FP Tree

Assignment 2

- 27 April 2017
  - VISHNU RAMESH(2015A7PS963H)
  - BHARGAV KANUPARTHI(2014A7PS527H)
  - KARTHIK MENON(2013B3A7487H)

Introduction

An FP-tree is a compact data structure that represents the data set in tree form. Each transaction is read and then mapped onto a path in the FP-tree.

FP Growth Algorithm

The FP growth algorithm is an alternative way to find frequent itemsets without using candidate generations, thus improving performance. For so much it uses a divide-and-conquer strategy.

In simple words, this algorithm works as follows: first it compresses the input database creating an FP-tree instance to represent frequent items. After this first step it divides the compressed database into a set of conditional databases, each one associated with one frequent pattern. Finally, each such database is mined separately. Using this strategy, the FP-Growth reduces the search costs looking for short patterns recursively and then concatenating them in the long frequent patterns, offering good selectivity.

In large databases, it's not possible to hold the FP-tree in the main memory. A strategy to cope with this problem is to firstly partition the database into a set of smaller databases (called projected databases), and then construct an FP-tree from each of these smaller databases.

The Apriori Algorithm

The FP Tree generation and the Apriori Algorithm are often used together for frequent subset generation.

The Apriori Algorithm is an influential algorithm for mining frequent itemsets for boolean association rules. 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: this has applications in domains such as market basket analysis.

Apriori is designed to operate on databases containing transactions Each transaction is seen as a set of items (an itemset). Given a threshold 'C', the Apriori algorithm identifies the item sets which are subsets of at least 'C' transactions in the database.

The Data Set

The raw data is sourced from National Institute of Diabetes and Digestive and Kidney Diseases.The data is based on females aged atleast 21 of PIMA Indian Heritage.

The attributes of the raw data contains:

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

Processing:

To convert to data to the typical market basket the data we had to first binarize the values. Each attribute was split into two bins or items. For each attribute if the attribute value was less than the median value it was put into one type of type and if it was greater or equal to the median value it was put into another type of item. The given data had 9 attributes and each attribute generates 2 items to the total number of items considered was 18.

Implementation:

Node structure – Node is a structure that stores the item it represents, its frequency in the FP tree and the it also stores a pointer to all its children.

getFirstItem – It is a function that returns the highest item number less than the lowest item number currently being considered.

addItem – This function is used to build the FP tree. It takes in a transaction and the a pointer to the root of the tree as an input and adds that transaction to the FP tree which is pointed to by that root.

getTempTree – This function takes an FP tree and an item ID as an input and generates a temporary tree which contains all the transactions that end with the given item ID. The corresponding counts of the nodes are also updated and finally the root of this temporary tree is returned.

getCounts – This function iterates over the given FP tree and updates the counts of all the items being considers based on the values in the tree

FPGrowth – This is the main function that generates all the frequent item sets. It takes a FP tree as an input and generates a temporary tree based on the value of lowed which is the item all the transactions end with. It then gets the counts and tries to growth the current frequent item set being considered one at a time. It considers each item to grow and then if it is above the threshold value it recursively calls the function with the new lowid and the trimmed tree. After this we have all our frequent item sets stored in a vector.

Results:

The frequency threshold was 0.2604 and the support confidence threshold was 0.5 . 212 interesting rules were generated from this algorithm .