A Logic-Based Approach to Mining Inductive Databases

Hong-Cheu Liu[1], Jeffrey Xu Yu[2], John Zeleznikow[3], and Ying Guan[4]

[1] Department of Computer Science and Information Engineering,
Diwan University, Taiwan
hcliu@dwu.edu.tw
[2] Department of Systems Engineering and Engineering Management,
The Chinese University of Hong Kong, China
yu@se.cuhk.edu.hk
[3] School of Information Systems, Victoria University, Australia
John.Zeleznikow@vu.edu.au
[4] School of Information Technology and Computer Science,
University of Wollongong, Australia
yguan@uow.edu.au

Abstract. In this paper, we discuss the main problems of inductive query languages and optimisation issues. We present a logic-based inductive query language and illustrate the use of aggregates and exploit a new join operator to model specific data mining tasks. We show how a fixpoint operator works for association rule mining and a clustering method. A preliminary experimental result shows that fixpoint operator outperforms SQL and Apriori methods. The results of our framework could be useful for inductive query language design in the development of inductive database systems.

1 Introduction

Knowledge discovery from large databases has gained popularity and its importance is well recognised. Ever since the start of research in data mining, it has been realised that the knowledge discovery process should be supported by database technology. However, most efforts on data mining and knowledge discovery have focused on developing novel algorithms and data structures and these researches concentrated on examining the efficient implementation issues. While Data Base Management Systems (DBMS) and their enabling technology have evolved successfully to deal with most of the data-intensive application areas including decision support systems with OLAP queries, these techniques are still insufficient in a knowledge discovery environment. Therefore, databases today are still using primarily a cache-mining approach, where the data is first extracted from the database to a memory cache, which is then processed using procedural mining methods.

Research on inductive databases focuses on the integration of databases with data mining. Such integration has been formalised in the concept of inductive databases. The key ideas are that data and patterns (or models) are first class

Y. Shi et al. (Eds.): ICCS 2007, Part I, LNCS 4487, pp. 270–277, 2007.

citizen, i.e., they are handled in the same way. The emergence of Inductive DBMS research aims to improve the current state of cache mining approach and make it easy to mine databases by their query languages [1,2]. The one of crucial criteria for the promising success of inductive databases is reasoning about query evaluation and optimisation. In this paper, we will focus on inductive query languages and optimisation issues.

Several specialised inductive query languages have been proposed and implemented, such as MSQL [3], DMQL [4] and MINE RULE [5]. These projects have made a number of contributions including exploring and demonstrating some of the key features required in an Inductive DBMS. Currently, these researches have not led to significant commercial deployments due to performance concern and practical limitations.

The data mining query languages proposed in [3,4,5] require users to only provide high-level declarative queries specifying their mining goals. The underlying inductive database systems need sophisticated optimiser with aim to selecting suitable algorithms and query execution strategies in order to perform mining tasks. Another tight-coupling approach using SQL implementations gives unrealistic heavy-burden on the users to write complicated SQL queries [6]. So it is reasonable to explore alternative methods that make inductive databases realisable with current technology. Logic-based database languages provide a flexible model of representing, maintaining and utilising high-level knowledge. This motivates us to study a logic-based framework and develop relational operators (e.g., fixpoint) for intelligent data analysis.

In this paper, we consider the logic paradigm for association rule mining that use the idea of least fixpoint computation. We also demonstrate a logic query language for data mining tasks and discuss optimisation issues. Some preliminary experimental results show that our fixpoint algorithm outperforms SQL-based and the Apriori methods.

2 Inductive Query Languages

A desired feature of inductive database systems is the ability to support ad hoc and interactive data mining in order to facilitate flexible and effective knowledge discovery. Data mining query languages can be designed to support such a feature. In particular, declarative query language support acts an important role in the next generation of Web database systems with data mining functionality. Query systems should provide mechanism of obtaining, maintaining, representing and utilising high level knowledge in an unified framework. A knowledge discovery support environment should be an integrated mining and querying system capable of representing domain knowledge, extracting useful knowledge and organising in ontology [7].

