# Project 2 Network Link Prediction
**Group 4**

Liu Dingdong, Wang Weiqi, Wang Yiren, Zhou Zixuan

## 1. Dataset Analysis

The dataset provides information on nodes and edges in a social network graph, where each node represents the ID of a user, and the edge represents that the two connected nodes are a friend to each other. The dataset is split into training, validation, and testing sets. Statistics of the three splits are shown in the table below:

| | Training | Validation | Testing |
|---|---|---|---|
| # Nodes | 8,343 | 5,457 | 5,425 |
| # Edges | 100,000 | 19,267 | 40,000 |
| Avg. # of edges per node | 23.97 | 7.06 | 14.75 |

*Table 1 Statistics of all three dataset splits.*

## 2. Methodology

### *2.1 Node2Vec*

Node2vec is an algorithmic framework for learning graph representations. Given a graph, node2vec is capable of learning continuous feature representations for the nodes, which can then be used for various downstream machine learning tasks. The most attractive feature of the node2vec framework is that it can learn low-dimensional representations for nodes in a graph by optimizing a neighborhood-preserving objective. The objective is flexible, and the algorithm accommodates various definitions of network neighborhoods by simulating biased random walks. Specifically, it provides a way for balancing the exploration-exploitation tradeoff, leading to representations obeying a spectrum of equivalences from homophily to structural equivalence.

### *2.2 DeepWalk*

DeepWalk is a type of graph neural network that operates directly on the target graph structure. It uses a randomized path traversing technique to provide insights into localized structures within networks. Utilizing these random paths as sequences is then used to train a Skip-Gram Language Model.

## 3. Experiment

### *3.1 Setup*

All experiments are conducted locally on a Windows machine with sufficient CPU calculation power and 16G memory installed. Specifically, we use the Word2Vec model from the Gensim[1] library to train our Skip-Gram model. All random walks are generated by considering the second-order information from the starting node. Cosine-similarity is used to calculate the score between every two nodes as the possibility of having an edge between them.

[1] https://radimrehurek.com/gensim/

### *3.2 Result*

For the node2vec framework, we finetune five parameters in total. The searching range of each parameter is presented in Table 2.

| *Hyperparameter* | *Search Space* |
|---|---|
| node_dim | 5, 7, 8, 9, 10, 11, 12, 13, 15, 20, 30, 40, 50 |
| num_walks | 5, 8, 10, 15, 20, 30 |
| walk_length | 5, 8, 10, 15, 20, 30 |
| p | 0.25, 0.5, 0.75, 1 |
| q | 0.25, 0.5, 0.75, 1 |

*Table 2 Searching space of all hyperparameters in the Node2vec framework.*

For the DeepWalk framework, we finetune two parameters as presented in Table 3 below. The node_dim hyperparameter is fixed to 10 in all experiments.

| *Hyperparameter* | *Search Space* |
|---|---|
| node_dim | 10 |
| num_walks | 5, 10, 20, 40 |
| walk_length | 10, 20, 40 |

*Table 3 Searching space of two hyperparameters in the DeepWalk framework.*

For the Deepwalk framework, we plot a heatmap presented in Figure 1 to demonstrate the maximum AUC score achieved by all hyperparameter combinations. The x-axis represents three possible values of walk_length, and the y-axis represents four possible values of num_walks. The result shows that when walk_length equals 20 and num_walks equals 10, the maximum AUC score is achieved at 0.9324.

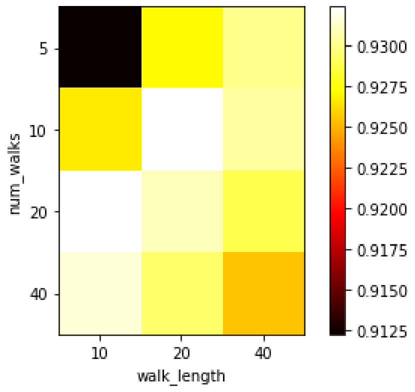


*Figure 1 Heatmap for DeepWalk*

As for the Node2vec framework, we plot a heatmap for every two hyperparameters among the five hyperparameters tuned in our experiments. Among all of them, we presented three interesting results.

