POLYTECHNIC UNIVERSITY
Department of Computer Science / Finance and Risk Engineering

Association Rules

K. Ming Leung

Abstract: The use of association rules for local-pattern discovery in unsupervized learning systems is discussed.

Directory

Last Revision Date: December 7, 2007

Table of Contents

1. Introduction

Using for example scanners, large supermarkets and department stores have been collecting huge amount of customer transaction records in their databases. Each record lists all the items bought by a customer on a single purchase transaction. Managers would be interested to know if certain groups of items are consistently purchased together. A business can use the knowledge of these buying patterns to improve the placement of these items in the store or the layout of the mail-order catalog pages and web pages.

This need has led to the development of techniques that automatically look for associations between items that are stored in databases. Association rules are one of the most common techniques of data mining for local-pattern discovery in unsupervized learning systems.[1]

2. Market-basket Analysis

A market basket refers to a collection of items purchased by a customer in a single transaction. For our purpose here, we are not interested in the quantity of items of a given kind that is bought in

a transaction. We are only interested in the different types of items purchased. We want to analyze the accumulated collection of transactions of numerous number of customers recorded over a long period of time.

The goal is to find sets of items, or **itemsets**, that appear together in many transactions. In other words we want to discover important associations among items such that the presence of some items in a transaction will imply the presence of some other items in the same transaction.

Association rules provide information of this type in the form of if-then statements. These rules are computed from the data and, unlike the if-then rules of logic, association rules are probabilistic in nature. In association analysis the antecedent (the if part of an if-then statements) and the consequent (the then part) are itemsets that are disjoint (do not have any items in common).

In addition to the antecedent and the consequent, an association rule has two important numbers that express the degree of usefulness and uncertainty about the rule. These two numbers must be predetermined by experts familiar with the business.

The first number is called the **support** for the rule. The **support count**, sc, is the number of transactions in D that include all items in the antecedent and consequent parts of the rule. The support count divided by $|D|$ is referred to as the support, s, and is commonly expressed as a percentage).

The other number is known as the **confidence** of the rule. Confidence is the ratio of the number of transactions that include all items in the consequent as well as the antecedent (namely, the support) to the number of transactions that include all items in the antecedent.

For example if a supermarket database has $100,000$ point-of-sale transactions, out of which $2,000$ include both items A and B and 800 of these include item C, the association rule "If A and B are purchased then C is purchased on the same trip" has a support count of 800 transactions (thus a support of $0.8\% = 800/100,000$) and a confidence of $40\%(= 800/2,000)$.

One way to think of support is that it is the probability that a randomly selected transaction from the database will contain all items in the antecedent and the consequent, whereas the confidence is the conditional probability that a randomly selected transaction

will include all the items in the consequent given that the transaction includes all the items in the antecedent.

Let the entire set of distinct items that can be found in a given database, D, be given by the set

$$\mathcal{I} = \{i_1, i_2, \ldots, i_n\}.$$

This set represents the entire collection of different types of items that a given company sells. Our database D is a collection of transactions where each transaction, T, is a set of items such that $T \subseteq I$, i.e. T is a subset of I. Each transaction is identified by a label called a transaction identifier, called TID.

Let A be an itemset. A transaction T is said to contain A if and only if $A \subseteq T$. An association rule is an implication of the form $A \Rightarrow C$, where $A \subset \mathcal{I}$ and $C \subset \mathcal{I}$ are disjoint itemsets, i.e. $A \cap C = \phi$. A is the antecedent and C is the consequent.

The association rule $A \Rightarrow C$ holds with support s, where s is the percentage of transaction in D that contains $A \cup C$, i.e. the union of sets A and C. This is taken to be the probability, $P(A \cup C)$.

The association rule $A \Rightarrow C$ has confidence c in the transaction

D, where c is the percentage of transactions in D containing A that also contain C. This is taken to be the condition probability, $P(C|A)$. That is

$$s(A \Rightarrow C) = P(A \cup C)$$

$$c(A \Rightarrow C) = P(C|A) = \frac{s(A \cup C)}{s(A)}.$$

Rules that satisfy both a minimum support threshold (*min_sup*) and a minimum confidence threshold (*min_conf*) are called **strong**.

A set of items is referred to as an itemset. An itemset that contains k items is a k-itemset.

An itemset whose support is larger than a prescribed minimum support threshold, *min_sup*, is referred to as a frequent itemset. The set of all the frequent k-itemsets in D is commonly denoted by L_k.

