COL 761 Data Mining Homework2

Varada Vinay Bhaskar (2020CS10405) - 50 %

Chinmayee Behara (2020CS10337) - 50 %

1 Question1

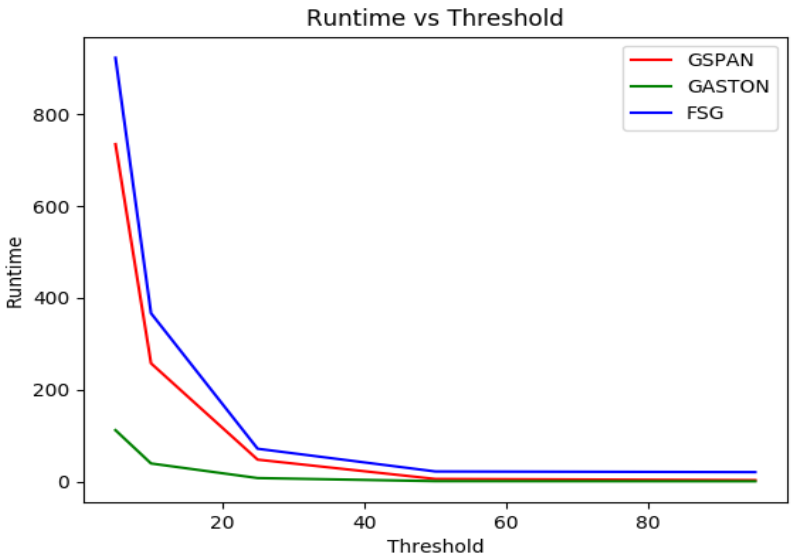
FSG: FSG, is similar to that of Apriori, and works as follows. FSG starts by enumerating all frequent single- and double-edge subgraphs. Then, it enters its main computational phase, which consists of a main iteration loop. During each iteration, FSG first generates all candidate subgraphs whose size is greater than the previous frequent ones by one edge, and then counts the frequency for each of these candidates and prunes subgraphs that do not satisfy the support constraint.

GSPAN: gSpan (graph-based Substructure pattern mining), which discovers frequent substructures without candidate generation. gSpan builds a new lexicographic order among graphs, and maps each graph to a unique minimum DFS code as its canonical label. Based on this lexicographic order, gSpan adopts the depth-first search strategy to mine frequent connected subgraphs efficiently.

GASTON : Searches for frequent structures by integrating a frequent path,tree and graph.

Threshold	FSG	GSPAN	GASTON
5%	922.6838586330414	734.6976864337921	112.0631694793701
10%	366.6271049976349	257.5869381427765	9.617027282714844
25%	71.64815282821655	47.94181990623474	7.820418834686279
50%	22.20700097084045	5.90467643737793	1.110994815826416
90%	20.76824903488159	3.295988559722900	0.847383499145507

37th Conference on Neural Information Processing Systems (NeurIPS 2023).



From the Plot we can see that the run time for all three techniques decreases as the threshold increases.

Among all the three Gaston is the fastest. Coming to speed GASTON>GSPAN>FSG

Coming to runtimes: FSG>GSPAN>GASTON.

We can see that as comparison to FSG and GSPAN GSPAN is faster.

The discrepancy in runtime between FSG and GSPAN is because of their methodologies. FSG uses the Apriori algorithm, utilizing a breadth-first approach that generates all potential candidates. However, verifying isomorphism in this exhaustive search can be time-consuming. Conversely, GSPAN adopts a depth-first approach, which significantly reduces the number of candidate graphs by employing minimal DFS code.

In GSPAN, K-sized candidates are efficiently generated by simply adding an edge to K-1 sized candidates, whereas FSG necessitates more complex processes, such as joining K-1 sized candidates with a common core, potentially resulting in multiple candidates.

Hence , FSG experiences longer runtimes due to its more exhaustive candidate generation process.

The gaston has less run time as it generates simple graphs for which checking of isomorphism is simple.

2 Question2

First we need to modify the given graphs.txt into the format that will be accepted by gSpan-64.

q2.cpp will do this part

Here i am using gSpan-64 for generating frequent sub-graphs from the given graphs

./gSpan-64 -f filename -s - k -o -i

Where k is the float and the out will be generated in filename.fp file.

We get all frequent sub-graphs whose absolute frequency is greater than k

After getting all the frequent sub-graphs with the above threshold (here i used 0.1 i.e 10 %) we will take the first 100 sub graphs of them as our features.This part will be done by check.cpp

Now we will label all the graphs by these features in the required format by final.cpp.

Results:

For **AIDS:**

Total Number of graphs : 2000

Total Number of subgraphs with frequency greater than 10% generated: 197

Train ROC_AUC: 0.991640625

Test ROC_AUC: 0.9633203124999999

For **Mutagenicity:**

Total Number of Graphs :4337

Total Number of subgraphs with frequency greater than 10% generated: 28404

Train ROC_AUC: 0.7476464843749999

Test ROC_AUC: 0.5201953125

For **NCI**:

Total Number of Graphs : 4110

Total Number of subgraphs with frequency greater than 10% generated: 21131

Train ROC_AUC: 0.6859277343750001

Test ROC_AUC: 0.54052734375

References

[1] https://sites.cs.ucsb.edu/ xyan/papers/gSpan-short.pdf

[2] http://glaros.dtc.umn.edu/gkhome/node/170

[3] https://liacs.leidenuniv.nl/ nijssensgr/gaston/gaston-april.pdf