Designing a comprehensive data mining language is challenging because data mining covers a wide spectrum of tasks and each task has different requirements. In this paper we provide some theoretical foundations for a logic-based data mining query language.

3 Relational Computation for Association Rules

In this section, we investigate relational computation methods and demonstrate a logic-based query paradigm for frequent itemset discovery that use the idea of least fixpoint computation.

3.1 Calculus+*Fixpoint*

We provide a non-inflationary extension of the complex value calculus with recursion and aggregate operation. We define a fixpoint operator which allows the iteration of calculus formulas up to a fixpoint. In effect, this allows us to define frequent itemsets inductively using calculus formulas.

Definition 1. *Let $\mathcal{S}^k(V)$ denote the set of all degree-k subset of V. For any two sets S and s, s is said to be a degree-k subset of S if $s \in \mathcal{P}(S)$ and $|s| = k$. $\mathcal{P}(S)$ denotes the powerset of S.*

The non-inflationary version of the fixpoint operator is presented as follows. Consider association rule mining from object-relational data. Suppose that raw data is first preprocessed to transform to an object-relational database. Let $D = (x, y)$ be a nested table in the mapped object-relational database. For example, $x = items$, $y = count$. *Items* is a set valued attribute. Let $S^k_x(D) = \{t \mid \exists u \in D, v = S^k(u[x]), t = (v, u[y])\}$. We develop a fixpoint operator for computing the frequent itemsets as follows. The relation J_n holding the frequent itemsets with support value greater than a threshold δ can be defined inductively using the following formulas

$$\varphi(T,k) = \sigma_{y\geq\delta}(\ {}_x\mathcal{G}_{sum(y)}\mathcal{S}^k_x(D)(x,y)) \longrightarrow T(x,y), \text{if } k=1$$
$$\varphi(T,k) = T(x,y) \vee \sigma_{y\geq\delta}({}_x\mathcal{G}_{sum(y)}(\exists u, v\{T(u,v) \wedge(\mathcal{S}^k_x(D)(x,y)) \wedge u \subset x \longrightarrow T(x,y)\})), \text{if } k>1$$

as follows: $J_0 = \emptyset$; $J_n = \varphi(J_{n-1}, n), n > 0$. Where $\mathcal{G}$ is the aggregation operator. Here $\varphi(J_{n-1}, n)$ denotes the result of evaluating $\varphi(T, k)$ when the value of T is J_{n-1} and the value of k is n. Note that, for each input database D, and the support threshold δ, the sequence $\{J_n\}_{n\geq 0}$ converges. That is, there exists some k for which $J_k = J_j$ for every $j > k$. Clearly, J_k holds the set of frequent itemsets of D. Thus the frequent itemsets can be defined as the limit of the forgoing sequence. Note that $J_k = \varphi(J_k, k + 1)$, so J_k is also a fixpoint of $\varphi(T, k)$. The relation J_k thereby obtained is denoted by $\mu_T(\varphi(T, k))$. By definition, μ_T is an operator that produces a new nested relation (the fixpoint J_k) when applied to $\varphi(T, k)$.

3.2 Fixpoint Algorithm

We develop an algorithm for computing frequent itemset by using the above defined fixpoint operator. We define a new join operator called *sub-join*.

Definition 2. *Let us consider two relations with the same schemes {Item, Count}. The sub-join, $r \bowtie^{sub,k} s = \{t \mid \exists u \in r, v \in s$ such that $u[Item] \subseteq v[Item] \wedge \exists t'$ such that $(u[Item] \subseteq t' \subseteq v[Item] \wedge |t'| = k)$, $t =< t', v[Count] >\}$*

Here, we treat the result of $r \bowtie^{sub,k} s$ as multiset meaning, as it may produce two tuples of $t^{'}$ with the same support value.

Example 1. *Given two relations r and s, the result of* $r \bowtie^{sub,2} s$ *is shown in Figure 1.*

r

$Items$	$Support$
$\{a\}$	0
$\{b, f\}$	0
$\{d, f\}$	0

s

$Items$	$Support$
$\{a, b, c\}$	3
$\{b, c, f\}$	4
$\{d, e\}$	2

$r \bowtie^{sub,2} s$

$Items$	$Support$
$\{a, b\}$	3
$\{a, c\}$	3
$\{b, f\}$	4

Fig. 1. An example of sub-join

Given a database $D = (Item, Support)$ and support threshold δ, the following fixpoint algorithm computes frequent itemset of D.