2.1. The Two Phases of Discovering Association Rules

There are two phases in the problem of data mining association rules.

1. Find all frequent itemsets: *i.e.* all itemsets that have support s above a predetermined minimum threshold.
2. Generate strong association rules from the frequent itemsets: these association rules must have confidence c above a predetermined minimum threshold.

After the large itemsets are identified, the corresponding association rules can be derived in a relatively straightforward manner. Thus the overall performance of mining association rules is determined primarily by the first step.

Efficient counting of large itemsets is thus the focus of most association rules mining algorithms.

2.2. An Example: Finding the Frequent Itemsets

Here let us consider a very simplified version of such a transaction database, D. There are a total of 9 transactions involving a total of 6 items. Each item in D is labeled by a positive number. Transaction 001 is a point-of-sale purchase of items 1, 2 and 5. Transaction 002 is a joint purchase of items 2, 4, etc. Note that the items within each

transaction are sorted lexicographically.

TID	Items
001	$1, 2, 5$
002	$2, 4$
003	$2, 3, 6$
004	$1, 2, 4$
005	$1, 3$
006	$2, 3$
007	$1, 3$
008	$1, 2, 3, 5$
009	$1, 2, 3$

Our task here is to derive association rules with minimum confidence threshold *min_conf* of 70% between itemsets in D that have a support count of at least 2. This means that the minimum support threshold, *min_sup*, is given by $2/9 = 22\%$. Since this database has so few transactions and each transaction contains only a small number of items, we can work out everything by hand. Our first job is

to find all the itemsets that have sc ≥ 2. We can do that simply by enumeration:

$\{1\}$ with support count of 6;
$\{2\}$ with support count of 7;
$\{3\}$ with support count of 6;
$\{4\}$ with support count of 2;
$\{5\}$ with support count of 2;
$\{1, 2\}$ with support count of 4;
$\{1, 3\}$ with support count of 4;
$\{1, 5\}$ with support count of 2;
$\{2, 3\}$ with support count of 4;
$\{2, 4\}$ with support count of 2;
$\{2, 5\}$ with support count of 2;
$\{1, 2, 3\}$ with support count of 2;
$\{1, 2, 5\}$ with support count of 2;

Out of a possible number of $2^6 - 1 = 63$ potential sets, there are only 13 such frequent itemsets.

2.3. Challenges

In real-world applications, finding all frequent itemsets in a database is a nontrivial problem because:

- the number of transactions in the database can be very large and may not fit in the memory of the computer's memory. Recall that Walmart has 20 million transactions/day and a 10 terabyte database.
- the potential number of frequent itemsets is exponential to the number of different items, although the actual number of frequent itemsets can be much smaller. Often a huge number of frequent itemsets are generated, especially if *min_sup* is low. This is because if an itemset is frequent, each of its subsets is frequent as well. A long itemset will contain a combinatorial number of shorter, frequent sub-itemsets. For example, a frequent itemset of length 100, such as $\{a_1, a_2, \ldots, a_{100}\}$, contains $\binom{100}{1} = 100$ frequent 1-itemsets: $\{a_1\}, \{a_2\}, \ldots, \{a_{100}\}$, $\binom{100}{2} = 4950$ frequent 2-itemsets: $\{a_1, a_2\}, \{a_1, a_3\}, \ldots, \{a_{99}, a_{100}\}$, and so on.

The total number of frequent itemsets that it contains is thus

$$\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1 \approx 1.27 \times 10^{30}$$

- we need to develop algorithms that are scalable (their complexity should increase linearly, not exponentially, with the number of transactions) and that examine as few infrequent itemsets as possible.

3. The Apriori Algorithm

We will now discuss the Apriori algorithm. [2, 3] By convention, the algorithm assumes that items within a transaction or itemset are sorted in lexicographic order. It employs an iterative approach known as a level-wise search, where $(k-1)$-itemsets are used to explore k-itemsets.

First, the set of frequent 1-itemsets is found by scanning the database to accumulate the count for each item, and collecting those items that satisfy minimum support. The resulting set is denoted L_1.

Next, L_1 is used to find L_2, which is then used to find L_3, and so on, until no more frequent itemsets can be found. The finding of each L_k requires one full scan of D.

To improve the efficiency of the level-wise generation of frequent itemsets, one take advantage of the **Apriori property**:

All nonempty subsets of a frequent itemset must also be frequent.

