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 t_i 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 $T = \{t_1, t_2, \dots, t_m\}$, 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 ith candidate 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 $Z_1 \subseteq Z_2$ are sets, then support $(Z_2) \leq$ support (Z_1). In particular, if Z_2 is large, then also Z_1 is large: $Z_1 \subseteq Z_2$ and Z_2 large ==> Z_1 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 L_k to be the kth large item set and C_k 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 L_k with L_k to produce C_{k+1}, this means for all sets in L_k 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 L_k gets us C_{k+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 Choices

Program environment

The program was implemented in Java, partially due to my familiarity with it but also it allows for good logging of memory usage and calculation time for statistical purposes.

The State space

The state consisted of the transactions in dataset, all items in the dataset, the minimum confidence, the minimum support, all the large item sets, all the association rules, a toggle to turn on pruning, and a pointer to the dataset. Each item set consisted of a set of items and its support. Each association rule consisted of a left side set, a right side set, and its confidence.

Result

The results were compiled for the large dataset ~8k rows and were as follows:

Time Taken:1.038 seconds	Time Taken:0.961 seconds
Purning: Enabled	Purning: Disabled
Processed: 308 Candidate set(s)	Processed: 332 Candidate set(s)
Min Support: 50.0 %	Min Support: 50.0 %
Min Confidence: 70.0 %	Min Confidence: 70.0 %
File Selected: LargeDataSetSample1.dat	File Selected: LargeDataSetSample1.dat

The general trend was that as the support level got lower and the data set bigger, the benefits of enabling pruning made a substantial difference it terms of candidate set(s) processed.

Possible Enhancements

While the algorithm worked well, but when used with contain large sets above 8K rows and a low support and low confidence there were some performance issues that arose; the run time was over 1 minute. I hope to implement a frequent pattern tree (FP-tree) structure to speed up the process. This is done by compressing a large database into a compact, Frequent Pattern tree (FP-tree) structure. It's a divide-and-conquer methodology: decompose mining tasks into smaller ones which helps avoid candidate generation as it does sub-database test only.

References

Agrawal, R. and Srikant, R. Fast Algorithms for Mining Association Rules. http://rakesh.agrawal-family.com/papers/vldb94apriori.pdf, 1994

Wasilewska, A. APRIORI Algorithm – Lecture Notes. http://www3.cs.stonybrook.edu/~cse634/lecture_notes/07apriori.pdf, 2014

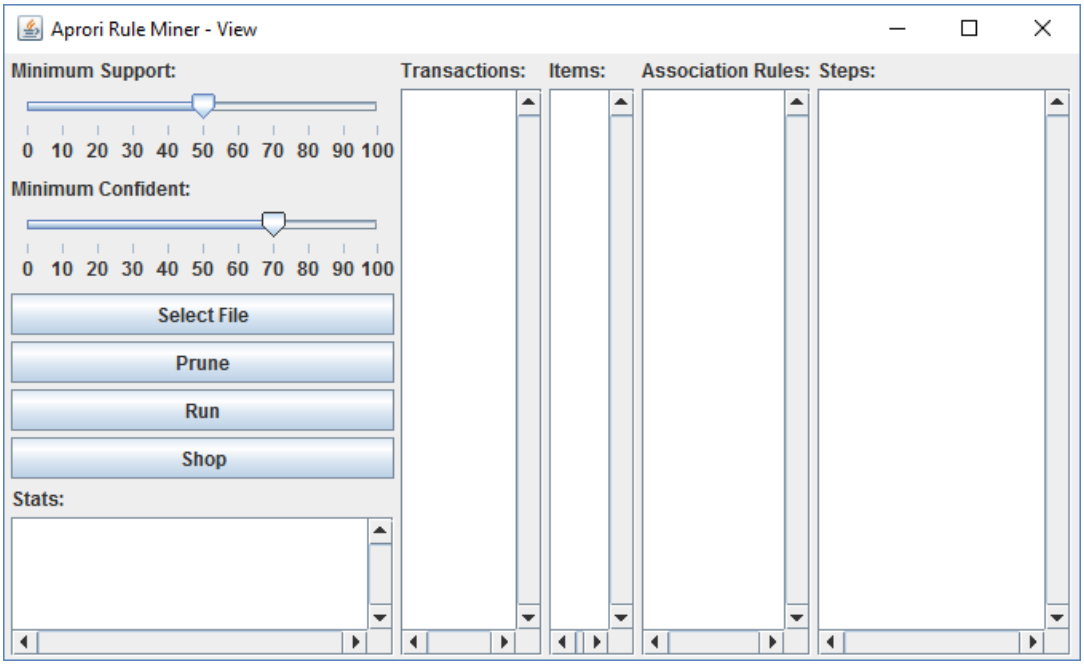
Appendix

The code can be downloaded here: https://github.com/AnkushVarshneya/Aprori

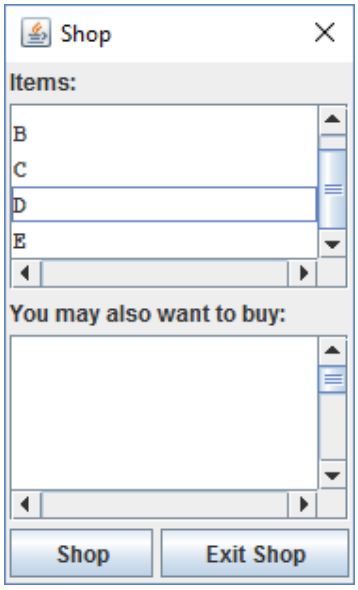
The program was coded in Eclipse and on Windows 10, and has not been thoroughly tested on other platforms. The program is GUI based.

How to run the program:

1. Switch Eclipse's workspace in to the folder where the project files are located. If it does not appear go to File->Import->Existing projects into workspace and import it.
2. Press Run, you may have to right hand click on Application.java -> Run As -> Java Application to be the file to run you will get the following application to open:



3. Click Select File to select a .dat or .csv file containing the dataset, some pre-defined data sets can be found in the data directory of the source code folder
4. Select the desired level of minimum confidence and support from the sliders on the top left corner on the application.
5. Select Prune to toggle on Aprori pruning or Dont Prune to toggle Aprori pruning off.
6. Click Run to run mine for rules
7. The Stats pane will display the statics of the run such as:
    a. Time taken in seconds
    b. If pruning was enabled
    c. How many candidate sets were processed
    d. The minimum support and confidence
    e. The data source
8. The Transaction pane shows all the transaction
9. The Items pane shows all the unique items
10. The Association Rule Pane shows all the Association rules that where mined
11. The Steps pane shows the step by step solution to mining the rules
12. Click the Shop button to shop for items. You will get a shop popup.
    a. Holding the <CTRL> key select (or unselect) the desired items
    b. Click the Shop button on the popup to get suggestions for what else you can buy
    c. Click exit shop to exit the shop.



If you have any issues running the program please contact me:
ankushvarshneya@outlook.com or
akushvarshneya@cmail.carleton.ca