**Chapter Five**

**Apriori Algorithm**

**5.1 : Preface**

Apriori is a classic algorithm for frequent item set mining and association rule learning over transactional databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database. [8]

**5.2 : Motivation and Structure of Association Rules**

In several areas of application, the systematic collection of data gives rise to massive lists of transactions that lend themselves to analysis through association rules in order to identify possible recurrences in the data. [1]

5.2.1 : Market Basket Analysis

When a customer makes a purchase at a point of sale and receives a receipt for her payment, this transaction is recorded by the information system of the retailer that manages the point of sale. For each transaction recorded, a list of purchased items is stored along with the price, time and place of the transaction. These transactions are gathered into a massive dataset, at least for large retailers, which can be exploited to perform a data mining analysis aimed at identifying recurrent rules that relate the purchase of a product, or group of products, to the purchase of another product, or group of products. For example, it might be possible to obtain association rules of the form ‘a customer buying breakfast cereals will also buy milk with a probability of 0.68’. The association rules for market basket analysis can be quite useful for marketing managers in planning promotional initiatives or defining the assortment and location of products on the shelves. [1]

5.2.2 : Web Mining

Within web mining analyses it is particularly useful to understand the pattern of navigation paths and the frequency with which combinations of web pages are visited by a given individual during a single session or consecutive sessions. In this case too the list of pages visited during a session is recorded as a transaction, possibly matched with a sequence number and the time of visit. It is therefore interesting to identify regular patterns possibly hidden in the data that allow the association of one or more pages being viewed with visits to other pages. The rules may assume a form such as ‘if an individual visits the site *timesonline.co.uk* then within a week she will also visit the site *economist.com* with a probability of 0.87’. Association rules of this kind may influence the structure of the links between pages, in order to ease the navigation and to recommend specific navigation paths, or to place advertisement banners and other promotional messages. [1]

5.2.3 : Purchases with a Credit Card

Association rules are also used to analyze the purchases made by credit card holders in order to direct future promotions. In this case each transaction consists of the purchases and the payments made by a credit card holder. Notice that products and services that can potentially be accessed by the credit card owner are virtually infinite in this situation. [1]

5.2.4 : Fraud Detection

In fraud identification, transactions consist of incident reports and applications for compensation for the damage suffered. The existence of special combinations may reveal potentially fraudulent behaviors and therefore justify an in-depth investigation on the part of the insurance company. [1]



**Figure (5.1): Recommendations Based on Association Rules [3]**

**5.3 : Mining Frequent Patterns and Association Rules**

One of the most popular data mining approaches is to find frequent itemsets from a transaction dataset and derive association rules. The problem is formally stated as follows. [2]

Let *I* = {*i*1*, i*2*, . . . , im*} be a set of items. Let *D* be a set of transactions, where each transaction *t* is a set of items such that *t* ⊆ *I*. Each transaction has a unique identifier, called its *TID*. A transaction *t* contains *X*, a set of some items in *I*, if *X* ⊆ *t*. An association rule is an implication of the form *X* ⇒ *Y* , where *X* ⊂ *I, Y* ⊂ *I*, and *X* ∩ *Y* = ∅. The rule *X* ⇒ *Y* holds in *D* with confidence *c* (0 ≤ *c* ≤ 1) if the fraction of transactions that also contain *Y* in those which contain *X* in*D* is *c*. The rule *X* ⇒ *Y* (and equivalently *X* ∪*Y* ) has support1 *s* (0 ≤ *s* ≤ 1) in*D* if the fraction of transactions in *D* that contain *X* ∪ *Y* is *s*. Given a set of transactions *D*, the problem of mining association rules is to generate all association rules that have support and confidence no less than the user-specified minimum support (called *minsup*) and minimum confidence (called *minconf*), respectively. [2]

Finding frequent2 itemsets (itemsets with support no less than *minsup*) is not trivial because of the computational complexity due to combinatorial explosion. Once frequent itemsets are obtained, it is straightforward to generate association rules with confidence no less than *minconf*. [2]