This property is based on the following observation. If an itemset A does not satisfy the minimum support threshold, *min_sup*, then A is not frequent; i.e. $P(A) < min\_sup$. If an item B is added to the itemset A, then the resulting itemset $A \cup B$ cannot occur more frequently than A. Therefore $A \cup B$ is not frequent either, that is $P(A \cup B) < min\_sup$.

This property belongs to a special category of properties called antimonotone in the sense that if a set cannot pass a test, then all of its supersets will fail the same test as well.

A two-step process is used to find L_k from L_{k-1}, for $k \geq 2$:

1. **The join step:** To find L_k, a set of candidate k−-itemsets is

generated by joining L_{k-1} with itself. This set of candidates is denoted by C_k. Let ℓ_1 and ℓ_2 be itemsets in L_{k-1}. The notation $\ell_i[j]$ refers to the jth item in ℓ_i. Thus in ℓ_1, the last item and the next to the last item are given respectively by $\ell_1[k-1]$ and $\ell_1[k-2]$. Any two itemsets L_{k-1} are joined if their first $(k-2)$ items are in common. That is, members ℓ_1 and ℓ_2 are joined if

$$(\ell_1[1] = \ell_2[1]) \wedge (\ell_1[2] = \ell_2[2]) \wedge \ldots \wedge$$
$$(\ell_1[k-2] = \ell_2[k-2]) \wedge (\ell_1[k-1] < \ell_2[k-1]).$$

The condition $\ell_1[k-1] < \ell_2[k-1]$ ensures that no duplicates are generated. The resulting itemset formed by joining ℓ_1 and ℓ_2 is $\{\ell_1[1], \ell_1[2], \ldots, \ell_1[k-2], \ell_1[k-1], \ell_2[k-1]\}$.

2. **The prune step:** Set C_k is a superset of L_k, because although all the frequent k-itemsets are included in C_k, its members may or may not be frequent. One could scan the database to determine the count of each candidate in C_k and eliminate any itemset that does not meet the minimum support threshold. This would then give L_k. However, C_k can be huge, and so this could be very time-consuming.

To eliminate the infrequent itemsets, the Apriori property is used as follows. Any $(k-1)$-itemset that is not frequent cannot be a subset of a frequent k-itemset. Hence, if any $(k-1)$-itemset of a candidate k-itemset is not in L_{k-1}, then the candidate cannot be frequent either and so can be removed from C_k. This subset testing can be done quickly by maintaining a hash tree of all frequent itemsets.

3.1. An Example of the Use of The Apriori Algorithm

We illustrate the use of the Apriori algorithm for finding frequent itemsets in our transaction database, D.

In the first iteration of the algorithm, each item is a member of the set of candidates 1-itemsets, C_1. The algorithm simply scans all the transactions in order to count the number of occurrences of each item.

C_1 itemset	Support count
{1}	6
{2}	7
{3}	6
{4}	2
{5}	2
{6}	1

The set of frequent 1-itemsets, L_1, consists of the candidate itemsets satisfying the minimum support count of 2. Thus all the candidates in C_1, except for {6}, are in L_1.

L_1 itemset	Support count
{1}	6
{2}	7
{3}	6
{4}	2
{5}	2

To discover the set of frequent 2-itemsets, L_2, the algorithm joins

L_1 with itself to generate a candidate set of 2-itemsets, C_2. Note that no candidates are removed from C_2 during the pruning step since each subset of the candidates is also frequent.

C_2 itemset
$\{1, 2\}$
$\{1, 3\}$
$\{1, 4\}$
$\{1, 5\}$
$\{2, 3\}$
$\{2, 4\}$
$\{2, 5\}$
$\{3, 4\}$
$\{3, 5\}$
$\{4, 5\}$

Next the transactions in D are scanned and the support count of each candidate itemset in C_2 is accumulated.

C_2 itemset	Support count
$\{1, 2\}$	4
$\{1, 3\}$	4
$\{1, 4\}$	1
$\{1, 5\}$	2
$\{2, 3\}$	4
$\{2, 4\}$	2
$\{2, 5\}$	2
$\{3, 4\}$	0
$\{3, 5\}$	1
$\{4, 5\}$	0

The set of frequent 2-itemsets, L_2, is then determined, consisting of those candidate 2-itemsets in C_2 having minimum support.

