Pattern Mining Based On Utility Functions

1. Anju Philip
HSST Computer Applicaions
Govt HSS Puthuvely
Kottayam,Kerala,India

2. Phijo J Cherickal
Asst. professor in Computer Science
MACFAST, Thiruvalla
Kottayam,Kerala,India

3. Ranjini Mariam Philipose
Asst. professor in Computer Science
MACFAST, Thiruvalla
Kottayam,Kerala,India

Abstract

This work is proposed to improve association rule mining. In association rule mining only frequent patterns can be mined. Frequent patterns are determined based on a threshold value and there is no standard method to choose this threshold. Threshold is chosen by trial and error method. In utility based pattern mining item sets are mined based on their utility. Decision-theoretic concept of utility is used to guide pattern mining. We present the use of utility functions as against thresholds and constraints as the mechanism to express user preferences. Using utility functions numeric values can be assigned to each transaction to denote its utility. We examine the problem of mining patterns with the best utility values and rank them based on utility in detail.

1. Association Rule Mining

Association rule mining problem can be divided into two sub problems - frequent pattern mining and association rule generation. Frequent patterns were first mined from the transaction database and association rules are then generated from them. The complexity of mining the dataset was isolated into the frequent pattern mining step. Since then much research has gone into increasing the speed and efficiency of this step.

1.1 Frequent Pattern Mining

Frequent Pattern Mining is the mining of patterns that occur frequently in a dataset. It is an important problem in data mining. In addition to frequent patterns themselves being of interest, frequent pattern mining also plays an essential role in several data mining problems, such as mining of association rules, correlations and sequential patterns. The frequent pattern mining problem can be described informally as follows. A set of items is chosen from the domain of interest and is called a catalog. An itemset is a subset of items from the catalog. The dataset to be mined is a database of transactions, where each transaction contains an itemset. For example, in the market-basket scenario, the catalog is the set of all items sold at a store. A transaction contains the set of items bought in one visit to the store. The transaction database is a set of such transactions. The support of an itemset is the number of times it occurs in the transaction database. A support threshold is specified and an itemset is said to be frequent if its support exceeds this minimum value. Frequent pattern mining is the mining of all itemsets from the transaction database that have a support greater than the specified support threshold.

1.1.1 Constrained Frequent Pattern Mining

In constrained frequent pattern mining, constraints are expressed over properties of items and itemsets. These constraints take the form of requiring the presence or absence of items in frequent patterns or of thresholds on the values of attributes of items or itemsets that every mined frequent pattern must meet. Constraints are integrated into the frequent pattern mining algorithm rather than being used as a filtering mechanism during or after generation of association rules from frequent patterns.

The selectivity of constraints is used to reduce the number of frequent patterns mined, making frequent pattern mining and rule generation more efficient.

Constrained frequent pattern mining has the following advantages over frequent pattern mining:

- Selectivity of constraints can be exploited to make the pattern mining process faster and more efficient.
- Users can focus mining on the broad phenomena they have in mind.

Although constrained frequent pattern mining offers two major advantages over frequent pattern mining, both suffer the following disadvantages:

- Constraints are typically specified as thresholds over values of attributes of items or itemsets. This poses several problems. It is not clear how a threshold is to be chosen. A user may guess a threshold and may have to adjust the threshold if too many or too few itemsets are returned.
- Constraint pattern mining presupposes a fixed level of interest on the part of the user. Every itemset that meets the constraint is interesting and all itemsets that do not meet the constraint are not. However, not all patterns returned may be of equal interest to the user. The user may have preference for one pattern over another, but there is no way to capture this preference. For example, amongst all the itemsets that meet a price threshold, the user may prefer the more expensive ones to the less expensive ones. Such a preference cannot be captured in the threshold guided setting.

2. Why Utility Based Pattern Mining

In order to capture user preferences precisely, we examine the use of the decision theoretic concept of utility to guide pattern mining. We present the use of utility functions as against thresholds and constraints as the mechanism to capture user preferences. The use of utility functions to express user preferences permits several pattern mining problems to be formulated.

Decision Theory and Utility Decision theory deals with the making of decisions. A decision consists of choosing an action from amongst alternatives. The set of all possible actions and outcomes are enumerated. Probability measures are used to indicate the likelihood of each possible outcome for each possible action.

A utility function is a measure of outcome value. It assigns a numeric value called utility to each outcome, called the outcomes utility. A utility function thus ranks outcomes according to the preferences of the decision maker. The expected utility of an action A is calculated as the utility of all possible outcomes weighted by their probability of occurrence for A. A rational action is the one which maximizes expected utility.

