Mining in Large Networks Assignment 4 Report

Name - Nirav Diwan
Roll Numer - 2017072

Q1

Evaluation
G - defined graph
G' - graph with edges removed
The evaluation was conducted based on assuming the missing links in G' (from G1) as the positive samples and the non-existent edges in G (edges which did not exist, as negative sample)

---------- Percentage = 25 ----------
Number of Positive edges = 525
Number of Negative edges = 189799
Number of test edges = 190324
Number of train_edges = 1577

Average over 10 folds
Jaccard Index = 0.8272977342697626
Common Neighbors = 0.8347831357433517
Preferentail Attachment = 0.7701897460948035
Adamic Adder = 0.8379978348021805
---------- Percentage = 35 ----------
Average over 10 folds
Jaccard Index = 0.8028856481290824
Common Neighbors = 0.8088724444724965
Preferentail Attachment = 0.7643347371456656
Adamic Adder = 0.8114550613210476

Number of Positive edges = 735
Number of Negative edges = 189799
Number of test edges = 190534
Number of train_edges = 1367

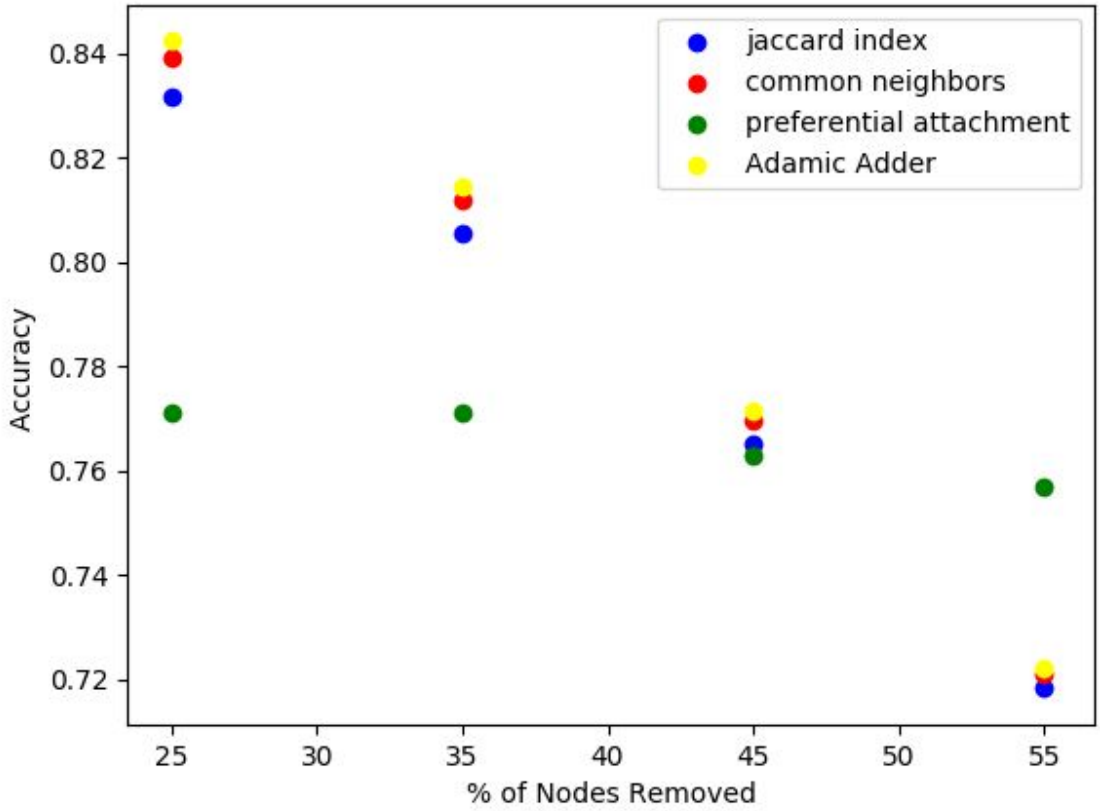
---------- Percentage = 45 ----------
Number of Positive edges = 945

