Lecture 16

Discovering Association Rules

in

Transaction Databases

What are Association Rules?

The availability of detailed information on customer transactions has led to the development of techniques that automatically look for associations between items that are stored in the database. An example is data collected using bar-code scanners in supermarkets. Such 'market basket' databases consist of a large number of transaction records. Each record lists all 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. They could use this data for store layouts to place items optimally with respect to each other, they could use such information for cross-selling, for promotions, for catalog design and to identify customer segments based on buying patterns. 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 addition to the antecedent (the "if" part) and the consequent (the "then" part) an association rule has two numbers that express the degree of uncertainty about the rule. In association analysis the antecedent and consequent are sets of items (called itemsets) that are disjoint (do not have any items in common).

The first number is called the *support* for the rule. The support is simply the number of transactions that include all items in the antecedent and consequent parts of the rule. (The support is sometimes expressed as a percentage of the total number of records in the database.)

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 of 800 transactions (alternatively 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.

Example 1 (Han and Kamber)

The manager of the AllElectronics retail store would like to know what items sell together. He has a database of transactions as shown below:

Transaction ID	Item Codes			
1	1	2	5	
2	2	4		
3	2	3		
4	1	2	4	
5	1	3		
6	2	3		
7	1	3		
8	1	2	3	5
9	1	2	3	

There are 9 transactions. Each transaction is a record of the items bought together in that transaction. Transaction 1 is a point-of-sale purchase of items 1, 2, and 5. Transaction 2 is a joint purchase of items 2 and 4, etc. Suppose that we want association rules between items for this database that have a support count of at least 2 (equivalent to a percentage support of 2/9=22%). By enumeration we can see that only the following itemsets have a count of at least 2:

{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.

Notice that once we have created a list of all itemsets that have the required support, we can deduce the rules that meet the desired confidence ratio by examining all subsets of each itemset in the list. Since any subset of a set must occur at least as frequently as the set, each subset will also be in the list. It is then straightforward to compute the confidence as the ratio of the support for the itemset to the support for each subset of the itemset. We retain the corresponding association rule only if it exceeds the desired cut-off value for confidence. For example, from the itemset {1,2,5} we get the following association rules:

> {1, 2} => {5} with confidence = support count of {1, 2, 5} divided by support count of {1, 2} = 2/4 = 50%;
> {1, 5} => {2} with confidence = support count of {1, 2, 5} divided by support count of {1, 5} = 2/2 = 100%;
> {2, 5} => {1} with confidence = support count of {1, 2, 5} divided by support count of {2, 5} = 2/2 = 100%;
> {1} => {2, 5} with confidence = support count of {1, 2, 5} divided by support count of {1} = 2/6 = 33%;
> {2} => {1,5} with confidence = support count of {1, 2, 5} divided by support count of {2} = 2/7 = 29%;
> {5} => {1,2} with confidence = support count of {1, 2, 5} divided by support count of {5} = 2/2 = 100%.

If the desired confidence cut-off was 70%, we would report only the second, third, and last rules.

We can see from the above that the problem of generating all association rules that meet stipulated support and confidence requirements can be decomposed into two stages. First we find all itemsets with the requisite support (these are called frequent or ‘large’ itemsets) ; and then we generate, from each itemset so identified, association rules that meet the confidence requirement. For most association analysis data, the computational challenge is the first stage.

The Apriori Algorithm Although several algorithms have been proposed for generating association rules, the classic algorithm is the Apriori algorithm of Agrawal and Srikant. The key idea of the algorithm is to begin by generating frequent itemsets with just one item (1-itemsets) and to recursively generate frequent itemsets with 2 items, then frequent 3-itemsets and so on until we

have generated frequent itemsets of all sizes. Without loss of generality we will denote items by unique, consecutive (positive) integers and that the items in each itemset are in increasing order of this item number. The example above illustrates this notation. When we refer to an item in a computation we actually mean this item number.

It is easy to generate frequent 1-itemsets. All we need to do is to count, for each item, how many transactions in the database include the item. These transaction counts are the supports for the 1-itemsets. We drop 1-itemsets that have support below the desired cut-off value to create a list of the frequent 1-itemsets.

