Association Rule Discovery with Constraint Logic Programming

PHAICHAYON KONGCHAI, NITTAYA KERDPRASOP, KITTISAK KERDPRASOP
Data Engineering Research Unit, School of Computer Engineering
Suranaree University of Technology
111 University Avenue, Nakhon Ratchasima 30000
THAILAND
Zaguraba_ii@hotmail.com, nittaya@sut.ac.th, kerdpras@sut.ac.th

Abstract: - Association rule discovery is one of a major data mining task that has gained much interest from researchers and general users. The knowledge from association mining can be used to recommend product, design catalogs and promotional management. But data processing for association rule discovery has expensive time because the relationship from data can be tremendously many more than other data mining tasks such as classification. As a consequence, most association mining software generally create so many rules from the association mining process and some of these rules are not beneficial to any users. To solve this problem, we propose to incorporate Apriori algorithm with constraint function for user to specify subset of association rules of interest items. Users can also identify length of the rules. Our Apriori-with-constraint algorithm can reduce processing time and reduce a great number of useless rules.

Key-Words: - Data mining, Association, Constraints Logic Programming

1 Introduction

Nowadays, data mining is a popular methodology adopted by many organizations in various fields as a result of data storage devices that have greater capacity but less cost, therefore people can access and use these devices and grow the massive information. Data mining is used to find the patterns of information that normally are available in large databases [4]. The pattern extraction techniques are based on statistics, machine learning and mathematical principles. The knowledge is gained from data mining and presented to users in many forms such as association rules, classification trees, clustering reports, and many more. Derived knowledge from data mining can be used in prediction and decision.

The focus of this paper is association mining. Association rules will model data by finding the relationship of data in databases. In business, practitioners are interested in the relationship of data. Because the derived knowledge in a form of association rules may be used for planning the product placement, recommending products or to stimulate the product sales, and analyzing the customers’ behavior in shopping [8]. However, association mining is known for its expensive time in processing and deriving many disinterested rule.

In this paper, we thus propose an approach to define constraints to be applied to the data mining steps. The proposed method is for reducing time of information processing as well as for removing disinterested rules. To define the constraints, the user can specify subset of association rules and define the length of rules. Thus, programming using constraint logic programming will enhance performance of processing while reducing the lines of code.

The organization of this paper is as follows. After this introduction section, Section 2 summarizes literatures that are related to the constraint association rules. In Section 3, the theory of association rules, Apriori algorithm, constraint logic programming and integrated Apriori with constraints logic programming, are discussed. In Section 4, we present the experimental results to compare the speed in processing with 3 programs: original Apriori, Apriori with function, and Apriori with constraints logic programming, and also the discussion. In Section 5, we conclude the paper.

2 Related Work

A lot of work on mining rules from data is extensive. Many researchers are interested to find some idea for enhancement the process of
association rule. In post-processing, many rules are returned from the mining association rule and the process uses a lot of execution time. Ramakrishnan Srikant, Quoc Vu and Rakesh Agrawal [9] presented mining association rule with specify the constraint by the user to receive interested rule and reduce execution time. The execution time is reduced because dropped rules is disinterested (pruning). There are several papers present improvements by using query database technique for mining association rule with constraint-based [1], [2], [3]. It is much more efficient to incorporate such constraints into the associations.

Tihomir Trifonov and Tsvetanka Georgieva. [11] presented an application that discovers the constraint-based association rules in an archive for Unique Bulgarian Bells. They used query database technique for mining association rule. The application can give association rules and implement with Java and SQL. With simple and fast method, the user can just specify subset of association rules which no need of understanding SQL or Java.

In this paper, we implemented the Apriori algorithm with constraint for mining association rule. For user can specify interested rules and reduce execution time.

3 Constrained Association Mining and Its Implementation

3.1 Association Rule

Association rule will find the pattern (rule) from large databases. The pattern has format; if...then... or if => then. For example,

Sandwich, Coke => Candy

or

if Sandwich, Coke then Candy

The rule means if the customer buys Sandwich and Coke then the customer must buys Candy as well. The mining association rule has two steps for creating rules from information [5].