Number of Negative edges = 189799
Number of test edges = 190744
Number of train_edges = 1157

Average over 10 folds
Jaccard Index = 0.7621740247570733
Common Neighbors = 0.7668723476361556
Preferentail Attachment = 0.7595417535972544
Adamic Adder = 0.7690013308704662

---------- Percentage = 55 ----------
Number of Positive edges = 1156
Number of Negative edges = 189799
Number of test edges = 190955
Number of train_edges = 946

Average over 10 folds
Jaccard Index = 0.7172816834494607
Common Neighbors = 0.7202944702327689
Preferentail Attachment = 0.7549416395902779
Adamic Adder = 0.7216456736119914



Observation

When we have a smaller test set, Adamic Adder is the best metric for link prediction. Preferential Attachment performs the worst in that scenario.

However, as the percentage removal increases i.e a larger set of positive edges, we observe that the performance of Adamic Adder along with common neighbours and Jaccard Index consistently decreases. Preferential attachment turns out to evade this pattern and performs better in a test set where more positive edges are to be predicted. This may be due to the pretty consistent average degree.

However, it is to be taken into account that the dataset is highly skewed towards the negative class. Perhaps some better metrics would be Precision, Recall and F1 Score.

Q2

a)

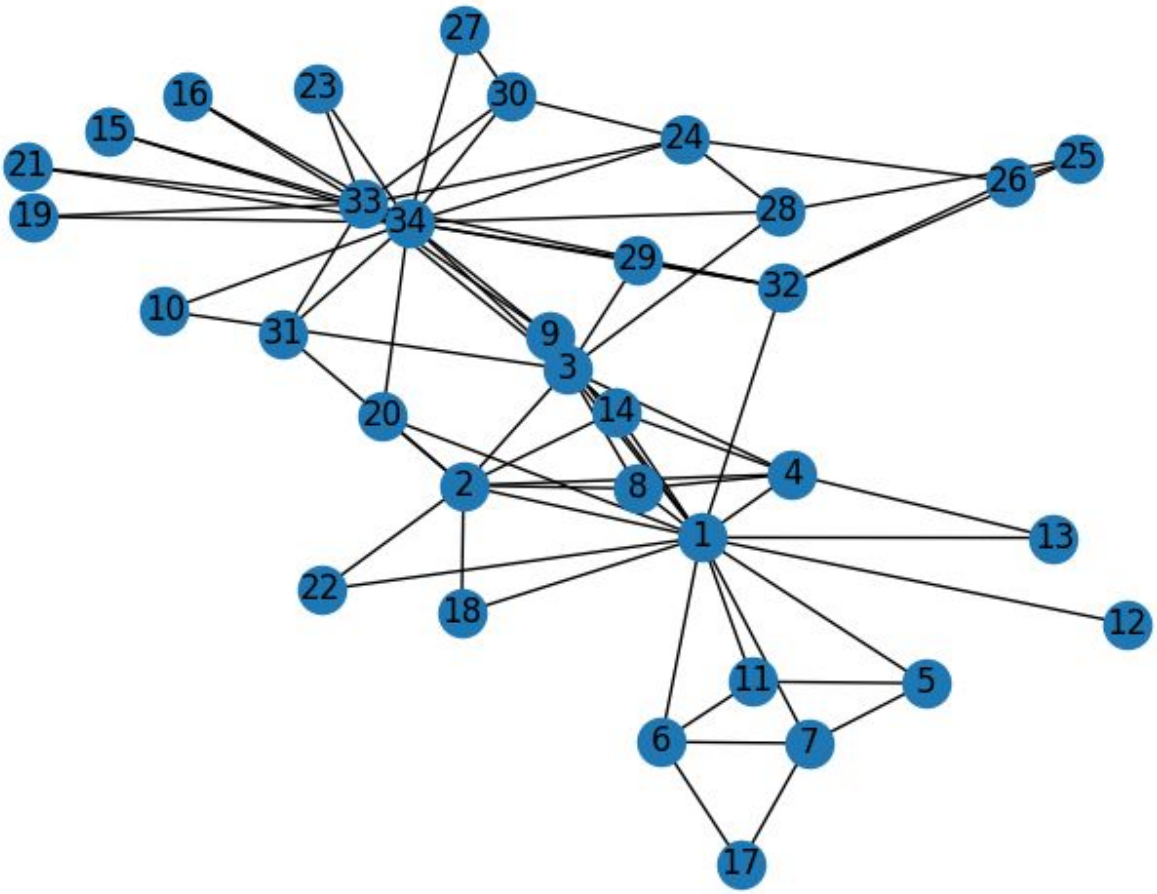


Figure 1:Visualization of the Karate Graph in a Spring Layout

Degrees

[('1', 16), ('32', 6), ('22', 2), ('20', 3), ('18', 2), ('14', 5), ('13', 2), ('12', 1), ('11', 3), ('9', 4), ('8', 4), ('7', 4), ('6', 4), ('5', 3), ('4', 6), ('3', 10), ('2', 9), ('31', 3), ('10', 2), ('33', 12), ('29', 3), ('28', 4), ('17', 2), ('34', 17), ('15', 2), ('16', 2), ('19', 2), ('21', 2), ('23', 2), ('24', 5), ('30', 4), ('26', 3), ('25', 3), ('27', 2)]

Role of Node 33

Node 33 is the node with the 2nd highest degree (12) behind node 34 (17). Node 33 acts as a hub or central figure in the graph. It is also connected to a lot of “peripheral” nodes with an extremely low degree (2). It introduces these nodes to the network. The connection between node 33 and Node 34 signifies that the most popular people also are connected to each other.

b)

Both node2vec and Deepwalk use the random walk to generate node embedding -

