CSE 601: Data Mining and Bioinformatics

Project 1 - Part 2

Association Analysis

Presenters:
Hemant Koti (50338178)
Sai Hari Charan Barla (50336868)
Shravya Pentaparthi (50337027)

PROBLEM

Implement the Apriori algorithm for finding frequent itemsets and generating rules for the gene expressions dataset based on minimum support and confidence threshold. Further filter rules that are based on the given template queries.

DATASET

We have implemented the apriori algorithm and rule generation on the gene expressions dataset.

ALGORITHM DETAILS

Apriori Algorithm:

- Apriori algorithm operates on databases containing transactions to generate frequent itemsets. It uses the property that states, any subset of a frequent itemset must also be a frequent itemset, and similarly, for an infrequent itemset, all its supersets must also be infrequent. Using this property non-frequent itemsets are pruned at each step.
- The algorithm begins with an initial set of length - 1 item sets and proceeds further by merging and generating itemsets of a greater length until all the frequent itemsets are obtained. The non-frequent item sets are ignored based on a given minimum support value at every step of execution. This process is known as the pruning where all item sets that do not satisfy the minimum support constraint are ignored.
- Once the frequent itemsets are obtained, association rules are generated using binary partitioning and are filtered based on a given minimum confidence value

Implementation of Apriori algorithm:

- Firstly, Code reads the dataset with dimensions 100 rows (records) x 101 columns from a .txt file and converts each record to a set and the complete file as a list of sets where each set corresponds to a single patient. Each value in column 1-100 represents a gene expression as Up or Down. So, we rearrange each value in column 1-100 with the respective gene expression identifier. For example, every value in column 1 is appended with 'G1_', every value in column 2 is appended with 'G2_' and so on till the 100^{th} column.

- Secondly, Frequent Itemset need to be generated. To generate frequent item set a function *apriori()* is defined. This function initially computes all the 1 length frequent itemsets and then subsequently prunes the itemsets based on support count. In the later iterations, the frequent items are used to generate combinations that are stored as a list of

sets. We use set as an ideal data structure for our implementation as it avoids duplicate elements.

- Finally, Association rules need to be generated. To generate association rules, a function *generateRules()* is defined. In this function, we take the frequent itemsets generated in our earlier step and the maximum number of k-length frequent items generated. We iterate from 0 to k and generate subsets for each itemset. These subsets are then used to calculate the confidence thereby eliminating infrequent rules. We store the final set of rules in a dataframe for code convenience.

- For template parsing, there are 3 functions defined and are used for 3 different templates. All those 3 functions take a query and generated rules as input and they return a list of association rules that satisfy the given query and the total count of those rules.

RESULTS

The following values are obtained for task 1 and task 2.

Task 1:

1. **Support is set to be 30% :**
   Number of length 1 frequent itemsets: 196
   Number of length 2 frequent itemsets: 5340
   Number of length 3 frequent itemsets: 5287
   Number of length 4 frequent itemsets: 1518
   Number of length 5 frequent itemsets: 438
   Number of length 6 frequent itemsets: 88
   Number of length 7 frequent itemsets: 11
   Number of length 8 frequent itemsets: 1
   Number of all lengths frequent itemsets: 12879

2. **Support is set to be 40% :**
   Number of length 1 frequent itemsets: 167
   Number of length 2 frequent itemsets: 753
   Number of length 3 frequent itemsets: 149
   Number of length 4 frequent itemsets: 7
   Number of length 5 frequent itemsets: 1
   Number of all lengths frequent itemsets: 1077

3. **Support is set to be 50% :**
   Number of length 1 frequent itemsets: 109
   Number of length 2 frequent itemsets: 63
   Number of length 3 frequent itemsets: 2
   Number of all lengths frequent itemsets: 174

4. **Support is set to be 60% :**
   Number of length 1 frequent itemsets: 34
   Number of length 2 frequent itemsets: 2
   Number of all lengths frequent itemsets: 36

5. **Support is set to be 70% :**
   Number of length 1 frequent itemsets: 7
   Number of all lengths frequent itemsets: 7

Task 2:

1. **Template 1:**

   (result11, cnt) = asso_rule.template1("RULE", "ANY", ['G59_UP'])
   Count of number of rules generated: 26

   (result12, cnt) = asso_rule.template1("RULE", "NONE", ['G59_UP'])
   Count of the number of rules generated: 91

   (result13, cnt) = asso_rule.template1("RULE", 1, ['G59_UP', 'G10_Down'])
   Count of the number of rules generated: 39

   (result14, cnt) = asso_rule.template1("HEAD", "ANY", ['G59_UP'])
   Count of the number of rules generated: 9

   (result15, cnt) = asso_rule.template1("HEAD", "NONE", ['G59_UP'])
   Count of the number of rules generated: 108

   (result16, cnt) = asso_rule.template1("HEAD", 1, ['G59_UP', 'G10_Down'])
   Count of the number of rules generated: 17

   (result17, cnt) = asso_rule.template1("BODY", "ANY", ['G59_UP'])
   Count of the number of rules generated: 17

   (result18, cnt) = asso_rule.template1("BODY", "NONE", ['G59_UP'])
   Count of the number of rules generated: 100

   (result19, cnt) = asso_rule.template1("BODY", 1, ['G59_UP', 'G10_Down'])
   Count of the number of rules generated: 24

2. **Template 2:**

   (result21, cnt) = asso_rule.template2("RULE", 3)

Count of number of rules generated: 9

(result22, cnt) = asso_rule.template2("HEAD", 2)
Count of the number of rules generated: 6

(result23, cnt) = asso_rule.template2("BODY", 1)
Count of the number of rules generated: 117

3. **Template 3:**

(result31, cnt) = asso_rule.template3("1or1", "HEAD", "ANY", ['G10_Down'], "BODY", 1,
['G59_UP'])
Count of number of rules generated: 24

(result32, cnt) = asso_rule.template3("1and1", "HEAD", "ANY", ['G10_Down'], "BODY", 1, ['G59_UP'])
Count of the number of rules generated: 1

(result33, cnt) = asso_rule.template3("1or2", "HEAD", "ANY", ['G10_Down'], "BODY", 2)
Count of the number of rules generated: 11

(result34, cnt) = asso_rule.template3("1and2", "HEAD", "ANY", ['G10_Down'], "BODY", 2)
Count of the number of rules generated: 0

(result35, cnt) = asso_rule.template3("2or2", "HEAD", 1, "BODY", 2)
Count of the number of rules generated: 117

(result36, cnt) = asso_rule.template3("2and2", "HEAD", 1, "BODY", 2)
Count of the number of rules generated: 3