Algorithm. *fixpoint*
Input: An object-relational database D and support threshold δ.
Output: L, the frequent itemsets in D.
Method:

begin
$L_1 := \sigma_{Support \geq \delta}(\ _{Item}\mathcal{G}_{sum(Support)}\mathcal{S}^k_{Item}(D)))$
for $(k = 2; T \neq \emptyset; k++)$ {
$S := sub\_join(L_{k-1}, D)$
$T := \sigma_{Support \geq \delta}(\ _{Item}\mathcal{G}_{sum(Support)}(S)$
$L_k := L_{k-1} \cup T$
}
return L_k;
end

procedure *sub_join*
(T: frequent k-itemset; D: database)
for each itemset $l_1 \in T$,
  for each itemset $l_2 \in D$,
    $c = l_1 \bowtie^{sub,k} l_2$
    if *has_infrequent_subset* (c, T)
    then delete c else add c to S;
return S;

procedure *has_infrequent_subset*
(c: candidate k-itemset,
T: frequent $(k-1)$-itemsets);
for each $k-1$-subset s of c
  if s not $\in T$ then return TRUE;
return FALSE;

4 A Logic Query Language

In this section, we present a logic query language to model the association rule mining, naive Bayesian classification and partition-based clustering.

Association rule mining

We present an operational semantics for association rule mining queries expressed in Datalog$^{cv,\neg}$ program from fixpoint theory.

We present a Datalog program as shown below which can compute the frequent itemsets.

1. $cand(J, C) \leftarrow freq(I, C),\ J \subset I,\ |J| = 1$
2. $large(J, C) \leftarrow cand(J, C),\ C > \delta$
3. $T(x, C_2) \leftarrow large(J, C_1),\ freq(I, C_2),$
   $x \subset I,\ J \subset x,\ |x| = max(|J|) + 1,$
4. $T(genid(), x, C) \leftarrow T(x, C),\ \neg has\_infrequent\_subset(x)$
5. $cand(x, sum < C >) \leftarrow T(id, x, C)$
6. $large(x, y) \leftarrow cand(x, y), y > \delta$

The rule 1 generates the set of *1-itemset* from the input frequency table. The rule 2 selects the frequent *1-itemset* whose support is greater than the threshold. The program performs two kinds of actions, namely, join and prune. In the join component, the rule 3 performs the *sub-join* operation on the table *large* generated in the rule 2 and the input frequency table. The prune component (rule 4) employs the Apriori property to remove candidates that have a subset that is not frequent. The test for infrequent subsets is shown in procedure $has\_infrequent\_subset(x)$.

Datalog system is of set semantics. In the above program, we treat T facts as multisets, i.e., bag semantics, by using system generated id to simulate multiset operation. The rule 5 counts the sum total of all supports corresponding to each candidate item set generated in table T so far. Finally, rule 6 computes the frequent itemsets by selecting the itemsets in the candidate set whose support is greater than the threshold.

We now show the program that defines $has\_infrequent\_subset(x)$.

$$has\_infrequent\_subset(x) \leftarrow s \subset x,\ |s| = |x| - 1,\ \forall y[large(y, C), y \neq s]$$

Once the frequent itemset table has been generated, we can easily produce all association rules.

Naive Bayesian Classification

Let us consider a relation r with attributes $A_1, ..., A_n$ and a class label C. The Naive Bayesian classifier assigns an unknown sample X to the class C_i if and only if $P(C_i|X) > P(C_j|X)$, for $1 \leq j \leq m,\ j \neq i$.

