Machine Learning with Graphs (MLG)

Homework 2 Report

吳承霖 成功大學電機所己組 N26090693

1 INTRODUCTION

Nowadays, the data structure behind applications could be described as graphs. Whether we want to know the the possibility that two users in social network may know each other, or we want to recommend some movies to the users in media platform. We all need to investigate the relationship between nodes in the graph, this type of problem is also called *link prediction* in graph.

As the scale of applications gets larger and larger, link prediction in the graph need a more efficient way to preform. Thus, in this project, we adopt neural network as method to predict whether any arbitrary two nodes in the graph dataset will establish a link or not based on only the attributes those nodes have.

In the following sections, we will illustrate the features we extraction from the given node attributes and try to explain the physical meaning of the extracted features. In the section of experiments, we train several models based on different set of features to investigate the relationship between input features and performance of prediction.

2 METHODOLOGY

In this section, we will focus on the neural network model for link prediction, including features extraction, model architecture, optimization and finally evaluation metrics of performance.

2.1 FEATURE EXTRACTION

The most straightforward features which can be generated from node pair attributes are number / rate of common neighbor and number of ‘interests’ (which has value 1 in the column) of a node. Besides the previous mentioned, we also calculate extra features including distribution of interests and number of interests in each N% of whole attributes.

Here are all features extracted from given node attributes:

Feature Name	Index of Feature Vector
Jaccard Coefficient	0
Common Neighbor	1
Preferential attachment	2
each-10% #1	7 ~ 16
each-10% #2	17 ~ 26
each-25% #1	27 ~ 30
each-25% #2	31 ~ 34
common rate of node 1	3
common rate of node 2	4
interest amount of node 1	5
interest amount of node 2	6
distance of mean of interest positions	35
standard deviation of position of node 1's interest	36
standard deviation of position of node 2's interest	37
product of two's standard deviation of interest position	38
standard deviation of position of common interests	39

Table 1. Extracted features

There're 40 features in total, the following subsections will explain each features calculation and intuition behind them.

2.1.1 Jaccard Coefficient

Jaccard coefficient, or *Intersection over Union*, is a intuitive way for predicting who similar the two nodes are. The formula of Jaccard coefficient is:

$$Jaccard\ Coefficient\ (from, to) = \frac{|F(from) \cap F(to)|}{|F(from) \cup F(to)|}$$

which $F(x)$ and $F(y)$ represent the interest of x and y respectively. The meaning of Jaccard Coefficient is the percentage of common interests of the node pair. If two nodes have exactly the same interests, then the value of Jaccard Coefficient would be 1. On the other hand, if two nodes have no any common interest, the value of Jaccard Coefficient would be zero. So the range of Jaccard Coefficient is 0 to 1.

2.1.2 Common Neighbor

In the given graph dataset, we can see attributes have value 1 as the neighbor of the node. That is, if any two nodes both have value 1 in the same attribute, we can say that they have this interest as common neighbor. The formula of Common Neighbor is:

$$Common\ Neighbor(from,\ to)\ =\ |F(from)\ \cap F(to)|$$

The intuition behinds the Common Neighbor is simple: If two have more common interests, then they are more potential to establish relationship.

2.1.3 Preferrential Attachment

The idea behind Preferential Attachment is that people who have many interests might be possible to follow others who also have many interests, so the formula of calculating preferential attachment is:

$$Preferential\ Attachment\ (from,\ to)\ =\ |F(from)|\ \times |F(to)|$$

2.1.4 Each-N% #1

Beside the global characteristics from the attributes mentioned above. We are also curious about the local information of the dataset. The *Each-N%* here means that we divide a node attributes into ten sub-part, the first sub-part is 0-10% of the original node attributes, and second sub-part is 11%-20% of node attributes, ands so on.

Each-N% #1 indicates that how many 1s in the aggregated node attributes, which are calculated by simply:

$$node\ attributes\ vector\ (from)\ +\ node\ attributes\ vector\ (to)$$

Value of attribute from aggregated vector equal to 1 means that there are only one person have this interest, and the other don't.