L_2 itemset	Support count
$\{1,2\}$	4
$\{1,3\}$	4
$\{1,5\}$	2
$\{2,3\}$	4
$\{2,4\}$	2
$\{2,5\}$	2

Next, C_3 is generated by joining L_2 with itself. The result is $C_3 = \{\{1,2,3\},\{1,2,5\},\{1,3,5\},\{2,3,4\},\{2,3,5\},\{2,4,5\}\}$. C_3 is then pruned using the Apriori property: All nonempty subsets of a frequent itemset must also be frequent. From the way each candidate of C_3 is formed, it is clear that all we need to check is the subset obtained from the last two members of the candidate set.

Since $\{2,3\}$ is a frequent itemset, we keep $\{1,2,3\}$ in C_3.
Since $\{2,5\}$ is a frequent itemset, we keep $\{1,2,5\}$ in C_3.
Since $\{3,5\}$ is not a frequent itemset, we remove $\{1,3,5\}$ from C_3.
Since $\{3,4\}$ is not a frequent itemset, we remove $\{2,3,4\}$ from C_3.
Since $\{3,5\}$ is not a frequent itemset, we remove $\{2,3,5\}$ from C_3.

Since $\{4, 5\}$ is not a frequent itemset, we remove $\{2, 4, 5\}$ from C_3.

Therefore after pruning, C_3 is given by:

C_3 itemset
$\{1, 2, 3\}$
$\{1, 2, 5\}$

The transactions in D are scanned to determine L_3, consisting of those candidates 3-itemsets in C_3 having at least minimum support.

C_3 itemset	Support count
$\{1, 2, 3\}$	2
$\{1, 2, 5\}$	2

Since both 3-itemsets in C_3 have the least minimum support, L_3 is therefore given by:

L_3 itemset	Support count
$\{1, 2, 3\}$	2
$\{1, 2, 5\}$	2

Finally L_3 is joined with itself to generate a candidate set of 4-itemsets, C_4. This results in a single itemset $\{1, 2, 3, 5\}$. However this itemset is pruned since its subset $\{3, 5\}$ is not frequent. Thus, $C_4 = \phi$, and the algorithm terminates, having found all of the frequent itemsets.

4. Association Rules

The second phase involves obtaining association rules based on the frequent itemsets found in the first phase of the algorithm. [4] Association rules can be generated as follows:

1. For each frequent itemset f, generate all its nonempty subsets.
2. For every nonempty subset g of f, output the rule

$$g \Rightarrow (f - g)$$

if

$$\frac{support\_count(f)}{support\_count(g)} \geq min\_conf.$$

4.1. Example

Let us consider our transaction database D. We want to find all the association rules that can be obtained from the frequent itemset $f = \{1, 2, 5\}$. The nonempty subsets of f are $\{1, 2\}$, $\{1, 5\}$, $\{2, 5\}$, $\{1\}$, $\{2\}$, $\{5\}$. The resulting association rules are as shown below, each listed with its confidence:

$\{1, 2\} \Rightarrow \{5\}$, $confidence = 2/4 = 50\%$
$\{1, 5\} \Rightarrow \{2\}$, $confidence = 2/2 = 100\%$
$\{2, 5\} \Rightarrow \{1\}$, $confidence = 2/2 = 100\%$
$\{1\} \Rightarrow \{2, 5\}$, $confidence = 2/6 = 33\%$
$\{2\} \Rightarrow \{1, 5\}$, $confidence = 2/7 = 29\%$
$\{5\} \Rightarrow \{1, 2\}$, $confidence = 2/2 = 100\%$

Since the minimum confidence threshold is 70%, only the second, third, and last rules above are strong.

4.2. Lift Ratio

A high value of confidence suggests a strong association rule. However this can be deceptive because if the antecedent and/or the consequent have a high support, we can have a high value for confidence even when they are independent!

A better measure to judge the strength of an association rule is to compare the confidence of the rule with the benchmark value where we assume that the occurrence of the consequent itemset in a transaction is independent of the occurrence of the antecedent for each rule. We can compute this benchmark from the frequency counts of the frequent itemsets. The benchmark confidence value for a rule is the support for the consequent divided by the number of transactions in the database. This enables us to compute the **lift ratio** of a rule. The lift ratio is the confidence of the rule divided by the confidence assuming independence of consequent from antecedent. A lift ratio greater than 1.0 suggests that there is some usefulness to the rule. The larger the lift ratio, the greater is the strength of the association.