We present a datalog program demonstrating that Naive Bayesian classification task can be performed in deductive environment. The program first evaluates the frequencies of the extension of r, each class and each pair of attribute A_i and class.

$$freq\_r(A_1, ..., A_n, C, count(*)) \leftarrow r(A_1, ..., A_n, C)$$
$$freq\_class(C, count(*)) \leftarrow r(A_1, ..., A_n, C)$$
$$freq\_A_i\_class(A_i, C, count(*)) \leftarrow r(A_1, ..., A_n, C)$$

Then we obtain the probabilities of $P(A_i \mid C)$, as follows.

$$Pr\_class(C,p) \leftarrow freq\_r(A_1,...,A_n,C,n_r),\ freq\_class(C,n_c),\ p = n_c/n_r$$
$$Pr\_A\_class(A,C,p) \leftarrow freq\_A\_class(A,C,n_{A,C}),\ freq\_class(C,n_c),\ p = n_{A,C}/n_c$$

Finally, we get the answer predicate $Classifier(x_1,...,x_k,class)$.

$$\begin{aligned}
Pr(x_1,...,x_k,class,p) \leftarrow\ & r(x_1,...,x_k),\ Pr_A_class(A,class,p),\\
& \exists t_i \in Pr_A_class,\ 1 \le i \le k,\\
& x_1 = t_1.A,...,x_k = t_k.A,\\
& t_1.class = ... = t_k.class,\ p = \textstyle\prod t_i.p\\
P(x_1,...,x_k,class,p) \leftarrow\ & Pr(x_1,...,x_k,class,p_1),\\
& Pr_class(class,p_2),\ p = p_1 \times p_2\\
Classifier(x_1,...,x_k,class) \leftarrow\ & P(x_1,...,x_k,class,p),\ p = max\{P.p\}
\end{aligned}$$

Example 2. *We wish to predict the class label of an unknown sample using naive Bayesian classification, given a training data. The data samples are described by the attributes age, income, student, and credit-rating. The class label attribute, buys-computer, has two distinct values, namely, YES, NO. The unknown sample we wish to classify is* $X = (age = \text{" <= 30"}, income = \text{"}medium\text{"}, student = \text{"}yes\text{"}, credit - rating = \text{"}fair\text{"})$.

The evaluation of the query $Classifier(x_1,...,x_k,class)$ returns the answer predicate $Classifier(age = \text{" <= 30"}, income = \text{"}medium\text{"}, student = \text{"}yes\text{"}, credit - rating = \text{"}fair\text{"}, \text{"}YES\text{"})$.

Cluster analysis: partitioning method
Given a database of n objects and k, the number of clusters to form, a partitioning algorithm organises the objects into k partitions, where each partition represents a cluster. We present a deductive program performing the partitioning-based clustering task, as follows. $P(Y,C_i) \leftarrow r(X),\ 1 \le i \le k,\ Y_i = X$; $Cluster(Y,C_i,m_i) \leftarrow P(Y,C_i),\ m_i = mean\{Y\}$ Where *mean* is a function used to calculate the cluster mean value; *distance* is a similarity function. First, it randomly select k of the n objects, each of which initially represents a cluster mean. For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the distance between the object and the cluster mean. It then computes the new mean for each cluster. This process iterates until some criterion function converges, i.e., eventually, no redistribution of the objects in any cluster occurs and so the process terminates.

The following two rules show the clustering process. An operational semantics for the following datalog program P is fixpoint semantics. The immediate consequence operator, T_P is the mapping from instances of schema of P to instances of schema of P.

$$\begin{aligned}
new_cluster(X,C) \leftarrow\ & r(X),\ Cluster(Y,C,m),\ Cluster(Y,C',m'),\\
& c \neq c',\ distance(X,m) < distance(X,m'),\\
Cluster(X,C,m) \leftarrow\ & new_cluster(X,C),\ m = mean\{new_cluster.X\}
\end{aligned}$$

5 Performance and Optimisation Issues

Most performance experiments have shown that SQL-based data mining algorithms are inferior to cache-mining approach. The main-memory algorithms used in today's data mining application typically employ sophisticated data structures and try to scan the dataset fewer times compared to SQL-based algorithms which normally require many complex join operations between input tables. However, database support knowledge discovery is important when data mining applications need to analyse current data which is so large that the in-memory data structures grow beyond the size of main-memory.