For example:

$$\textit{node attributes } A = (0,\ 0,\ 1,\ 0,\ 1,\ 0,\ 0,\ 1,\ 1,\ 0)$$
$$\textit{node attributes } B = (1,\ 1,\ 0,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 1)$$
$$\textit{aggregated node attributes} = (1,\ 1,\ 1,\ 1,\ 2,\ 0,\ 0,\ 1,\ 1,\ 1)$$

If we use *50* as *N*, then the each-50% #1 features will be:

$$each - 50\%\ \#1 = (4,\ 3)$$

2.1.5 Each-N% #2

Similar to Each-N% #1, Each-N% #2 also calculated from aggregated node attributes vectors. What makes the difference is that #2 indicates how many common interests (common neighbors) in the sub-part of attributes vector.

Use the previous example again:

$$\textit{node attributes } A = (0,\ 0,\ 1,\ 0,\ 1,\ 0,\ 0,\ 1,\ 1,\ 0)$$
$$\textit{node attributes } B = (1,\ 1,\ 0,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 1)$$
$$\textit{aggregated node attributes} = (1,\ 1,\ 1,\ 1,\ 2,\ 0,\ 0,\ 1,\ 1,\ 1)$$

If we use *50* as *N*, then the each-50% #1 features will be:

$$each - 50\%\ \#1 = (1,\ 0)$$

The idea of calculating #1 and #2 in Each-N% of node attributes is that we would like to know whether certain attributes as common interest (common neighbor) will positively determine the link between two people.

The value of N we used in this project are 10 and 25 respectively. The lower number of N might lead to a higher prediction ability, and the larger N might let the neural network model have better ability of generalization.

2.1.6 Common Rate of Interests

Although *Common Neighbor* (mentioned above) indicates the amount of common interests of two nodes, we still need to have features describe the percentage of common interests over node's interest, the formula is:

$$common\ rate\ (from)\ =\ \frac{common\ neighbor\ (from,\ to)}{interests\ amount\ (from)}$$

In this project we calculate $common\ rate(from)$ and $common\ rate(to)$ respectively because we assume that the these two features might have different effect.

2.1.7 Amount of Interest

Similar to *Preferential Attachment*, we assume that amount of interests of "from" node and "to" node will have different degree of influence, so we adopt these features.

2.1.8 Distance of Mean of Interest Position

All mentioned features above are discuss about the amount and rate of common interests. And we are curious about whether the distribution of interests will benefits the link prediction. The first feature to calculate about distribution is *mean of position of interest*. Here we design an example for better understanding:

$$node\ attributes\ (from)\ =\ (0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 0,\ 1,\ 0,\ 0)$$
$$node\ attributes\ (to)\ =\ (1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,0)$$
$$mean\ of\ position\ of\ interest\ (from)\ =\ 7$$
$$mean\ of\ position\ of\ interest\ (to)\ =\ 1.5$$
$$distance\ of\ mean\ (from,\ to)\ =\ |7\ -\ 1.5\ |\ =\ 5.5$$

$$normalized\ distance\ of\ mean\ (from,\ to)\ =\ \frac{5.5}{10}\ =\ 0.55$$

From the example we can observe that the interest of the two nodes are center at positions with very far distance. Based on the observation, we assume that larger of distance of mean of interest position will make it less possible to build link.