Here a simplified concept of utility is used where each action is associated with an outcome and a rational action is the one which maximizes utility or leads to the outcome preferred most by the decision maker.

3. Utility Based Pattern Mining

The decision theoretic concept of utility is applied to pattern mining as follows. The user's preferences over itemsets are captured by a utility function. The utility function is a measure of the value of itemset to the miner. It assigns a numeric utility value to each itemset, such that when a miner prefers an itemset A over an itemset B, the utility of the itemset A is greater than that of itemset B. The utility function thus ranks itemsets according to user preferences. Several interesting data mining problems that use utility functions can be formulated. Utility guided pattern mining consists of using utility functions to guide pattern mining.

Utility guided pattern mining addresses the problems with frequent and constrained frequent pattern mining as follows:

- Rather than specifying thresholds, the user can describe the interestingness of itemsets by constructing an appropriate utility function. Utility functions allow expression of interestingness directly. For instance, the utility of an itemset can be expressed as a function of its attributes. The utility function imposes an ordering over itemsets and an appropriate pattern mining problem can be chosen to select itemsets of interest to the user.

- Utility functions allow user preferences to be captured precisely. An itemset with a higher utility is more interesting than another with a lower utility.

Examples

In this section, we present and discuss the market basket scenario in detail. We use it as an aid to present examples of utility functions and pattern mining based on utility.

The market basket scenario consists of a store that sells some products and buyers that buy them at the store.

Let the catalog of items sold at a store be C={a,b,c,d,e,f,g,h,i,j } .The items have prices as presented in Table 1.1. A transaction database, D is presented in Table 1.2. Each transaction in the transaction database contains the set of items bought in one visit to the store by a customer.

Item	Price
a	3
b	2
c	6
d	4
e	2
f	1
g	2
h	1
i	1
j	1

Table 1.1 : A sample catalog with prices of items

TID	Itemset
1	{f,a,b,d,h,j}
2	{a,e,h,j}
3	{c,a,i,g,j,e,d}
4	{e,f,g}
5	{i,d,b,e,g}
6	{e}
7	{d,i,g,a,f,b}
8	{b,e,f,g,h,i}

Table 1.2: A sample transaction Database, TID denotes the transaction identification number

Examples of Utility Functions

1. Let the utility of an itemset be defined as its support in the transaction database. The utility of the itemset {a,j} is 3.

Some interesting pattern mining problems using this utility function are mining the most frequent patterns of each length and mining the most frequent patterns from the transaction database.

The frequent pattern mining problem can be formulated as a utility guided pattern mining problem in two ways. A binary utility function may be used, that assigns a utility of one to itemsets with support greater than or equal to the minimum support threshold and a utility of zero to all others. For instance, if the support threshold is set at 5, then the itemsets {e} and {g} have a utility of one and all others have a utility zero. The utility of an itemset can also be defined as its support and all itemsets with utility greater than a specified utility threshold can be mined. This leads to the expression that all itemsets with support above the threshold are interesting and an itemset with greater support is more interesting than another one with lesser support. If the support threshold is set at 2, then both itemsets {a,j} and {e,g} are interesting, but {e,g} is more interesting than {a,j}.

2. In the market basket scenario, an interesting problem is mining itemsets that have fetched the highest revenues over a period of time. The utility of the itemset here is its revenue, described as the product of the price of the itemset and its support. For instance the utility of itemset {e,f,g} is 2 x 5,that is 10.

3. Suppose a supermarket chain sells its own line of cheaper products alongside other well known and possibly more expensive brands. If a transaction contains a larger share of brand name products as against products from the supermarket's line, it could signal that the user prefers brand name products. A product from the supermarket's line in this transaction would indicate that the product fulfils a need that the brand name products aren't fulfilling. On similar lines, a brand name product in a transaction with a majority of the supermarket's line of products could signal that there is a need that the supermarket line is not fulfilling yet. Utility functions can be specified that mine both such sets of itemsets.

4. Pattern Mining using Utility Functions

Let C be the catalog of all possible items. N=|C| denote the number of items in C. An itemset is defined as a subset of C. Itemsets is sets and therefore have no repetitions of items. Let L be a set of transaction identifiers. A transaction n is a two-tuple, (TID, I) where TID Σ L and I Σ Ī. If t denotes a transaction (TID, I) t.tid denotes TID and t.itemset denotes I.

4.1 Ranking Itemsets by Utility

In this section we briefly discuss the problem of ranking all itemsets in I(D) by their utility. This is the basic utility guided pattern mining problem in the mathematical sense.

