{JOB}^{0.8}$ = {{1}, {2,3,4}, {5,6}, {7,8}}. The partition with respect to EXPERIENCE is $\pi_{EXP}^{0.9}$ = {{1,3,4,6}, {2,5,7}, {8}}. The partition with respect to SALARY is $\pi_{SAL}^{0.9}$ = {{1,2}, {3,4,5}, {6,7,8}}.

The concept of partition refinement gives almost directly approximate dependencies. A partition π is a refinement of another partition π' if every equivalence class in π is a subset of some equivalence class of π'. We have the following lemma [6].

Lemma 1 An approximate dependency X → A holds if and only if π_X refines $\pi_{\{A\}}$.

Example 2 Continuing example 1, to find out whether the approximate dependency JOB → SALARY holds, we can compare the partitions $\pi_{JOB}^{0.8}$ and $\pi_{EXP}^{0.9}$ and check whether $\pi_{JOB}^{0.8}$ refines $\pi_{EXP}^{0.9}$. Since equivalence class {2,3,4} of $\pi_{JOB}^{0.8}$ is not contained in any equivalence class of $\pi_{EXP}^{0.9}$, the approximate dependency JOB → SALARY does not hold. However, approximate dependency {JOB, EXPERIENCE} → SALARY holds, since $\pi_{JOB,EXP}^{(0.8,0.9)}$ = {{1}, {2}, {3,4}, {5}, {6}, {7}, {8}} is a refinement of $\pi_{SAL}^{0.9}$ = {{1,2}, {3,4,5}, {6,7,8}}.

There is an even simpler test for determining the approximate dependency X → A. If π_X refines $\pi_{\{A\}}$, then adding A to X does not increase any equivalence classes of π_X, thus $\pi_{X \cup \{A\}}$ equals π_X. Therefore, we have the following lemma [6].

Lemma 2 An approximate dependency X → A holds if and only if $|\pi_X| = |\pi_{X \cup \{A\}}|$.

3. SEARCHING ALGORITHM

An approximate dependency X → A is minimal (in R) if A is not approximately dependent on any proper subset of X, i.e., if Y → A does not hold in R for any $Y \subset X$. The dependency X → A is trivial if $A \in X$. Our task is to find the following: given a relation R, find all minimal non-trivial approximate dependencies that hold in R.

To find all minimal non-trivial dependencies X → A, we search through the space of all possible non-trivial dependencies and test the validity and minimality of each dependency. The validity test means testing $|\pi_X| = |\pi_{X \cup \{A\}}|$ as described in lemma 2, which requires computing the partitions. Minimality test reduces the number of dependencies we have to consider. For example, if we find X → A holds, then Z → A is not minimal for any proper superset Z of X.

The collection of all possible left-hand sides of dependencies is the collection of all attribute sets. They constitute a set containment lattice such as in Figure 2 [6,7]. The level-wise algorithm starts the search from the singleton sets, and work its way through the lattice level by level until the minimal dependencies that hold are found. During this level-wise search, false dependencies are eliminated as early as possible with some pruning techniques.

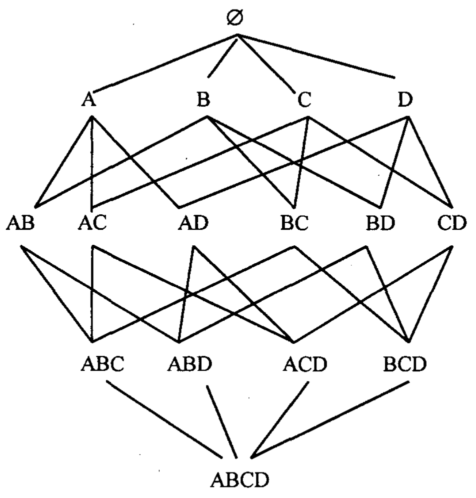

Fig. 2 The set containment lattice for {A,B,C,D} representing the search space of all possible left-hand sides

The possible right-hand sides consist of single attribute that can be obtained with a single bread-first or level-wise pass through the lattice. In addition, there is a one-to-one correspondence between the edges of the lattice and the non-trivial dependencies: the edge between sets X and X∪{A} is viewed as a non-trivial dependency X → A.

The efficiency of the level-wise algorithm is based on reducing the computation of each level by using results from previous levels. We do not need to compute the partitions from scratch for every set of attributes. In fact, the partitions of the next level in the lattice can be obtained as a product of two earlier partitions in the previous level. The product of two partitions $\pi^{'}$ and $\pi^{''}$, denoted $\pi^{'} \cdot \pi^{''}$, is the *least refined partition* that refines both $\pi^{'}$ and $\pi^{''}$.

Lemma 3 For all X, Y ⊆ r, $\pi_X \cdot \pi_Y = \pi_{X \cup Y}$.