2.1.9 Standard Deviation of Position of Interest

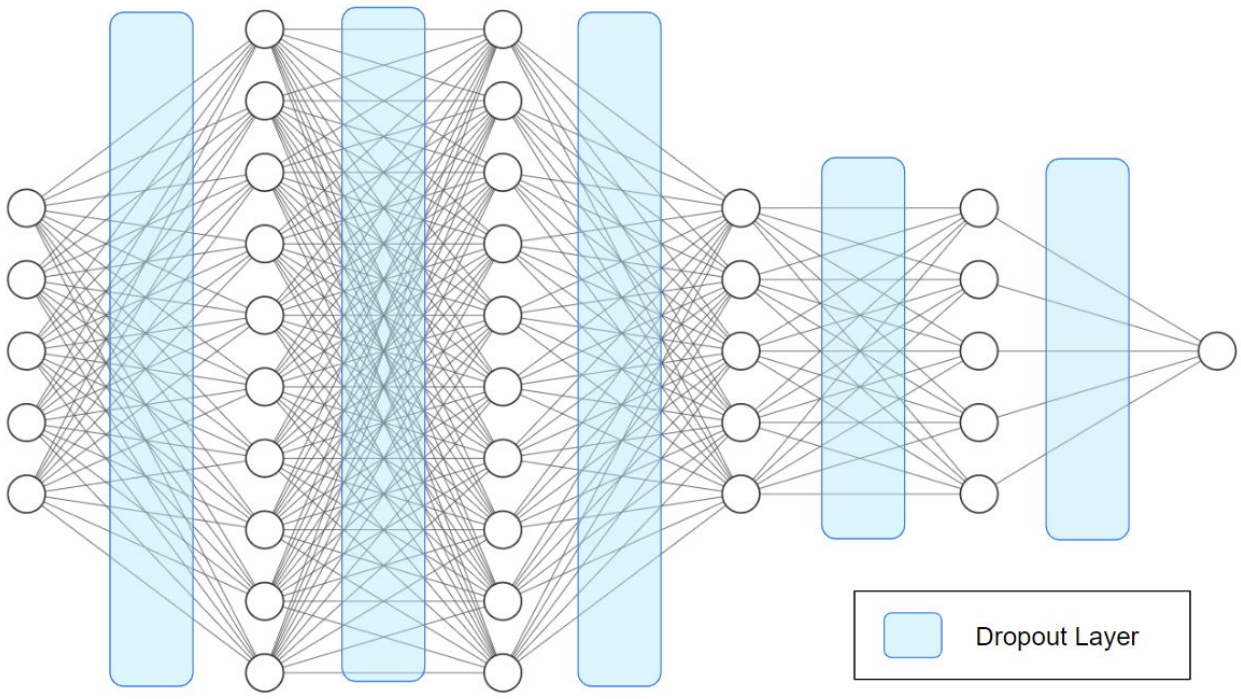
Standard Deviation of Position of Interest indicate how wide of the interests this person spread, if the value of standard deviation of position of interests is small, it possibly means that the person's interests densely gather at certain category.

2.1.10 Standard Deviation of Position of Common Neighbor

The method of calculating Standard Deviation of Position of Common Neighbor is similar to the features mentioned in last sub-section, but it calculate using common neighbor. The idea behind it is that: If common interests are grouped together densely, it's more possible that two people have more possibility to establish link.

However, the features described in section 2.1.8 to section 2.1.10 is based on the assumption that same category of attributes would be placed together in the node attributes of given dataset. If the attributes of same category (e.g. Sports) are scatter around the attribute vector. These calculated features might not work so well.

2.2 NETWORK MODEL



In this project, we use Pytorch to construct neural network models with 4 fully connected hidden layers. And we also add several dropout layers between hidden layers to prevent overfitting. The size of input vector is just for demonstrated, at the model of predicting test data, the size of input features is 35 (i.e we use 35 features for predicting).

2.3 OPTIMIZATION OBJECTIVE

Here we adopt binary cross entropy as loss function of model training. The reason why we choose binary cross entropy as optimization objective instead of mean square error is because that the loss would be larger in binary cross entropy when the predicted z and ground truth are literally opposite. Here is the formula of binary cross entropy:

$$BCE\ Loss\ (z,\ y)\ =\ y * log\sigma(z)\ +\ (1-y) * log(1\ -\ \sigma(z))$$

Where σ means sigmoid function. If z is very close to 1 and y is very close to 0, then the BCE Loss will give a much larger error than MSE do. This characteristic is beneficial for model training.

2.4 PERFORMANCE EVALUATION

To measure the performance of our trained model, we utilize AUC (area under curve) on ROC curve to measure the relationship between true positive rate and false positive rate in different classification threshold. In addition to AUC of ROC curve, we also use precision to measure how much percentage of the predicted result is true (true positive / predicted positive).

3 EXPERIMENT & ANALYSIS

In this section, we use several method to predict and use different set of features to train our model. After training been completed, we plot the trend of loss and precision as well as ROC AUC.

