COMP 4106 Project Final Report

Ankush Varshneya – 100853074

Association Rule Mining using Apriori Algorithm

### Motivation

The medication behind this project is that I wanted to build a mining tool that would be used to perform market basket analysis to build a smart platform for suggesting items in a simulated shopping environment.

### Problem Domain and Algorithm used

Association rule mining is an important data mining model studied extensively by the database and data mining community. This is used on data from a transaction database to determine association rules highlight general trends in the database; it is commonly used for market basket analysis to find how items purchased by customers are related to one another.

So for example if we analyse all shopping baskets for a particular shop via the cashiers, then we can use Association rule mining to learn association rules about what people buy what based other items in their shopping cart. This is beneficial for stores as they can strategically place items together so as to make the potential shopper want to buy a set of related items together rather than one item. For example if we find the rule that people that buy beer tend to buy diapers, then we can place beer and diaper together in the store so as to make people more inclined to buy both of the items as they are together. The can also use it to suggest items to a shopper.

#### The model:

We model the problem as I the set of all items offered by the store and T as the set of all transactions of items taken out by all customers. Further, we define a transaction ti which is all the non-repetitive items in a shopping cart. So we have the set of all transactions, all the shopping carts of all customers at the store, as T = {t1, t2, … , tm}, each transaction is also a subset of “I”. Using these transactions we can mine for patterns such as what items are usually bough together. So this means we want to extract rules in the form of X ==> Y, where X and Y are both subsets of “I” and X and Y are mutually exclusive, i.e. When items from X are present in a transaction, then also items from Y are present in it. The problem I am trying to solve is how we can derive such rules. This is called association rule mining, and we can use algorithms such as the Apriori algorithm to find such patterns.

#### Key Concepts:

We define support (X ==> Y) as Probability (X or Y being in any transaction) which is Number of times X OR Y are in T / M (The total number of transactions)

We define confidence (X ==> Y) as Probability (X or Y being in any transaction) / Probability (X being in any transaction) which is support(X OR Y) / support (X)

#### How it works:

Given a set of transactions we build what is known as a candidate set. The ithcandidate item set contains all subset of length i from the item set. We calculate the 1st candidate item set along with the support of each of the contained subsets. If the support is greater than or equal to a pre-set minimum support cut off then the candidate set is added to the ith large item set. The large item set contains all the sets that have a support greater than or equal to min support and as known as the frequent item sets.

Next you iteratively calculate the candidate item followed by the large item set for i=2, you keep repeating this process increasing I by one each time until the next larger item set in empty or until i = the size of the item set – 1.

Immediately we can see a problem with the above process, in each iteration we have to go throughout the entire data set, this can makes the approach not very scalable. The solution to this is to use the Aprori algorithm. Which works of the following property:

Anti-monotonicity: if Z1 ⊆ Z2 are sets, then support (Z2) ≤ support (Z1). In particular, if Z2 is large, then also Z1 is large: Z1 ⊆ Z2 and Z2 large ==> Z1 large. This allows us to avoid exploring many candidate sets. We use this with join and prune steps mentioned below to be able to get a more scalable way to calculate large item sets.

We define Lk to be the kth large item set and Ck to be the kth candidate item set

Given the Kth large item set we can calculate the K+1th candidate item set by:

Join Step: Join Lk with Lk to produce Ck+1, this means for all sets in Lk we find all pairs of sets just that only there first k-1 elements are the same and we make a union of such sets to get a set of size k+1. Doing this for all sets in Lk gets us Ck+1.

Prune Step: Using the Anti-monotonicity property we can computer Lk. So for each set in Ck+1 we generate the subsets of size k+1, and if any of the sub sets are not larger item sets we prune, ie remove the set, and not count it as a large set.

To further speed up support calculation we can maintain a map of previously explored sets and their support.

So we have a refined process: We calculate the 1st candidate set along with the support of each of the contained subsets. If the support is greater than or equal to a pre-set minimum support cut off then the candidate set is added to the ith large item set; we repeat this then again for i=2. Then we use the 2nd large item set to systematically compute the 3rd candidate set by the join step and then we use the prune step to shorten the candidate set. For each set in the reduced candidate set, if the support is greater than or equal to a pre-set minimum support cut off then the candidate set is added to the ith large item set. The large item set contains all the sets that have a support greater than or equal to min support and as known as the frequent item sets.

Next you iteratively calculate the candidate item followed by the large item set, by the join and prune steps, increasing i by one each time until the next larger item set in empty or until i = the size of the item set – 1. Apriori achieves higher efficiency through the Apriori elimination/pruning of certain candidate item sets.

After calculating all larger / frequent item sets we calculate the association rules. For each frequent item set f, we generate an item all nonempty subsets s. For all f, s combination we generate a rule s ==> (f-s) such that s AND (f-s) = 0; we then calculate the confidence of each rule in the form of s ==> (f-s) as support (f) / support (s). All rules that have a confidence greater than or equal to min confidence (given as input) are known as the association rules.

We can now use the association rules for tasks such as market basket analysis, to suggest items to but based on current shopping cart.

### Design Choice