node2vec
It biases the random walk so as to perform a specific task such as identifying community structure. This is usually dependent on how we leverage the random walk in terms manages the bias of the random walk in terms of exploring the local structure (BFS) or exploring the global structure (DFS).

DeepWalk
Deepwalk consists of walks which are unbiased in nature. Hence when p=q=1, then there is no particular bias towards doing a BFS iteration or a DFS iteration of the random walk. Instead, it is likely that both the structures are being captured. However, it does not perform too well on a specific to the task, such as identifying community structure.

c)

I set p = 0.01, q = 100

Reasoning
We aim to detect nodes with the shortest paths. We have to examine the local structure w.r.t node 33. Hence, we would prefer a strategy such as BFS over DFS.
Generally, 1/p indicates the probability of return in the random walk, a low p would indicate a higher chance of a BFS route.
Similarly, 1/q is the probability of DFS to BFS, so a higher q indicates a higher chance of BFS and a lower chance of DFS which would indicate a more local structure is taken into account.

Nodes with shortest paths to Node 33
Node = 30 shortest_path_length with node 33 = 1 Cosine Similarity with node 33 = 0.009879469157988544
Node = 3 shortest_path_length with node 33 = 1 Cosine Similarity with node 33 = 0.005553272114242431
Node = 16 shortest_path_length with node 33 = 1 Cosine Similarity with node 33 = 0.004786691850127555
Node = 6 shortest_path_length with node 33 = 3 Cosine Similarity with node 33 = 0.004461477289363449
Node = 21 shortest_path_length with node 33 = 1 Cosine Similarity with node 33 = 0.003981658190223703

d)

I set p = 100, q = 0.01

Reasoning

We aim to detect nodes with a higher degree. We have to examine the global structure w.r.t every node. Hence, we would prefer a strategy such as DFS over BFS.
Generally, 1/p indicates the probability of return in the random walk, a high p would indicate a lower chance of a BFS route.
Similarly, 1/q is the probability of DFS to BFS, so a lower q indicates a lower chance of BFS and a higher chance of DFS which would indicate a more global structure is taken into account.