3.1 BASELINE 1

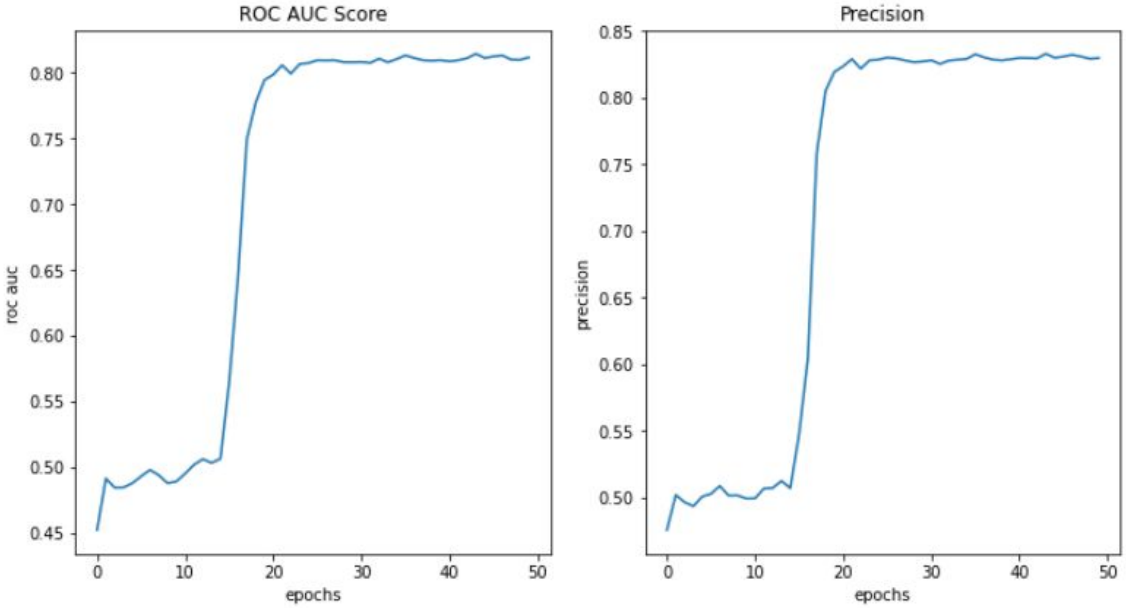
Baseline 1 simply outputs 0 (None) for every input node pair, and the results of precision and AUC (on dataset 1) are:

result \ Dataset	Dataset 1	Dataset 2	Dataset 3
AUC of ROC curve	0.5	0.5	0.5
Precision	0.4978	0.4952	0.4949

The result indicates that baseline 1 might not be the proper way of prediction for Dataset 1, 2 and 3. However, if someday our dataset is very skewed, say only 0.1% of ground truth of examples are positive and 99.9% ground truth of examples are negative. Then predicting every input with 0 would lead a very good result.