Step 1, Find all frequent itemset which greater than or equal minimum support value. We can find the support value of the itemsets from all transactions.

\[
support(A) = \frac{A}{\text{Transaction}}
\]

Step 2, Generate rules with frequent 2-itemsets or more which greater than or equal minimum confidence value. We can find the confidence of the rules from formula.

\[
\text{confidence}(A \rightarrow B) = \frac{\text{support}(A \cap B)}{\text{support}(A)}
\]

Data mining discovers relationships in databases. The important step is to find all frequent itemsets. There are a lot of ideas and techniques in mining association rules. In this paper, we use Apriori algorithm for finding all frequent itemsets.

3.2 Apriori Algorithm

Rakesh Agrawal and Ramakrishnan Srikant [2] have improved algorithms AIS that it works faster and renamed this algorithm to Apriori. The Apriori algorithm uses support-based technique for pruning (cut or reduce) the number of candidate itemsets instead of enumerates all itemsets. The principle of Apriori is if an itemset is a subset of the frequent Itemset then all subset must be frequent Itemset as well. For explanation, Apriori algorithm can be used to reduce the number of Candidate itemsets. Assume a itemset is composed of three Items \{A,B,C\} and the subsets are \{A\}, \{B\}, \{C\}, \{A,B\}, \{A,C\}, \{B,C\}. If \{A,B,C\} is frequent 3-Itemset then subset of \{A,B,C\} must be frequent Itemset.

Frequent 1-itemsets = \{A\}, \{B\}, \{C\}

Frequent 2-itemsets = \{A,B\}, \{A,C\}, \{B,C\}

We presented this algorithm for finding frequent itemset and integrated algorithm with constraint logic programming because it has high performance. However, processing of algorithm has expensive time and generates disinterested rules.

3.3 Constraint Logic Programming (CLP)

CLP is integrated logic programming and constraint solving, use high techniques from Operations Research and finite mathematics [6]. Ordinary logic programming is used for iteration, called recursion. Besides, CLP has special
predicate such as for, foreach, fromto, labeling and alldifferent which makes programming easier. CLP can also solve mathematical equations.

```prolog
:- lib(ic). % include library
solve(R) :-
    R = [B, E], % define variables
    R :: 1..5, % define constraint
    A #> B, % # is special symbol for ic
    alldifferent([B]), % [B] value not the same.
    labeling(R).
```

Fig.1 Example of CLP

In figure 1 shows format of CLP [7] and query with command solve(R). This program will solve R. We will explain 2 steps of solving, first step defines R = A, B which assign value 1 to 5. Second step compares A, B where A must greater than B and B must different from backtrack. The result is R = [2, 1]; [3, 1]; [3, 2]; [4, 1]; [4, 2]; [4, 3]; [5, 1]; [5, 2]; [5, 3]; [5, 4]. According to the example CLP can give all possible answer, coding shortly, more flexible and efficient.

3.4 Apriori with Constraint Logic Programming

This research aims to develop data mining for association rule by the algorithm apriori with CLP. In designing the algorithm Apriori with CLP, we use the main concept and additional part of constraint of the algorithm as well. The concept is shown as figure 2.

- **User specify** the user must specify the minimum support, minimum Confidence and can specify subset in association rules.
- **Find all frequent itemsets** to find itemsets which greater than or equal minimum support.
- **Constraint** the conditions to be compared user-defined set with frequent itemsets. The set of user-defined is whether subset of frequent itemsets or not. Assume the user interested item \{A,B\} and disinterested item \{D\} from frequent itemsets \{A,B\}, \{A,B,C,D\}, \{E,C\}, \{A,B,D\} the result of constraint is \{A,B\}.
- **Generate Rule**, with frequent 2-itemsets or more which greater than or equal minimum confidence value. Suppose \{A,B\} is frequent 2-itemset and minimum confidence is 80%. \{A,B\} can generate rule A => B, confidence = 70% and B => A, confidence = 90% the result of generate rule is B => A.

3.5 Apriori with Constraint Logic Programming

The Apriori algorithm implements with CLP. CLP allows users to define items those are members of the association rules independently because condition is a mathematical. This program is a logic programming. In preparing data must be provided in the format logic programming (Fig.3).
- **Item** is to declare each item to the program to be tested. In Fig.3 the item consists of all 11 Items.