Example 3 Continuing from Example 1, the partitions with respect to JOB, EXPERIENCE, SALARY are $\pi^{0.8}_{JOB}$ = {{1}, {2,3,4}, {5,6}, {7,8}}, $\pi^{0.9}_{EXP}$ = {{1,3,4,6}, {2,5,7}, {8}}, $\pi^{0.9}_{SAL}$ = {{1}, {2}, {3,4,5}, {6}} respectively. Partition $\pi^{(0.8,0.9)}_{JOB,EXP}$ = {{1}, {2}, {3,4}, {5}, {6}, {7}, {8}} is the least refinement of $\pi^{0.8}_{JOB}$ and $\pi^{0.9}_{EXP}$. Partition $\pi^{(0.8,0.9)}_{JOB,SAL}$ = {{1}, {2}, {3,4}, {5}, {6}} is the least refinement of $\pi^{0.8}_{JOB}$ and $\pi^{0.9}_{SAL}$. Partition $\pi^{(0.9,0.9)}_{EXP,SAL}$ = {{1}, {2}, {3,4}, {5}, {6}, {7}, {8}} is the least refinement of $\pi^{0.9}_{EXP}$ and $\pi^{0.9}_{SAL}$.

For partitions π_X, |X| ≥ 2, they are computed as a product of partitions with respect to two subsets of X. In fact, any two different subsets of size |X|-1 will do.

When the algorithm is processing a set X, it will test dependencies of the form X\{A} → A, where A ∈ X. This allows validity testing based on Lemma 2, since both π_X and $\pi_{X\setminus\{A\}}$ have already been computed. To test minimality of X\{A} → A, we need to know whether Y\{A} → A, holds for some proper subset Y of X. The information is stored in the set $C^+(X)$ of right-hand side candidates of X [6], where $C^+(X)$ = { A ∈ r | for all B ∈ X, X \ {A,B} → B does not hold}.

To find all valid minimal non-trivial dependencies, we search the set containment lattice in a level-wise manner. A level L_ℓ is the collection of attribute sets of size ℓ such that the sets in L_ℓ can potentially be used to construct dependencies based on the considerations of previous sections. We start with L_1 = {{A} | A ∈ r}, and compute L_2 from L_1, and L_3 from L_2, and so on. According to these information, we obtain the following algorithm derived from Huhtala [6], where the pruning and next level generations procedures can be found.

Algorithm: level-wise search of dependencies.
```
1.  $L_0 := \{\varnothing\}$
2.  $C^+(\varnothing) := r$
3.  $L_1 := \{\{D_j^{\alpha_j}\} \mid D_j^{\alpha_j} \in r, 1 \le j \le m\}$
4.  $\ell := 1$
5.  while $L_\ell \neq \varnothing$
6.      COMPUTE-PARTITION($L_\ell$)
7.      COMPUTE-DEPENDENCIES($L_\ell$)
8.      PRUNE($L_\ell$)
9.      $L_{\ell+1}$ := GENERATE-NEXT-LEVEL($L_\ell$)
10. $\ell := \ell + 1$
```

4. DISCUSSION

We have presented an approach for finding approximate dependencies from fuzzy relational databases based on similarity relations. The similarity-relation-based fuzzy data model has been recognized as a model that is most suitable for analogical data over discrete domains. Approximate dependencies considered here is defined as a functional dependency based on equivalence classes of attribute values, instead of equality of attribute values. The mining algorithm is based on level-wise search of Mannila. The determination of approximate dependency is an extension of Huhtala's approach, which discovers functional dependency on crisp data. The concept of approximate dependency can be further extended to other types of fuzzy data models and will widen applications in the areas of database management, reverse engineering and query optimization.

5. REFERENCE

[1] M. Anvari, and G.F. Rose, "Fuzzy relational databases", in Bezdek, Ed., Analysis of Fuzzy Information, Vol. II

(CRC Press, Boca Raton, FL, 1987).

[2] P. Bosc, and O. Pivert, "Fuzzy querying in conventional databases", In Fuzzy Logic for the Management of Uncertainty, Zadeh, L. and Kacprzyk, J. Eds, John Wiley, New York, 1992, 645-671.

[3] B.P. Buckles and F.E. Petry, "A fuzzy representation of data for relational databases", Fuzzy Sets and Systems, 7, 1982, 213-226.

[4] W.J. Frawley, G. Piatesky-Shapiro, and C.J. Matheus, "Knowledge Discovery in Databases: An Overview", Knowledge Discovery in Databases, G. Piateskey-Shapiro and W.J. Frawley, eds, pp 1-27, AAAI/MIT Press, 1991.

[5] R. Haux and U. Eckert, "Nondeterministic dependencies in relations: an extension of the concept of functional dependency", Information Systems, 10 (2), 1985, 139-148.

[6] Y. Huhtala, J Karkkainen, P. Porkka, and H. Toivonen, "Efficient discovery of functional and approximate dependencies using partitions", Proceedings of IEEE International Conference on Data Engineering, 1998, 392-410.

[7] H. Mannila and H. Toivonen, "Levelwise search and borders of theories in knowledge discovery", Data Mining and Knowledge Discovery, 1 (3), 1997, 241-258.

[8] J.M. Medina, M.A. Vila, J.C. Cubero, and O. Pons, "Towards the implementation of a generalized fuzzy relational database model", Fuzzy Sets and Systems, 75, 1995, 273-289.

[9] K.V.S.V.N. Raju and A.K. Majumdar, "Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems", ACM Transactions on Database Systems, 13 (2), 1988, 129-166.

[10] A.N. Saharia, T.M. Barron, "Approximate dependencies in database systems", Decision Support Systems, 13, 1995, 335-347.

[11] L.A. Zadeh, "Similarity relations and fuzzy orderings", Info. Sci., vol 3, no. 1, Mar. 1971, 177-200.