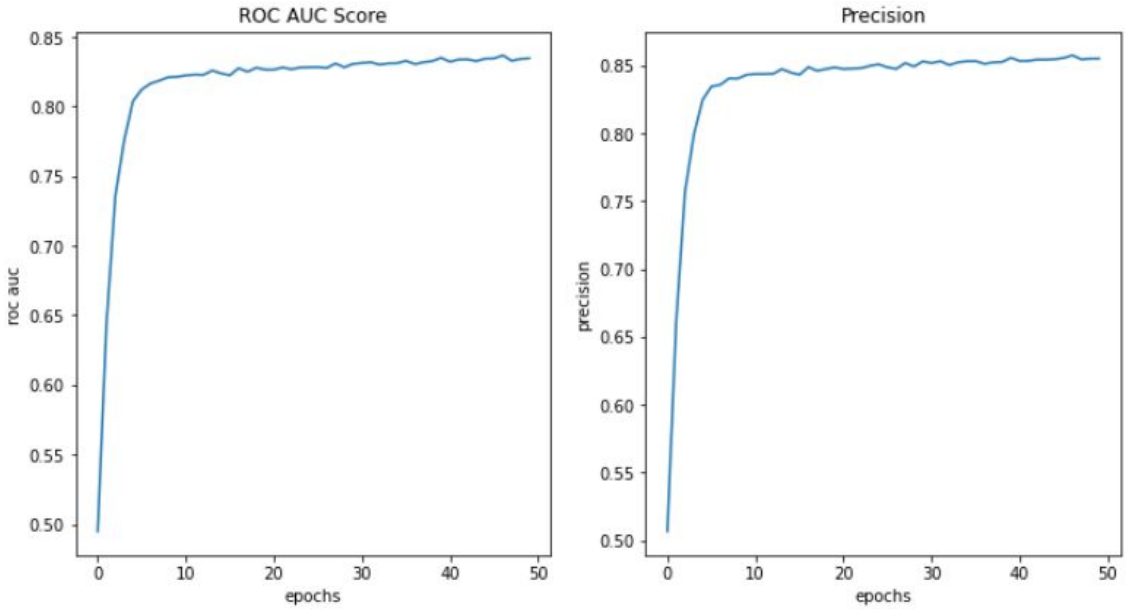
3.2 BASELINE 2

In baseline 2, we use Common Neighbor, Preferential Attachment and Jaccard Coefficient as well as other features describing amount of common interest to train the model. And the trend of losses and predicting performance along with epochs are plotted below:
from the plots we can see that the accuracy of model has been dramatically raised after several epochs. It's highly possible that using amount of interests and common neighbor is a effective way to predict links.



3.3 PROPOSED METHOD 1

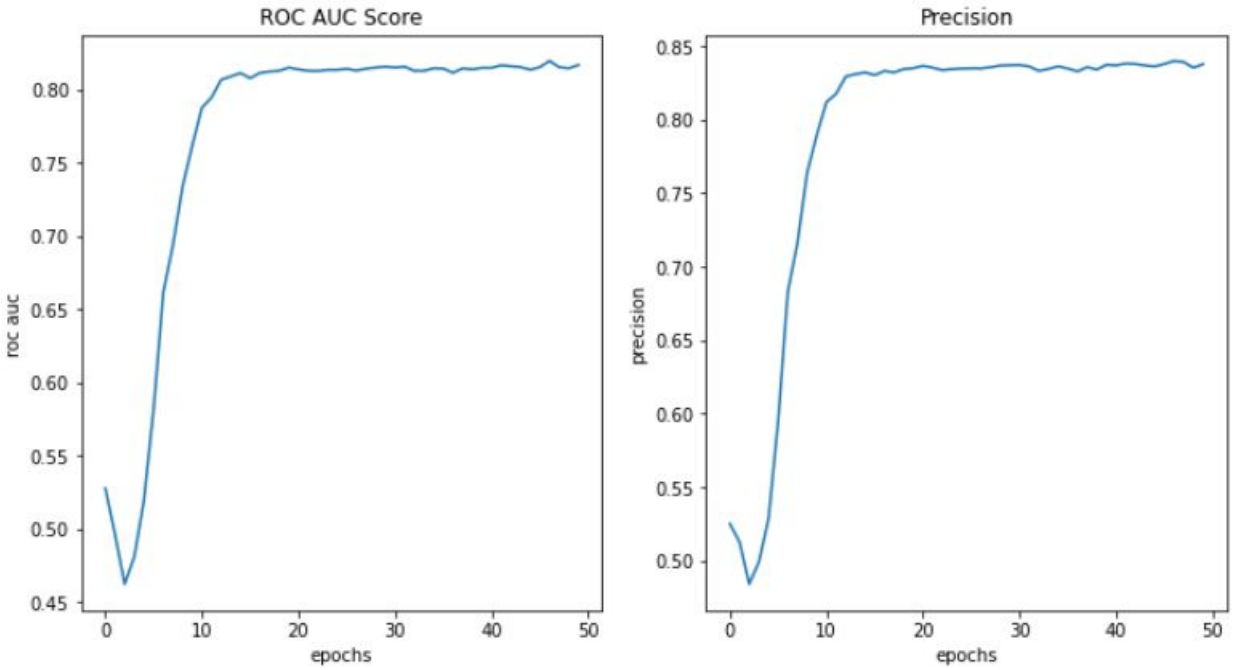
In proposed method 1, we instead use Each-10% #1 and Each-10% #2 as input features. The following plot is the performance of accuracy:



From the plots we can conclude that amount of common interests in different sections of node attributes vector have different influence in establishing links. That is, some of common interests may cause the two people create relationship, but some common interests don't.

