Matching Project Report

Sinan Seymen

June 12, 2020

1 Introduction

In this project we worked with 18 datasets to find the maximum cardinality matchings for each. My project was using Machine Learning models to solve this problem. I believe we were not allowed to look for existing solutions, therefore, my methods were basically I applied basic ideas and see which one performed better and try to build on that.

2 Method

I have created my data randomly. Only common point was the number of nodes were the same with the datasets given. The training and testing data have two separate components, one is the randomly created edges between given number of nodes. Second one is the results of the created data, found by networkx package. While choosing if an edge between two nodes exists or not I used uniform distribution with choosing a number as a threshold. Threshold is found by checking the probability of having an edge between two different nodes in all of the given datasets. I put some examples of the data I have created but not all of them.

After creating the data, what I do in my method is giving Adjacency matrix of the data and the matrix of edges which are in the maximum cardinality matching for that data. Therefore, everything is in binary. I flatten both of these matrices, and put them in multiple linear layers. I match the dimensions of these two flattened matrices and then using MSE loss function I try to understand how we can obtain which edges are in the matching or not. So, if an edge is in the matching it gets the value 1, and using MSE and linear layers we want the first value of the output to be 1 to minimize the loss, and if it is 0 then it is considered as error=1 and loss function increases.

Next, we obtain probabilities for each edge. We sort them from maximum to minimum. Then we start with an empty set, and start putting the edges and the nodes corresponding to that edges inside the set starting from maximum probability to minimum. If we want to put a new edge to the set but one of the nodes touching that edge is already in the set, we skip that edge. That is the greedy matching algorithm I wrote using the probabilities of the edges given. At the ends, optimal results are compared with the results of the greedy algorithm.

3 Results

While posting my results on Slack I changed the following:

- For each dataset I trained a different model, such that I took multiple runs with different learning rated, number of data used, numbers of hidden layers and number of nodes in those layers. However, in this write-up, because I want to follow a standard between all of the datasets I will pick one structure randomly and be consistent with that structure for all of the datasets.
- While getting the accuracy results on Slack I ignored some of the large datasets because I could not work with them. For the rest I got the best results of different runs and take the average of the best results.

Because of my large data time constraints I have used one randomly created data, I did not store the largest data efficiently therefore it was very time consuming to move the files to my local drive. Also I could not get results for the two largest datasets. I train for 5 epochs with learning rate 0.05. I took one hidden layer of node number 200 at the end because of time issues, 2 hidden layer versions performed similarly.

Therefore, for the large data I divided the data to smaller parts and used a smaller model weights to apply to each divided part separately. But this method was giving me low accuracy when I last tried which I broke something. After that I trained a small model and copied the same values for the rest of the big data until dimensions match. I took accuracy as 0.5 for the datasets I could not run because of the time limitation I am having right now.

Rest of the results are as shown for 3 separate runs for each dataset:

name	R1	R2	R3
662bus	259	258	263
b2ss	499	492	494
bcspwr01	15	15	15
bcspwr10	-	-	-
bcsstk05	67	63	65
can62	22	21	24
dwt72	29	28	29
dwt198	88	86	88
dwt2680	-	-	-
G15	362	364	368
G17	370	376	373
lpe226	171	168	172
lshp406	194	193	190
msc01440	709	705	711
poli	-	-	-
sphere3	120	119	120
mark3jac020sc	-	-	-
bayer04	-	-	-