**5.4 : Apriori**

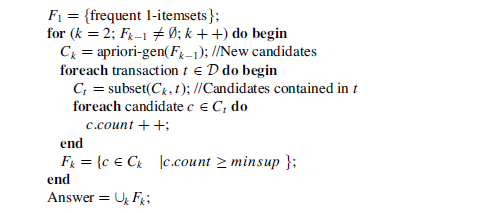
Apriori is an algorithm to find all sets of items (itemsets) that have support no less than *minsup*. The support for an itemset is the ratio of the number of transactions that contain the itemset to the total number of transactions. Itemsets that satisfy minimum support constraint are called *frequent itemsets*. Apriori is characterized as a level-wise complete search (breadth first search) algorithm using anti-monotonicity property of itemsets: “If an itemset is not frequent, any of its superset is never frequent,” which is also called the *downward closure property*. [2]

The algorithm makes multiple passes over the data. In the first pass, the support of individual items is counted and frequent items are determined. In each subsequent pass, a seed set of itemsets found to be frequent in the previous pass is used for generating new potentially frequent itemsets, called *candidate itemsets*, and their actual support is counted during the pass over the data. At the end of the pass, those satisfying minimum support constraint are collected, that is, frequent itemsets are determined, and they become the seed for the next pass. This process is repeated until no new frequent itemsets are found.[2]

By convention, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order. The number of items in an itemset is called its *size* and an itemset of size *k* is called a *k*-itemset. Let the set of frequent itemsets of size *k* be *Fk* and their candidates be *Ck* . Both *Fk* and *Ck* maintain a field, support count. [2]

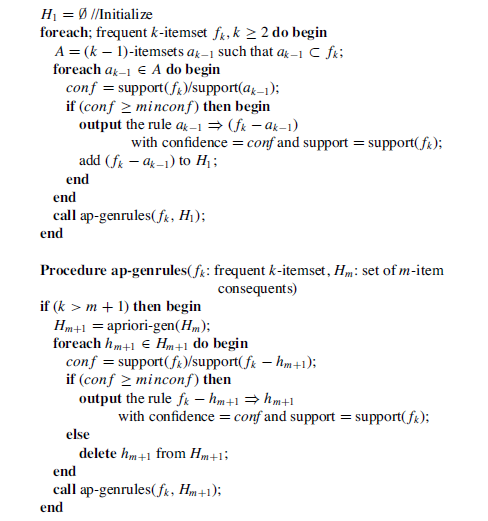
Apriori algorithm is given in Algorithm 5.1. The first pass simply counts item occurrences to determine the frequent 1-itemsets. A subsequent pass consists of two phases. First, the frequent itemsets *Fk*−1 found in the (*k* − 1)-th pass are used to generate the candidate itemsets *Ck* using the apriori-gen function. Next, the database is scanned and the support of candidates in *Ck* is counted. The subset function is used for this counting. [2]

The apriori-gen function takes as argument *Fk*−1, the set of all frequent (*k* − 1)- itemsets, and returns a superset of the set of all frequent *k*-itemsets. Then, in the prune step, all the itemsets *c* ∈ *Ck* for which some (*k* − 1)-subset is not in *Fk*−1 are deleted. [2]



**Figure (5.2): Apriori Algorithm [2]**

The remaining task is to generate the desired association rules from the frequent itemsets. A straightforward algorithm for this task is as follows. To generate rules, all nonempty subsets of every frequent itemset *f* are enumerated and for every such subset *a*, a rule of the form *a* ⇒ ( *f* − *a*) is generated if the ratio of support( *f* ) to support(*a*) is at least *minconf*. Here, note that the confidence of the rule ˆ*a* ⇒ ( *f* − ˆ*a*) cannot be larger than the confidence of *a* ⇒ ( *f* −*a*) for any ˆ*a* ⊂ *a*. This in turn means that for a rule ( *f* −*a*) ⇒ *a* to hold, all rules of the form ( *f* − ˆ*a*) ⇒ ˆ*a* must hold. Using this property, the algorithm to generate association rules is given in Algorithm 5.2 . [2]