The general procedure to obtain k-itemsets from (k-1)-itemsets for k = 2, 3, … , is as follows. Create a candidate list of k-itemsets by performing a join operation on pairs of (k-1)-itemsets in the list. The join is over the first (k-2) items, i.e. a pair is combined if the first (k-2) items are the same in both members of the pair. If this condition is met the join of pair is a k-itemset that contains the common first (k-2) items and the two items that are not in common, one from each member of the pair. All frequent k-itemsets must be in this candidate list since every subset of size (k-1) of a frequent k-itemset must be a frequent (k-1) itemset. However, some k-itemsets in the candidate list may not be frequent k-itemsets. We need to delete these to create the list of frequent k-itemsets. To identify the k-itemsets that are not frequent we examine all subsets of size (k-1) of each candidate k-itemset. Notice that we need examine only (k-1)-itemsets that contain the last two items of the candidate k-itemset (Why?). If any one of these subsets of size (k-1) is not present in the frequent (k-1) itemset list, we know that the candidate k-itemset cannot be a frequent itemset. We delete such k-itemsets from the candidate list. Proceeding in this manner with every itemset in the candidate list we are assured that at the end of our scan the k-itemset candidate list will have been pruned to become the list of frequent k-itemsets. We repeat the procedure recursively by incrementing k. We stop only when the candidate list is empty.

A critical aspect for efficiency in this algorithm is the data structure of the candidate and frequent itemset lists. Hash trees were used in the original version but there have been several proposals to improve on this structure.

There are also other algorithms that can be faster than the Apriori algorithm in practice.

Let us examine the output from an application of this algorithm to a small randomly generated database of 50 records shown in Example 2.

Example 2

Tr#	Items				
1	8				
2	3	4	8		
3	8				
4	3	9			
5	9				
6	1	8			
7	6	9			
8	3	5	7	9	
9	8				
10	8				
11	1	7	9		
12	1	4	5	8	9
13	5	7	9		
14	6	7	8		
15	3	7	9		
16	1	4	9		
17	6	7	8		
18	8				
19	8				
20	9				
21	2	5	6	8	
22	4	6	9		
23	4	9			
24	8	9			
25	6	8			
26	1	6	8		
27	5	8			
28	4	8	9		
29	9				
30	8				
31	1	5	8		
32	3	6	9		
33	7	9			
34	7	8	9		
35	3	4	6	8	
36	1	4	8		
37	4	7	8		
38	8	9			
39	4	5	7	9	
40	2	8	9		

41	2	5	9		
42	1	2	7	9	
43	5	8			
44	1	7	8		
45	8				
46	2	7	9		
47	4	6	9		
48	9				
49	9				
50	6	7	8		

Association Rules Output

Input Data: $A$5:$E$54

Min. Support: 2 = 4%

Min. Conf. %: 70

Rule #	Confidence %	Antecedent (a)		Consequent (c)	Support (a)	Support (c)	Support (a U c)	Confidence If pr(c|a) = pr(c) %	Lift Ratio (conf/prev.col.)
1	80	2	=>	9	5	27	4	54	1.5
2	100	5, 7	=>	9	3	27	3	54	1.9
3	100	6, 7	=>	8	3	29	3	58	1.7
4	100	1, 5	=>	8	2	29	2	58	1.7
5	100	2, 7	=>	9	2	27	2	54	1.9
6	100	3, 8	=>	4	2	11	2	22	4.5
7	100	3, 4	=>	8	2	29	2	58	1.7
8	100	3, 7	=>	9	2	27	2	54	1.9
9	100	4, 5	=>	9	2	27	2	54	1.9

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 occurance of the antecedant 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. (What does a ratio less than 1.00 mean? Can it be useful to know such rules?)

In our example the lift ratios highlight Rule 6 as most interesting in that it suggests purchase of item 4 is almost 5 times as likely when items 3 and 8 are purchased than if item 4 was not associated with the itemset {3,8}.

Shortcomings

Association rules have not been as useful in practice as one would have hoped. One major shortcoming is that the support confidence framework often generates too many rules. Another is that often most of them are obvious. Insights such as the celebrated "on Friday evenings diapers and beers are bought together" story are not as common as might be expected. There is need for skill in association analysis and it seems likely, as some researchers have argued, that a more rigorous statistical discipline to cope with rule proliferation would be beneficial.

Extensions

The general approach of association analysis utilizing support and confidence concepts has been extended to sequences where one is looking for patterns that evolve in time. The computation problems are even more formidable, but there have been several successful applications.