Data Mining(CS6720): Assignment4

Abhishek Yadav, CS12B032

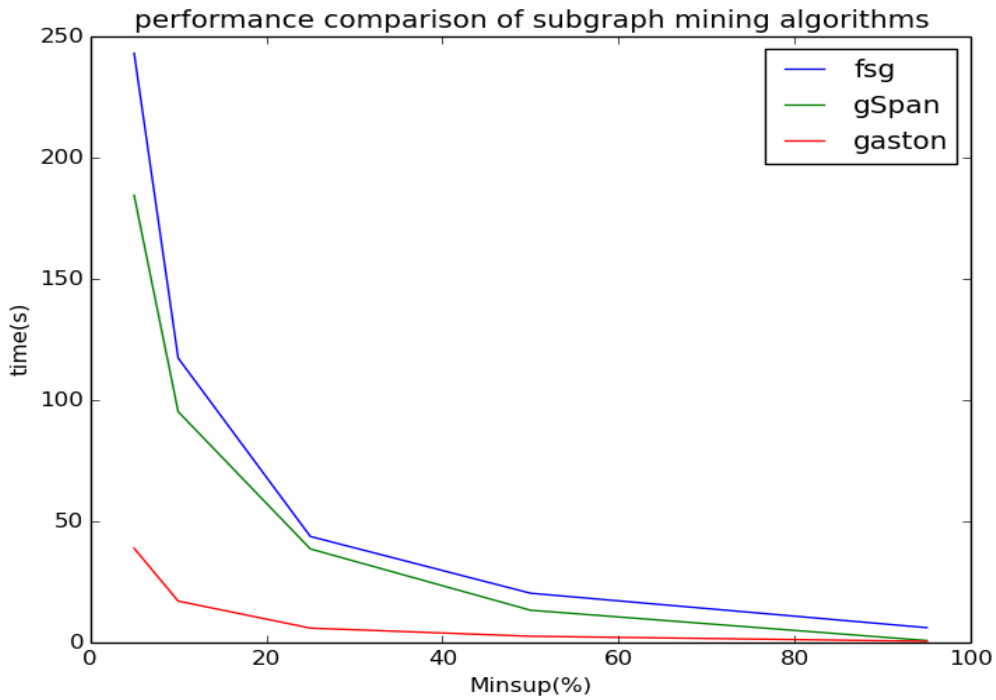
April 1, 2016

Q1

Details of the time(s) taken by **gSpan**, **FSG** and **Gaston** to generate the frequent subgraphs having the certain minsup values are listed as following:

Minsup Threshold	**FSG**	**gSpan**	**Gaston**
5	242.96	184.4	38.7575
10	117.217	95.1	16.9704
25	43.6418	38.5	5.82853
50	20.2219	13.2	2.46855
95	6.00204	0.7	0.397861

Corresponding minsup vs time(s) plot:



- **Trends Observed:** As Minimum threshold increase, time taken to mine frequent subgraphs decreases exponentially.
- **In Terms of Space: gaston** uses hash tables to store the graph representations which makes it faster than **gSpan** and **fsg**, which use adjacency lists, because of the speed in retrieving the results.
- **Subgraph Generation Method:** Between **fsg** and **gSpan**, gSpan uses rightmost extension which is faster than the single edge extension used in fsg.
- **Algorithms: fsg** uses apriori-based algorithm which transforms graphs into itemsets followed by frequent itemset discovery; the resulting frequent itemsets are transformed back to subgraph, as compared to the pattern-growth used by gSpan and gaston. Gaston is the fastest among the three due to the fact that it uses quick-start principle, where paths in a graph are considered first, which are then enumerated to trees and finally trees are enumerated to find the subgraphs.
- **Frequent Evaluation:** FSG uses transaction identifier (TID) lists for frequency counting. Each frequent subgraph has a list of transaction identifiers which support it. For computing frequency of a k subgraph, the intersection of the TID lists of (k – 1) subgraphs is computed. gSpan uses DFS lexicographic ordering for frequency evaluation. Here, each graph is mapped into a DFS sequence fol- lowed by construction of a lexicographic order among them based on these sequences, and thus a search tree is developed. Now the mini- mum DFS code obtained from this tree for a particular graph is the canonical label of that graph which helps in evaluating the frequency. Frequency counting process for Gaston is carried out with the help of embedding lists, where all the occurrences of a particular label are stored in the embedding lists.
- fsg uses BFS strategy whereas gSpan and gaston use DFS startegy.

Q2

Algorithm: In this case, we use gSpan to generate the frequent subgraphs from the training graphs. We have used minimum frequency threshold 0.1 in this case, which gives the best prediction results because considering smaller minsups biases results in favor of one class and keeping it high reduces the discriminating subgrpahs. Considering these frequent subgraphs as features we performed **Logisitic Regression** with load balancing for to train the classification model and use this model to predict the class label of the incoming molecules/graphs.

Q3

Algorithm: In the beginning of the algorithm, the neighborsList of all nodes of all graphs are sorted in ascending order of the edge Codes where edge Codes are denoted as **(srcNodeLabel, edgeLabel, destNodeLabel)**. Next, we find

all the nodes at which at least one edge with the least code(lexicographic) in the entire graph is incident. For example if we discover that the edge with encoding (C,1,C) is the least lexicographically then all the nodes incident to edges with this code are added to the start set of our algorithms. To further prune some start-nodes we search one more level down the graph from the start-nodes previously found and remove the those whose next-level least edge-code is larger than the others. Finally, dfs is called on all these start nodes and the least of these dfs codes is taken as final result.
DFS: to start **dfs** on a node, we first find out all the unvisited neighbors of the current node which are connected through the similar and smallest edge among the unvisited edges and then apply dfs on all of them and return the smallest of all codes.(**Recursive!!**). We use this strategy for every node at any point of time in the traversal tree. Most, importantly, while generating the codes, standard gSpan code generation/edge extension rules are followed.