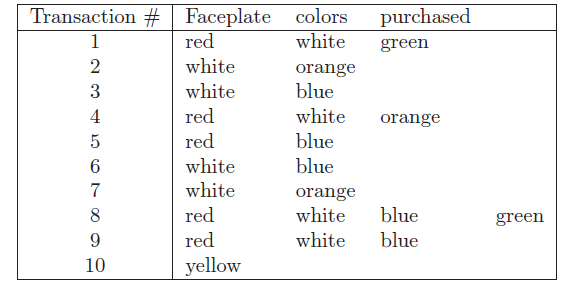


**Figure (5.3): Association Rule Generation Algorithm [2]**

Apriori achieves good performance by reducing the size of candidate sets. However, in situations with very many frequent itemsets or very low minimum support, it still suffers from the cost of generating a huge number of candidate sets and scanning the database repeatedly to check a large set of candidate itemsets. [2]

**5.5 : Example - Synthetic Data on Purchases of Phone Faceplates**

A store that sells accessories for cellular phones runs a promotion on faceplates. Customers who purchase multiple faceplates from a choice of six different colors get a discount. The store managers, who would like to know what colors of faceplates customers are likely to purchase together, collected the following transaction database: [3]



**Figure (5.4): Transactions for Purchases of Different Colored Cellular Phone Faceplates [3]**

**5.6 : Generating Candidate Rules**

The idea behind associations rules is to examine all the possible rules between items in an "if-then" format, and select only those that are most likely to be indicators of true dependence. [3]

We use the term *antecedent* to describe the "if" part, and *consequent* to describe the "then" part. In association analysis the antecedent and consequent are sets of items (called *item sets*) that are disjoint (do not have any items in common). [3]

Returning to the phone faceplate purchase example, one example of a possible rule is "if red then white", meaning if a red faceplate is purchased, then a white one is too. Here the antecedent is "red" and the consequent is "white". The antecedent and consequent each contain a single item in this case. Another possible rule is "if red and white, then green". Here the antecedent includes the item set {red,white} and the consequent is {green}.[3]

The first step in affinity analysis is to generate all the rules that would be candidates for indicating associations between items. Ideally, we might want to look at all possible combinations of items in a database with *p* distinct items (in the phone faceplate example *p* = 6). This means finding all combinations of single items, pairs of items, triplets of items, etc. in the transactions database. However, generating all these combinations requires long computation time that grows exponentially in *k*. A practical solution is to consider only combinations that occur with higher frequency in the database. These are called *frequent item sets*. [3]

Determining what consists of a frequent item set is related to the concept of *support*. The support of a rule is simply the number of transactions that include both the antecedent and consequent item sets. It is called a support because it measures the degree to which the data "support" the validity of the rule. The support is sometimes expressed as a percentage of the total number of records in the database. For example, the support for the item set {red,white} in the phone faceplate example is 4 (100 × 4/10 = 40%). What constitutes a frequent item set is therefore defined as an item set that has a support that exceeds a selected minimum support, determined by the user.[3]

Several algorithms have been proposed for generating frequent item sets, but the classic algorithm is the Apriori algorithm of Agrawal and Srikant (1993). The key idea of the algorithm is to begin by generating frequent item sets with just one item (1-item sets) and to recursively generate frequent item sets with 2 items, then with 3 items, and so on until we have generated frequent item sets of all sizes. [3]

It is easy to generate frequent 1-item sets. 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-item sets. We drop 1-item sets that have support below the desired minimum support to create a list of the frequent 1-item sets. [3]

To generate frequent 2-item sets, we use the frequent 1-item sets. The reasoning is that if a certain 1-item set did not exceed the minimum support, then any larger size item set that includes it will not exceed the minimum support. In general, generating *k*-item sets uses the frequent *k-* 1-item sets that were generated in the previous step. Each step requires a single run through thedatabase, and therefore the Apriori algorithm is very fast even for a large number of unique itemsin a database. [3]

**5.7 : Selecting Strong Rules**

From the abundance of rules generated, the goal is to find only the rules that indicate a strong dependence between the antecedent and consequent item sets. To measure the strength of association implied by a rule, we use the measures of *confidence* and *lift ratio* , as described below.[3]

5.7.1 : Support and Confidence

In addition to support, which we described earlier, there is another measure that expresses the degree of uncertainty about the "if-then" rule. This is known as the *confidence* of the rule. This measure compares the co-occurrence of the antecedent and consequent item sets in the database.