All the association rules that can be derived from the frequent

itemsets obtained for our example are given below. The antecedent, A, and consequent, C, parts of the associate rules are given in columns 1 an 2, respectively. Columns 3, 4, and 5 contain the support counts for A, C, and $A \cup C$.

The confidences of the association rules,

$$c(A \Rightarrow C) = P(C|A) = \frac{s(A \cup C)}{s(A)}.$$

can be obtained by dividing the result in column 5 by those in column 3, are shown in column 6. Those with confidence values higher than the minimum confidence threshold of 60% are displayed in red.

Column 7 contains the benchmark confidence values, conf.$'$, computed under the assumption that A and C are totally uncorrelated. Therefore we have

$$\text{conf.}'(A \Rightarrow C) = P(C) = \frac{s(C)}{|D|},$$

which can be computed by dividing the results in column 4 by the total number of transactions in D (i.e. $|D| = 9$).

The lift ratios are then given by

$$\text{lift ratio} = \frac{\text{conf.}}{\text{conf.}'}.$$

The highest lift ratio turns out to be 9/4, and there are 2 such cases (displayed in red).

Only the association rule $\{5\} \Rightarrow \{1, 2\}$ has a confidence above the minimum confidence threshold and the highest lift ratio.

The association rule $\{1, 2\} \Rightarrow \{5\}$ does not meet the minimum confidence threshold requirement and yet it has the highest lift ratio.

The association rule $\{2, 5\} \Rightarrow \{1\}$ has a confidence above the minimum confidence threshold and the next highest lift ratio. One may debate whether it is really meaningful. The same applies to the association rule $\{1, 5\} \Rightarrow \{2\}$.

A	C	sc(A)	sc(C)	sc($A \cup C$)	conf.	conf.′	lift rati
$\{1,2\}$	$\{5\}$	4	2	2	2/4	2/9	9/4
$\{1,5\}$	$\{2\}$	2	7	2	2/2	7/9	9/7
$\{2,5\}$	$\{1\}$	2	6	2	2/2	6/9	9/6
$\{1\}$	$\{2,5\}$	6	2	2	2/6	2/9	9/6
$\{2\}$	$\{1,5\}$	7	2	2	2/7	2/9	9/7
$\{5\}$	$\{1,2\}$	2	4	2	2/2	4/9	9/4
$\{1,2\}$	$\{3\}$	4	6	2	2/4	6/9	9/12
$\{1,3\}$	$\{2\}$	4	7	2	2/4	7/9	9/14
$\{2,3\}$	$\{1\}$	4	6	2	2/4	6/9	9/12
$\{1\}$	$\{2,3\}$	6	4	2	2/6	4/9	3/4
$\{2\}$	$\{1,3\}$	7	4	2	2/7	4/9	9/14
$\{3\}$	$\{1,2\}$	6	4	2	2/6	4/9	3/4
$\{1\}$	$\{2\}$	6	7	4	4/6	7/9	6/7
$\{1\}$	$\{3\}$	6	6	4	4/6	6/9	1
$\{1\}$	$\{5\}$	6	2	2	2/6	2/9	3/2
$\{2\}$	$\{3\}$	7	6	4	4/7	6/9	6/7
$\{2\}$	$\{4\}$	7	2	2	2/7	2/9	9/7
$\{2\}$	$\{5\}$	7	2	2	2/7	2/9	9/7

5. Final Remarks

One major shortcoming of association rules data-mining is that the support-confidence framework often generates too many rules. However, there have been a number of modifications and extensions to improve the Apriori algorithm.[1]

References

[1] Jiawei Han and Micheline Kamber, *Data Mining: Concepts and Techniques*, Elsevier 2006, ISBN 1558609016. This lecture notes is based on materials in chapter 5. 3, 27

[2] R. Agrawal and R. Srikant, "Fast algorithm for mining association rules", *Proc. 1994 Int. Conf. Very Large Data Bases (VLDB'94)*, pages 487-499, Santiago, Chile, Sept. 1994. 12

[3] H. Mannila, H. Toivonen, and A. J. Verkamo, "Efficient algorithms for discovering association rules", *Proc. AAAI'94 Workshop Knowledge Discovery in Databases (KDD'94)*, pages 181-192, Seattle, WA, July, 1994. 12

[4] R. Agrawal and R. Srikant, "Fast algorithm for mining association rules in large databases", *Research Report RJ 9839*, IBM Almaden Research Center, San Jose, CA, June, 1994.

21