3.4 PROPOSED METHOD 2

In proposed method 2, we instead use Each-25% #1 and Each-25% #2 as input features. The following plot is the performance of accuracy:

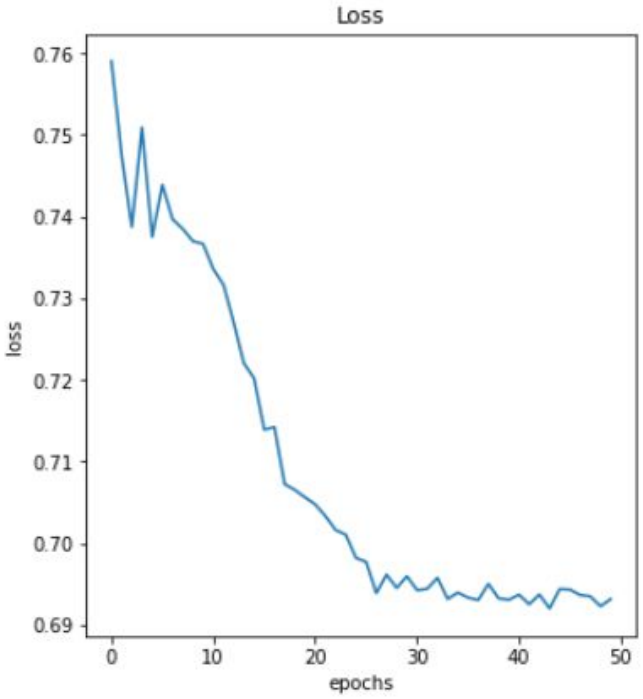


Similarly, these features could be used at predicting links just like Each-10% features. But the plot shows that Each-25% common neighbor will have lower accuracy thant Each-10% have.

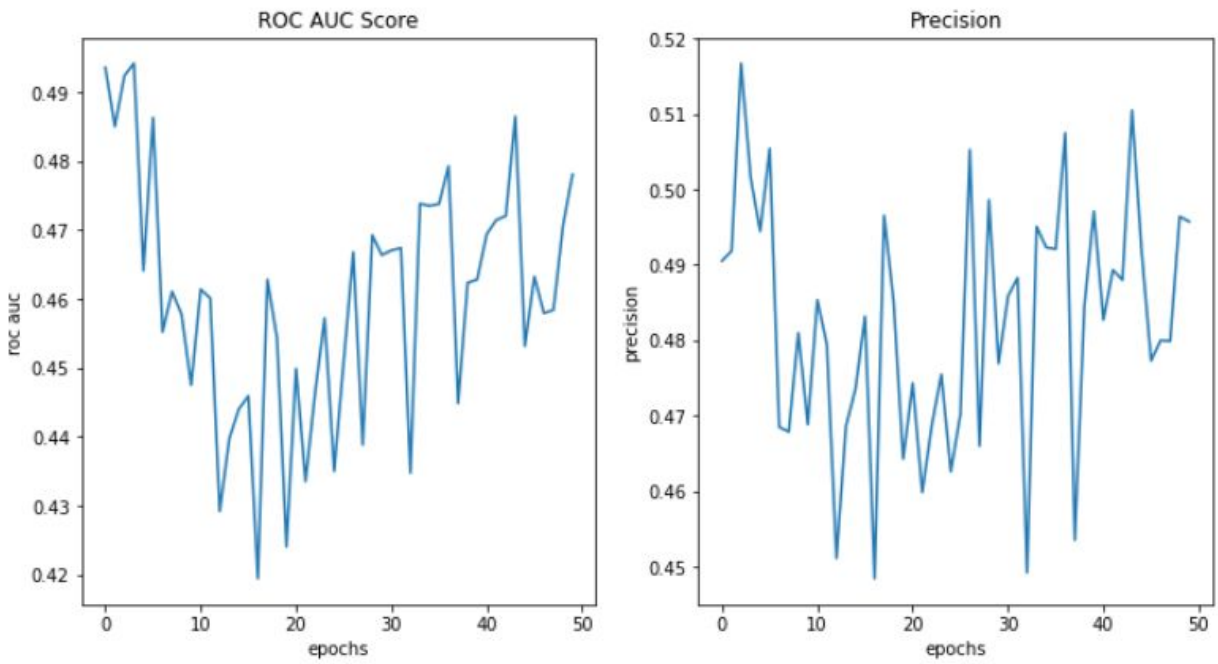
The phenomenon could be explained by the oversampling: some common interests will contribute more to creating node link, and some common interests are not relevant to node link, but they are group together to count the number of common interest. Thus the performance on smaller number of N will give better accuracy.