Nodes with Moderatly HIgh Degree similar to node 34
Node = 19 Degree = 2 Cosine Similarity with node 34 = 0.03445100967626038
Node = 33 Degree = 12 Cosine Similarity with node 34 = 0.02775731415390107
Node = 16 Degree = 2 Cosine Similarity with node 34 = 0.02729416215486905
Node = 3 Degree = 10 Cosine Similarity with node 34 = 0.02447548320172943
Node = 15 Degree = 2 Cosine Similarity with node 34 = 0.024328739813511517

e)

The minimum Euclidean Distance is for node 34 with a distance of 2.286141117057134

Degree of node 34 = 17
Degree for node 33 = 12

N1 N2 L2 Euclidean Distance with 33
34 33 2.286141117057134
1 33 4.86368739401253
3 33 3.5993404781614786
6 33 6.9083321736527195
7 33 6.981647504893327
2 33 4.344652826677239
5 33 7.156876958492163
11 33 7.366277426629677
24 33 3.958252947952833
28 33 3.6771511897626987
32 33 3.1902859399063996
17 33 6.13556549766286
14 33 2.900716034368218
4 33 4.549675893580177
30 33 3.436324670481893
26 33 3.7983292392723027
9 33 2.653638920362842

31 33 3.4287612668281304
25 33 3.72926824376154
29 33 2.7080761004653993
27 33 3.391958431675837
20 33 2.702318140894972
13 33 4.439842603767406
8 33 3.6450446851261975
10 33 2.391985681145629
15 33 2.6837385416607646
16 33 2.616670716335337
21 33 2.5249420578139725
23 33 2.5436946106280103
19 33 2.702748552474582
22 33 3.9916332080234382
18 33 3.807795645372756
12 33 4.284719235333518

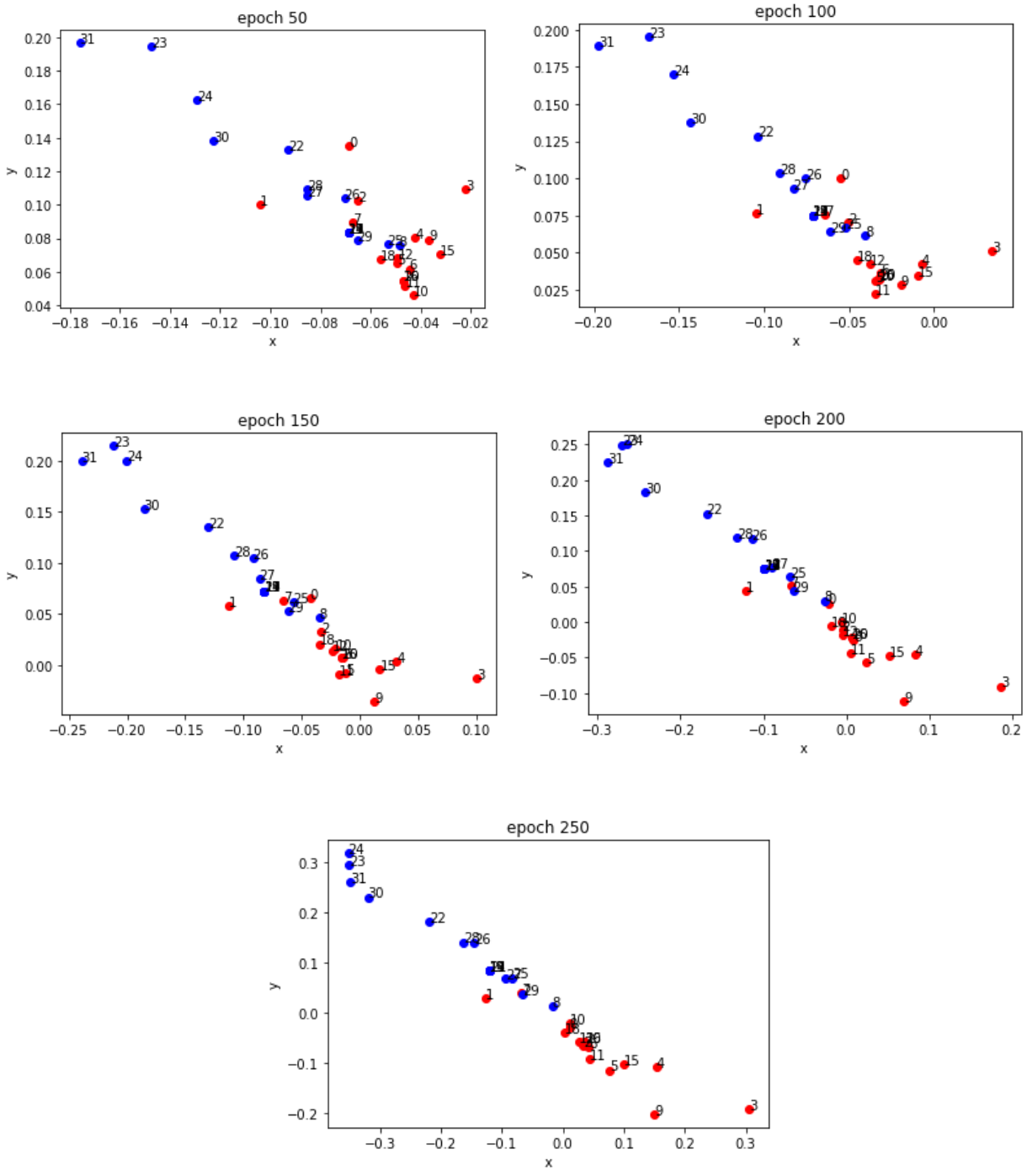
Explanation
The nodes which have a high number of common neighbours with node 33. We observe that their embeddings have a higher cosine similarity with node 33's embeddings.