An SQL3 expression mapping to a least fixpoint operator has been presented in [8]. The fixpoint operator of that article has significant strengths in term of iterative relational computation. But it also requires substantial optimisation and prune component to remove candidates that have a subset that is not frequent. The prune component can be implemented by a k-way join. The SQL3 implementation of a fixpoint operator discussed in [8] does not claim to have achieved performance level that is comparable to those SQL-based approaches, nor does it claim to have identified a query-optimisation strategy.

We argue that if SQL would allow expressing our sub-join, $\bowtie^{sub,k}$, in an intuitive manner and algorithms implementing this operator were available in a DBMS, this would greatly facilitate the processing of fixpoint queries for frequent itemset discovery.

A Datalog expression mapping to our fixpoint operator has more intuitive than SQL expressions. In our opinion, a fixpoint operator is more appropriate exploited in the deductive paradigm which is a promising approach for inductive database systems. The main disadvantage of the deductive paradigm for inductive query languages is the concern of its performance. However, optimisation techniques from deductive databases can be utilised and the most computationally intensive operations can be modularised.

There exist some opportunities for optimisation in the expressions and algorithms expressed in the deductive paradigm. Like in the algebraic paradigm, we may improve performance by exploiting relational optimisation techniques, for example, optimizing subset queries [9], index support, algebraic equivalences for nested relational operators [10]. Optimisation for data mining queries also requires novel techniques, not just extensions of object-relational optimisation technology.

We conducted several experiments on a PC with Pentium(R)4, 3.0G processor running Windows XP with 1GB memory, 1.2GB virtual memory and 40GB hard drive. These implementations were developed in C++. The preliminary experimental result shows that Calculus-Fixpoint alogrithm outperforms Apriori algorithm. We also chose SQL-92 implementation to compare with our algorithm. In SQL-92 implementation we used the sub-query approach. The experimental result shows that Calculus-Fixpoint program outperforms SQL sub-query approach.

6 Conclusion

Relational computation for association rule mining that uses the idea of least fixpoint computation has been demonstrated in this paper. We also present a logic-based inductive query language and illustrate the use of aggregates and exploit a new join operator to model specific data mining tasks. The results provide theoretical foundations for inductive database research and could be useful for data mining query language design in inductive database systems.

Acknowledgments

The work reported in this paper was supported by a grant (CUHK418205) from the Research Grants Council of the Hong Kong Special Administrative Region, China and by a Faculty of Commerce research grant from University of Wollongong. Part of the work performed while the first author was at University of Wollongong.

References

1. Raedt, L.D.: A perspective on inductive databases. SIGKDD Explorations **4** (2002) 69–77
2. Zaniolo, C.: Mining databases and data streams with query languages and rules. In: Proceedings of 4th Workshop on Knowledge Discovery in Inductive Databases, LNCS3933. (2005) 24–37
3. Imielinski, T., Virmani, A.: Msql: A query language for database mining. Data Mining and Knowledge Discovery **2** (1999) 373–408
4. Han, J., Fu, Y., Koperski, K., Wang, W., Zaiane, O.: Dmql: A data mining query language for relational databases. In: Proceedings of ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery. (1996)
5. Meo, R., Psaila, G., Ceri, S.: An extension to sql for mining association rules. Data mining and knowledge discovery **2** (1998) 195–224
6. Sarawagi, S., Thomas, S., Agrawal, R.: Integrating association rule mining with relational database systems: alternatives and implications. Data mining and knowledge discovery **4** (2000) 89–125
7. Giannotti, F., Manco, G., Turini, F.: Towards a logic query language for data mining. Database Support for Data Mining Applications, LNAI **2682** (2004) 76–94
8. Jamil, H.M.: Bottom-up association rule mining in relational databases. Journal of Intelligent Information Systems (2002) 1–17
9. Masson, C., Robardet, C., Boulicaut, J.F.: Optimizing subset queries: a step towards sql-based inductive databases for itemsets. In: Proceedings of the 2004 ACM symposium on applied computing. (2004) 535–539
10. Liu, H.C., Yu, J.: Algebraic equivalences of nested relational operators. Information Systems **30** (2005) 167–204