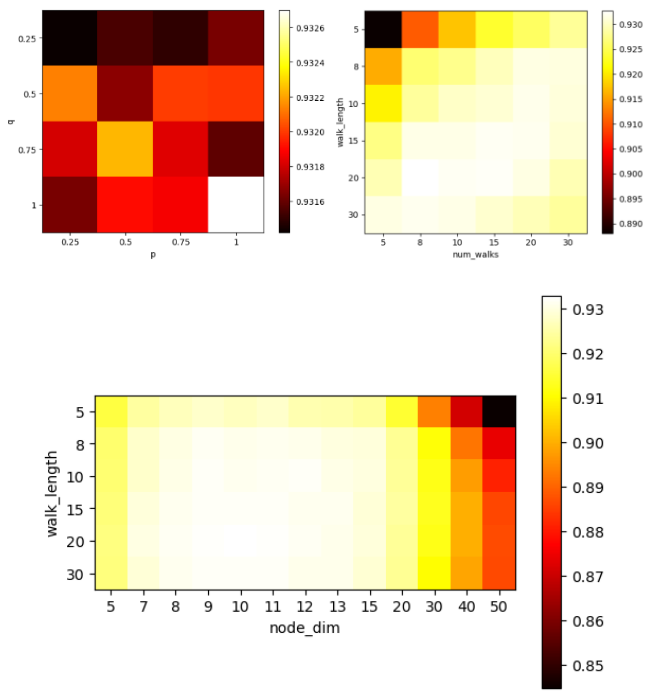


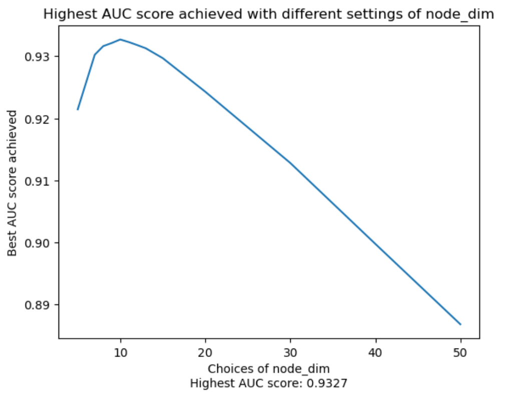
*Figure 2 Heatmap for Node2vec*

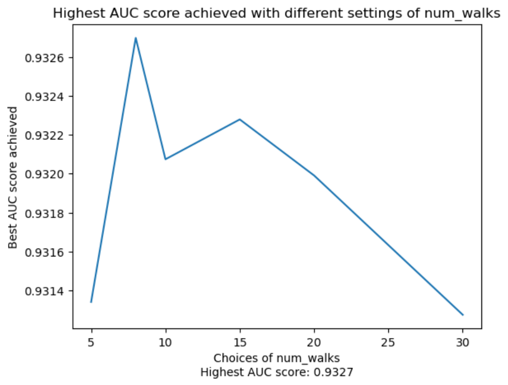
From these plots, we can observe that the p and q can achieve maximum performance when they are both set to 1. For the node_dim hyperparameter, we can observe that the AUC score increases when node_dim is getting larger from 5 and decreases when node_dim is getting smaller from 15. This indicates that a node_dim that is either small or large is unsuitable for the node2vec framework in our graph learning. In total, we have experimented with $13 * 6 * 6 * 4 * 4 = 7488$ settings. The highest AUC score is achieved by node_dim=10, num_walks=8, walk_length=20, p=q=1, with an AUC score of 0.9327. Both of the highest AUC scores in Deepwalk and Node2vec are higher than the 100% baseline (92.9%) on the validation set.

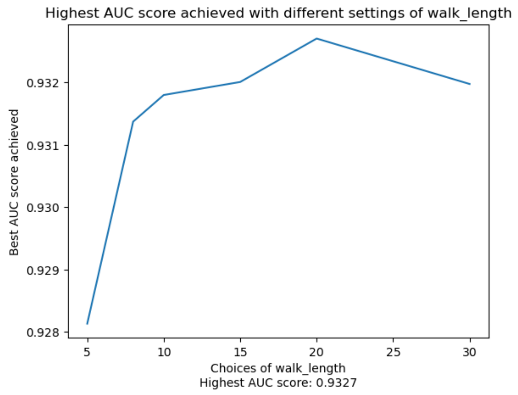
### *3.3 Analysis*

#### *Choosing node_dim*

To more carefully analyze the significance of every hyperparameter in the Node2Vec framework, we plotted the highest AUC score achieved for every possible value of every hyperparameter. For the node_dim hyperparameter, we can clearly observe the pattern that the highest AUC score is rising when node_dim is increasing to 10 but also lowering when decreasing to 50. Indicating that node_dim = 10 is the best choice even considering all other hyperparameters. Same conclusions can also be drawn from the plot of walk_length and num_walks. The best selection of num_walks is 8, while the best walk_length is 20. Thus we can know that when operating node2vec on our social network, it's better to have a low node embedding dimension and few random walks, while the length of each walk should be higher.







## 4. Conclusion

In this project, we performed a link prediction task on the social network where each node represents a user, and each edge represents a friendship between two users. We implemented two frameworks: DeepWalk and node2vec, to finetune their hyperparameters and search for the highest AUC score. Cosine similarity is used to calculate the link possibility between two nodes. Experiment results show that the highest AUC scores achieved by DeepWalk and Node2vec are 0.9324 and 0.9327, respectively. More specifically, we investigated the performance of hyperparameters by plotting heatmap and highest AUC curve. The best hyperparameters are then located for both frameworks.