To the occurrence of the antecedent item sets. Confidence is defined as the ratio of the number of transactions that include all antecedent and consequent item sets (namely, the support) to the number of transactions that include all the antecedent item sets:

C:\Users\MHT~1.M\AppData\Local\Temp\msohtmlclip1\01\clip_image001.png

For example, suppose a supermarket database has 100,000 point-of-sale transactions. Of these transactions, 2000 include both orange juice and (over-the-counter) flu medication, and 800 of these include soup purchases. The association rule "IF orange juice and flu medication are purchased THEN soup is purchased on the same trip" has a support of 800 transactions (alternatively 0.8% = 800/100,000) and a confidence of 40% (= 800/2000).

To see the relationship between support and confidence, let us think about what each is measuring (estimating). One way to think of support is that it is the (estimated) probability that a randomly selected transaction from the database will contain all items in the antecedent and the consequent:

C:\Users\MHT~1.M\AppData\Local\Temp\msohtmlclip1\01\clip_image001.png

In comparison, the confidence is the (estimated) *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:

C:\Users\MHT~1.M\AppData\Local\Temp\msohtmlclip1\01\clip_image001.png

A high value of confidence suggests a strong association rule (in which we are highly confident). 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! For example, if nearly all customers buy bananas and nearly all customers buy ice cream, then the confidence level will be high regardless of whether there is an association between the items. [3]

5.7.2 : Lift Ratio

A better way to judge the strength of an association rule is to compare the confidence of the rule with a benchmark value, where we assume that the occurrence of the consequent item set in a transaction is independent of the occurrence of the antecedent for each rule. In other words, if the antecedent and consequent item sets are independent, what confidence values would we expect to see? Under independence, the support would be:

C:\Users\MHT~1.M\AppData\Local\Temp\msohtmlclip1\01\clip_image001.png

and the benchmark confidence would be:

C:\Users\MHT~1.M\AppData\Local\Temp\msohtmlclip1\01\clip_image001.png

The estimate of this benchmark from the data, called the *benchmark confidence value* for a rule is computed by:

C:\Users\MHT~1.M\AppData\Local\Temp\msohtmlclip1\01\clip_image001.png

We compare the confidence to the benchmark confidence by looking at their ratio: this is called 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.

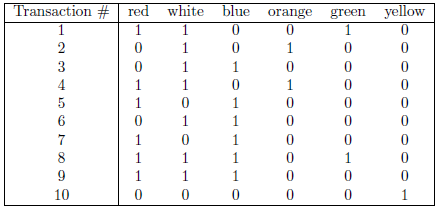
C:\Users\MHT~1.M\AppData\Local\Temp\msohtmlclip1\01\clip_image001.png

A lift ratio greater than 1.0 suggests that there is some usefulness to the rule. In other words, the level of association between the antecedent and consequent item sets is higher than would be expected if they were independent. The larger the lift ratio, the greater the strength of the association. [3]

To illustrate the computation of support, confidence, and lift ratio for the cellular phone face- plates example, we introduce a presentation of the data better suited to this purpose. [3]

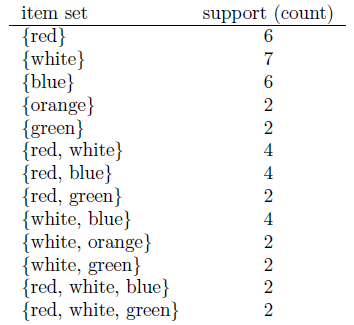
**5.8 : Data Format**

Transaction data are usually displayed in one of two formats: a list of items purchased (each row representing a transaction), or a binary matrix in which columns are items, rows again represent transactions, and each cell has either a "1" or a "0," indicating the presence or absence of an item in the transaction. For example, Figure 5.4 displays the data for the cellular faceplate purchases in item list format. We can translate these into a binary matrix format: [3]



**Figure (5.5): Transactions for purchases in binary matrix format [3]**

Now, 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 2/10=20%). In other words, rules based on items that were purchased together in at least 20% of the transactions. By enumeration we can see that only the following item sets have a count of at least 2: [3]