Note:

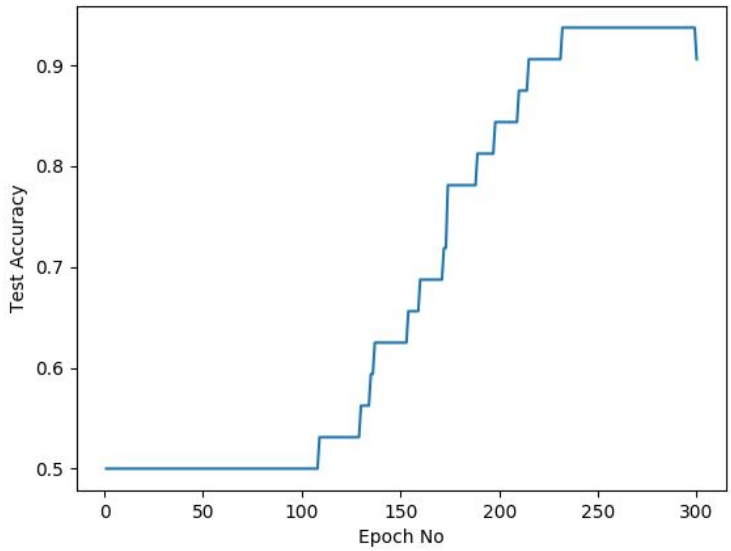
1) The code Initial model loading, setting of default hyperparameters and embeddings saving and loading part was directly referred by the node2vec GitHub library tutorial.
2) Setting Hyperparameters
   dimensions = 16,32 (an embedding size of > 34 does not make sense since #nodes is 34)
   walk_length = 60 (Tried for 15,30,45,60) , 60 gave the most consistent results
   num_walks = 200 (Tried for 10,20,50,100,200) - 200 gave the most consistent results
3) For parts c),d),e) all the parts utilized cosine similarity to measure the similarity of the node embeddings