- **Transaction** is to declare each transaction to the program to be tested where each transaction must be in notation [ ]. In Fig.3 the transaction consists of all 12 transactions.

- **Minimum Support** is to define the minimum support or it can define at the dialog box when run the program.

To query the program for association rules must query in CLP format. In figure 4 shows query for finding association rules, where wine is subset, fruitveg is not subset in association rules, the length of the association rule rather than 1 item and minimum confidence is 100%.

```
association(R,ItemSet), ConS::[wine], ConS1::[fruitveg],
( foreach(Set,R), param(ConS,ConS1,ItemSet) do
  Set = X_ _ . length(X,LenItem),LenItem > 1, [Y, Y1] :: [X,
  ConS=Y, \ + ConS1=Y1 -> findRule([Set]-ItemSet-100),
  ! ; true
 ).
```

Fig.4 Query the program with CLP format

- **association(R, ItemSet)** is a predicate to find all frequent itemsets and store in variable R. ItemSet is a variable to store each transaction to generate association rules.

- **findRule([Set]-ItemSet-100)** is a predicate to generate association rules. [Set] is the result of constraint. 100 is minimum confidence.

- **param(ConS,ConS1,ItemSet)** is a variable declaration to import focused variable from outside the predicate foreach can be used inside.

- **ConS::[wine], ConS1::[fruitveg]** define variable ConS has a subset that is wine and variable ConS1 has a subset that is fruitveg.

- **ConS&::=Y** is a constraint. Condition is true when ConS is subset of Y.

- **\+ConS1&::=Y1** is a constraint. Condition is true when ConS is not subset of Y.

- **LenItem > 1** is the length of the association rule which more than 1 item.

- **foreach(Set,R)** is a special predicate in CLP and take value from list R into variable Set.

- **Set=X_ _** is the pattern matching from the logic programming paradigm. Set is a pattern format, which consists of a variable X that is frequent itemset and _ is anonymous variable.

When query the program with Fig.4 with using data from Fig.3 the program will show dialog box for define minimum support. In this query we define minimum support to 2. The result is shown in Fig 5.

![ECLiPSe 6.0 Toplevel](image)

```
File Query Tools Help
Query Entry
eclipse +association(R,ItemSet), ConS::[
run more Yes make interrupt

Results

Y = Y
Y1 = Y1
Yes (0.02s cpu)

Output and Error Messages

/C/;Documents and Settings/Zag;iraba_ui/Desktop/ lists.exe loaded in 0.00 seconds
If [wine] 4 then [confectionery] 4
If [softdrink, wine] 2 then [confectionery] 2
If [confectionery, softdrink] 2 then [wine] 2
```

Fig.5 The result are association rules.

The result composed of three association rules. Wine is a member of the rules but fruitveg is not. From the first rule (If [wine] 4 then [confectionery] 4) means if four people buy wine then four people will buy candy together. But if run the program without the constraint there will be 24 association rules in the result.

4 Evaluation

This section provides an evaluation of Apriori with the Constraint Logic Programming that compared time of processing with the two
programs. The first program is Original Apriori (working same as Apriori). The second program is Apriori with three functions.

1) The function for user to specify subset in association rules.

2) The function for user to specify not subset in association rules.

3) The function for user to specify the length of the association rules.

We use data-sets in figure 3. It consists of 12 transactions and duplicate transaction to 2,167, 32,530 and 63,890 transactions for observation the difference between them.

5 Conclusion

This paper present the algorithm Apriori with CLP. It can pruning disinterested frequent itemsets, the user can define subset and length in association rules that results to reduce execution time of processing. In section Evaluation, Original Apriori use execution time of processing more than Apriori with CLP and Apriori with Function. Because Apriori Original not pruning disinterested frequent itemsets. Apriori with Function is a static program because we have to edit its condition in source code when we want to but Apriori with CLP can edit at query command.

References:


[8] Ng, R.T., Lakshmanan, L.V.S., Han, J., Pang, A. Exploratory mining and pruning