To solve this problem we need to generate every itemset in I(D), calculate its utility and store it on a priority queue. It is an interesting problem in the mathematical sense since it requires the enumeration of the set I(D). We only present a broad outline of the algorithm for this problem. The actual details of enumerating the set I(D) will be examined in conjunction with the discussion of mining N best itemsets.

Algorithm util-rank-outline is presented below.

Algorithm util-rank-outline.

Inputs

1. Utility function (U)

2. Transaction database

Output: The set I(D) ranked in the descending order according to U.

Method

1. Create a priority queue PQ to store itemsets. The utility of an itemset is its priority on PQ.

2. Enumerate the set I(D).

3. For each I Σ I(D)

(a) Calculate U (I,D)

(b) Add I to PQ with priority U (I,D).

4. Return itemsets in the decreasing order of priority from PQ.

4.1.1 Analysis of the Algorithm

The algorithm takes time O (I(D)).

The priority queue uses a space that is O (I(D)), since it stores every itemset from I(D).

4.2 Mining N itemsets with the best utility values

Itemsets with the best utility values are analogous to rational actions, in that they are the itemsets that are most preferred by the user. These provide maximal utility according to the utility function the user has defined. Henceforth in this work, we will focus on the problem of mining N itemsets with the best utility values.

In order to find N itemsets with best utility values, every itemset in I(D) has to be enumerated and its utility calculated. Thus the entire search space has to be explored. Some properties of the utility function allow parts of the search space to be pruned.

The broad outline of the solution to the problem of returning N itemsets with best utility values is similar to that of the problem of ranking all itemsets in I(D) by their utility.

However, since we are only interested in finding N itemsets, a size bounded priority queue can be used to store itemsets while mining. The space taken by the priority queue then becomes O(N), much smaller compared to the space overhead of the priority queue in algorithm util-rank-outline.

Size-bounded priority queues are used in the algorithm util-nbest-outline, to mine N best itemsets

Algorithm to mine N itemsets with best utility values

The algorithm util-nbest-outline is presented below which enumerates and calculates the utility of every itemset in I(D). It uses a priority queue of size N to store N itemsets with the best utility values of the itemsets generated while mining.

Algorithm util-nbest-outline.

Inputs

1. Utility function.

2. Transaction database.

3. The number N of itemsets to be returned.

Output

N itemsets from I(D)with best utility values w.r.t. U and their utility values.

Method

```
1. Create a size-bounded priority-queue of size N pqN=create(N)
2. For each itemset I Σ I(D)do
    (a) Calculate U(I,D). The priority of I is U(I,D)
    (b) pqN.insert(I)
3. Return pqN.rankElements()
```

5. Conclusion

Traditional threshold guided pattern mining requires the user to specify thresholds over the values of attributes of itemsets and all patterns meeting the threshold are returned.

Specification of thresholds is not an easy or precise task. The user usually has to make a guess and then adjust thresholds depending on whether too few or too many patterns were returned. Threshold guided pattern mining also assumes a fixed level of interest, in that all itemsets that meet the threshold are interesting and those that do not are not.

In this work we proposed the use of the decision theoretic-concept of utility and formulated several pattern mining problems that use utility functions. Utility functions are a natural expression of user preferences. They help capture user preferences precisely. An appropriate utility-guided pattern mining problem can be chosen to express user focus.

In this work, we examined the problem of mining itemsets with best utility values in detail. This problem is particularly interesting since itemsets with the best utility values are analogous to rational actions, in that they are the itemsets that are most preferred by the user. They provide maximal utility according to the utility function the user has defined.

Bibliography

[1] Ramesh C. Agarwal, Charu C. Aggarwal, and V. V. V. Prasad. A tree projection algorithm for generation of frequent item sets. Journal of Parallel and Distributed Computing, 61(3):350–371, 2001.

[2] Michael P. Wellman and Jon Doyle. Modular utility representation for decision-theoretic planning.

[3] Gosta Grahne, Laks V. S. Lakshmanan, and Xiaohong Wang. Efficient mining of constrained correlated sets. In ICDE, pages 512–521, 2000.

[4] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In Weidong Chen, Jeffrey Naughton, and Philip A. Bernstein, editors, 2000 ACM SIGMOD Intl. Conference on Management of Data, pages 1–12. ACM Press, 05 2000

[5] Jian Pei, Jiawei Han, and Laks V. S. Lakshmanan. Mining frequent item sets with convertible constraints. In ICDE, pages 433–442, 2001.