3.5 PROPOSED METHOD 3

Now we are tying to use distribution of position of interest including distance of mean and standard deviation as input features to train our mode. The following figure is the trend of loss:



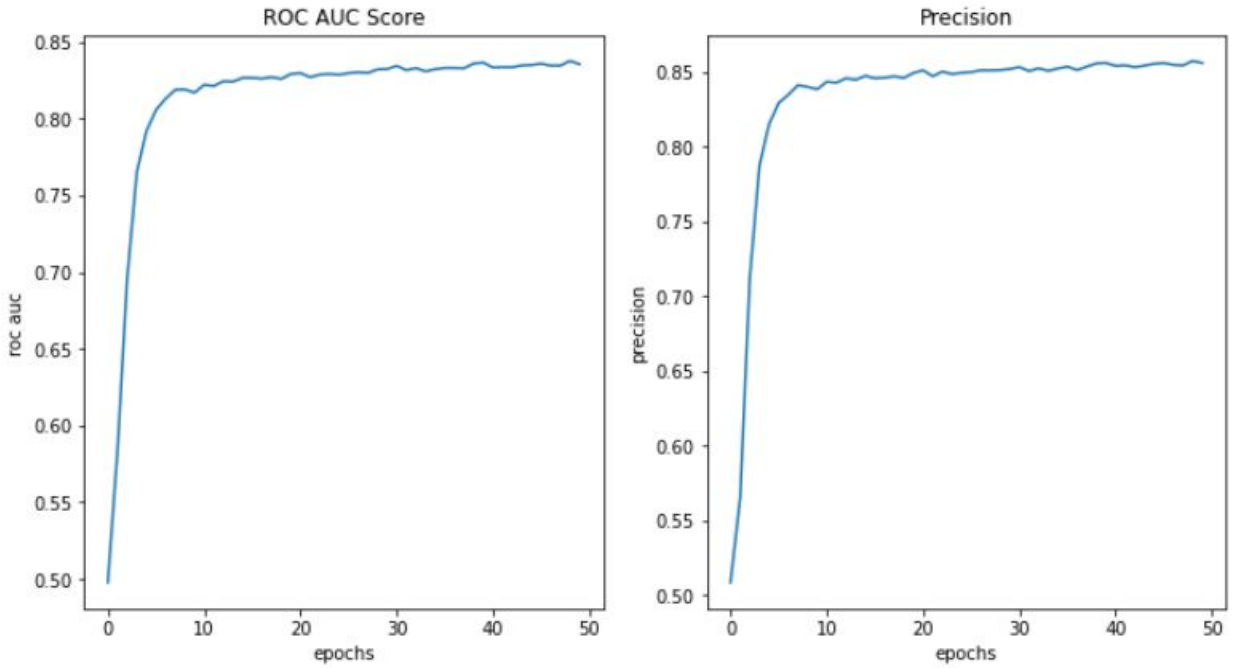
We can see that the loss did decline during training, however, when we look at the performance of precision:



The plotted precision shows that these features are useless in predicting links. It further implies that the attributes of the node are possibly not placed in categorical order.

3.6 PROPOSED METHOD 4

Proposed method 4 is a combination of baseline 2 and proposed method 1, 2 (except for proposed 3, which turns out to be useless). From the accuracy plotted in the following figures we can see that the ROC AUC score and precision are relatively higher than previous proposed method and baselines.



This series of experiments show that some of custom features are beneficial for predicting links and some are not relevant to. And we finally adopt the proposed method 4 as the model to predict test datasets.

4 CONCLUSION

In this project, we demonstrate how to investigate the relationship of two nodes in graph using features create from node attributes. Several experiments about custom features also indicate that the influence of common neighbors in different attributes might be different, and the amount of common interests is the key factor to determine the link establishing. Also, the metrics of evaluating performance of model are also useful tool when there's no specific threshold in binary classification problem.