Q3

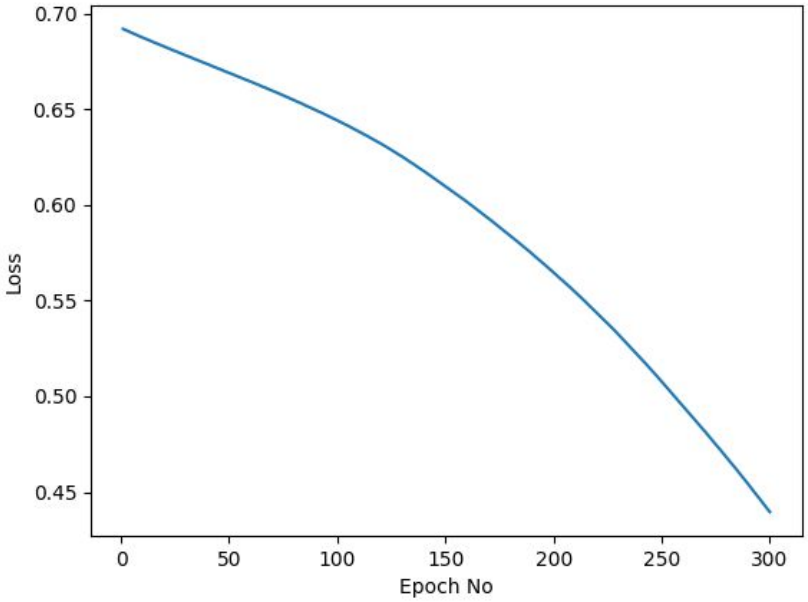
Visualization of features at different timestamps



Accuracy Plot



Loss Plot



Selected Epoch Results

Epoch 00005 | Loss 0.6894 | Test Acc 0.5000 | Time(s) 0.0395
Epoch 00006 | Loss 0.6889 | Test Acc 0.5000 | Time(s) 0.0384

Epoch 00112 | Loss 0.6365 | Test Acc 0.5312 | Time(s) 0.0399
Epoch 00113 | Loss 0.6359 | Test Acc 0.5312 | Time(s) 0.0399

Epoch 00133 | Loss 0.6222 | Test Acc 0.5625 | Time(s) 0.0407
Epoch 00134 | Loss 0.6215 | Test Acc 0.5938 | Time(s) 0.0406
Epoch 00135 | Loss 0.6207 | Test Acc 0.5938 | Time(s) 0.0406
Epoch 00136 | Loss 0.6200 | Test Acc 0.6250 | Time(s) 0.0406
Epoch 00137 | Loss 0.6192 | Test Acc 0.6250 | Time(s) 0.0405

Epoch 00170 | Loss 0.5920 | Test Acc 0.6875 | Time(s) 0.0399
Epoch 00171 | Loss 0.5911 | Test Acc 0.7188 | Time(s) 0.0398
Epoch 00172 | Loss 0.5902 | Test Acc 0.7188 | Time(s) 0.0399
Epoch 00173 | Loss 0.5893 | Test Acc 0.7812 | Time(s) 0.0399
Epoch 00193 | Loss 0.5706 | Test Acc 0.8125 | Time(s) 0.0398

Epoch 00229 | Loss 0.5323 | Test Acc 0.9062 | Time(s) 0.0394
Epoch 00230 | Loss 0.5311 | Test Acc 0.9062 | Time(s) 0.0395
Epoch 00231 | Loss 0.5300 | Test Acc 0.9375 | Time(s) 0.0394

Best Accuracy:

Epoch 00293 | Loss 0.4483 | Test Acc 0.9375 | Time(s) 0.0393

Note:

A significant part of the code for Q3 has been adopted from the tutorial mentioned in the question. The modifications are made accordingly as the question mentions.
The above results are based on

1) Changing the message and reduce function as given in the question
2) Using 1-hot encodings as features
3) Assuming semi-supervised setting and training based on the knowledge that node 0 and node 33 belong to different classes
4) A 2-layer GCN model is utilized