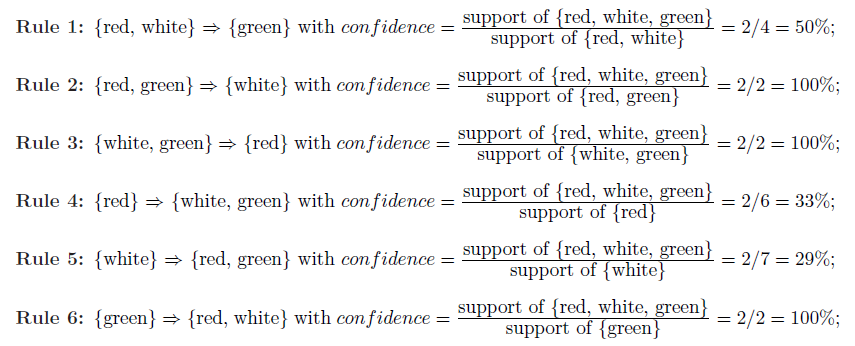
**Figure (5.6): Item sets have a count of at least 2 [3]**

The first item set {red} has a support of 6, because 6 of the transactions included a red faceplate. Similarly the last item set {red, white, green} has a support of 2, because only 2 transactions included red, white, and green faceplates. [3]

**5.9 : The Process of Rule Selection**

The process of selecting strong rules is based on generating all association rules that meet stipulated support and confidence requirements. This is done in two stages. The first stage, described in section 5.5 consists of finding all "frequent" item sets, those item sets that have a requisite support. In the second stage we generate, from the frequent item sets, association rules that meet a confidence requirement. The first step is aimed at removing item combinations that are rare in the database. The second stage then filters the remaining rules and selects only those with high confidence. For most association analysis data, the computational challenge is the first stage, as described in discussion of the Apriori algorithm. [3]

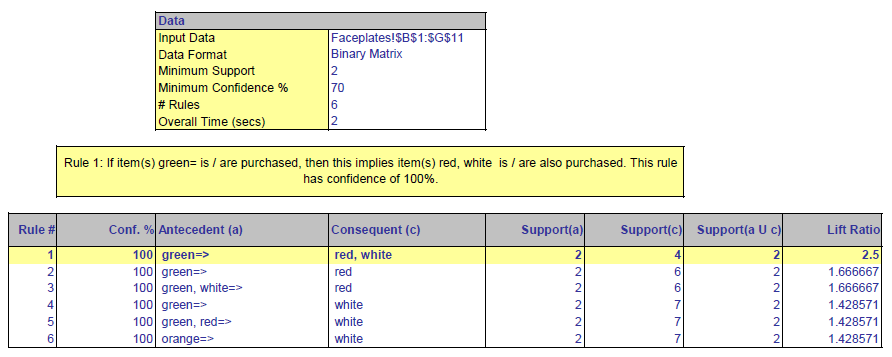
The computation of confidence in the second stage is simple. Since any subset (e.g., {red} in the phone faceplate example) must occur at least as frequently as the set it belongs to (e.g. {red, white}), each subset will also be in the list. It is then straightforward to compute the confidence as the ratio of the support for the item set to the support for each subset of the item set. We retain the corresponding association rule only if it exceeds the desired cutoff value for confidence. For example, from the item set {red,white,green} in the phone faceplate purchases we get the following association rules: [3]



**Figure (5.7): Association Rules from the item set {red,white,green} [3]**

If the desired minimum confidence is 70%, we would report only the second, third, and last rules. [3]

In XLMiner (a mining tool), we can generate association rules by specifying the minimum support count (2) and minimum confidence level percentage (70%). Figure 5.8 shows the output. Note that here we consider all possible item sets, not just {red, white, green} as above. [3]



**Figure (5.8): Association Rules for Phone Faceplates Transactions- XLMiner Output [3]**

The output includes information on the support of the antecedent, the support of the consequent, and the support of the combined set (denoted by *Support*(*a* ∪ *c*)). It also gives the confidence of the rule (in %) and the lift ratio. In addition, XLMiner has an "interpreter" that translates the rule from a certain row into English. In the snapshot shown in Figure 5.8, the first rule is highlighted (by clicking), and the corresponding English rule appears in the yellow box: [3]

Rule 1: If item(s) green is/are purchased, then this implies item(s) red, white is/are also purchased. This rule has confidence of 100%