Association report

Team members

Somayeh Dejbord

Vineesh Ravindran

Namitha Paramel Somachudan

Apriori Algorithm and implementation

Apriori Algorithm is designed to generate frequent itemsets of the database of transactions (in general word). Apriori uses the property that states, if an itemset is frequent, then all of its subsets must also be frequent and hence for an infrequent itemset, all its supersets must also be infrequent and using this property non-frequent itemsets are pruned at each step. (Anti-monotone property)

Implementation of Apriori Algorithm:

The Code first reads the text file and converts each record to a set and the complete file as list of sets where each set corresponds to a single patient.

For generating the frequent itemsets, support percent and list of sets containing all the records should be passed to the Apriori () function. It generates frequent itemsets of sizes 1 by find_frequent_1_itemsets () function, and the entries that do not satisfy the minimum support requirement is deleted from the dictionary of freq_itemsets. Then, apriori_gen() function generates frequent itemsets of different sizes > 1 (say K), and prunes them by:

- has_infrequent_subset() function, to eliminate candidate itemsets containing subsets of length K-1 that are infrequent (Anti-monotone property)
- scan_count() and check_sublist(), to examine the support of each candidate by scanning the database

After every iteration the size of the frequent item-sets to be computed is increased by 1. For frequent sets of length greater than 2, sets with length k that have same k-1 values are combined to form sets with length k+1. These sets are added to list of frequent sets if they satisfy the minimum support condition. To calculate support for each candidate itemset, the number of times that the regarded candidate is present in the complete data is checked.

Part 1_Template Results:

Support is set to be 70%
Number of length-1 frequent itemsets: 7

Support is set to be 60%
Number of length-1 frequent itemsets: 34
Number of length-2 frequent itemsets: 2

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

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

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

Rule generation

For rule generation, we first take single items from each frequent item sets and check the corresponding confidence values. Any item that does not satisfy the confidence criteria is eliminated and the rest are used to prepare the candidates (subsets) of size 2 and the confidence for these are calculated. This process continues until we reach the length of the set. For any set that satisfies the confidence criteria, the rule Fk -> (Fk- Hm). (Confidence is calculated as sup(Fk) / Sup(Fk – Hm)).

Part2_Results for queries:

(result11, cnt) = asso_rule.template1("RULE", "ANY", ['G59_UP'])	26
(result12, cnt) = asso_rule.template1("RULE", "NONE", ['G59_UP'])	91
(result13, cnt) = asso_rule.template1("RULE", 1, ['G59_UP', 'G10_Down'])	39
(result14, cnt) = asso_rule.template1("HEAD", "ANY", ['G59_UP'])	9
(result15, cnt) = asso_rule.template1("HEAD", "NONE", ['G59_UP'])	108
(result16, cnt) = asso_rule.template1("HEAD", 1, ['G59_UP', 'G10_Down'])	17
(result17, cnt) = asso_rule.template1("BODY", "ANY", ['G59_UP'])	17
(result18, cnt) = asso_rule.template1("BODY", "NONE", ['G59_UP'])	100
(result19, cnt) = asso_rule.template1("BODY", 1, ['G59_UP', 'G10_Down'])	24
(result21, cnt) = asso_rule.template2("RULE", 3)	9
(result22, cnt) = asso_rule.template2("HEAD", 2)	6
(result23, cnt) = asso_rule.template2("BODY", 1)	117
(result31, cnt) = asso_rule.template3("1or1", "HEAD", "ANY", ['G10_Down'],"BODY", 1, ['G59_UP'])	24
(result32, cnt) = asso_rule.template3("1and1", "HEAD", "ANY",['G10_Down'], "BODY", 1, ['G59_UP'])	1

(result33, cnt) = asso_rule.template3("1or2", "HEAD", "ANY", ['G10_Down'],"BODY", 2)	11
(result34, cnt) = asso_rule.template3("1and2", "HEAD", "ANY",['G10_Down'], "BODY", 2)	0
(result35, cnt) = asso_rule.template3("2or2", "HEAD", 1, "BODY", 2)	117
(result36, cnt) = asso_rule.template3("2and2", "HEAD", 1, "BODY", 2)	3