**Project 2: link prediction**

**Table of Contents**

* Project introduction and workflow
* Question 1: node similarity calculation
* Question 2: node embedding and link prediction
* Data set and source code
* Submission

**Project introduction and workflow**

Networks are constructed from the observed interactions between entities, which may be incomplete or inaccurate. Link prediction refers to the task of predicting either missing interactions or links that may appear in the future in an evolving network, which attempts to discover missing links between nodes in a complex network, is of fundamental importance in numerous tasks in countless different domains. Link prediction is pervasive in bio- logical network analysis, where verifying the existence of links between nodes requires costly experimental tests. Limiting the experiments to links ordered by presence likelihood has been shown to be very cost effective.

For instance, link prediction can identify likely but not yet established links in an evolving social network, thus enabling recommendations to be presented to users.

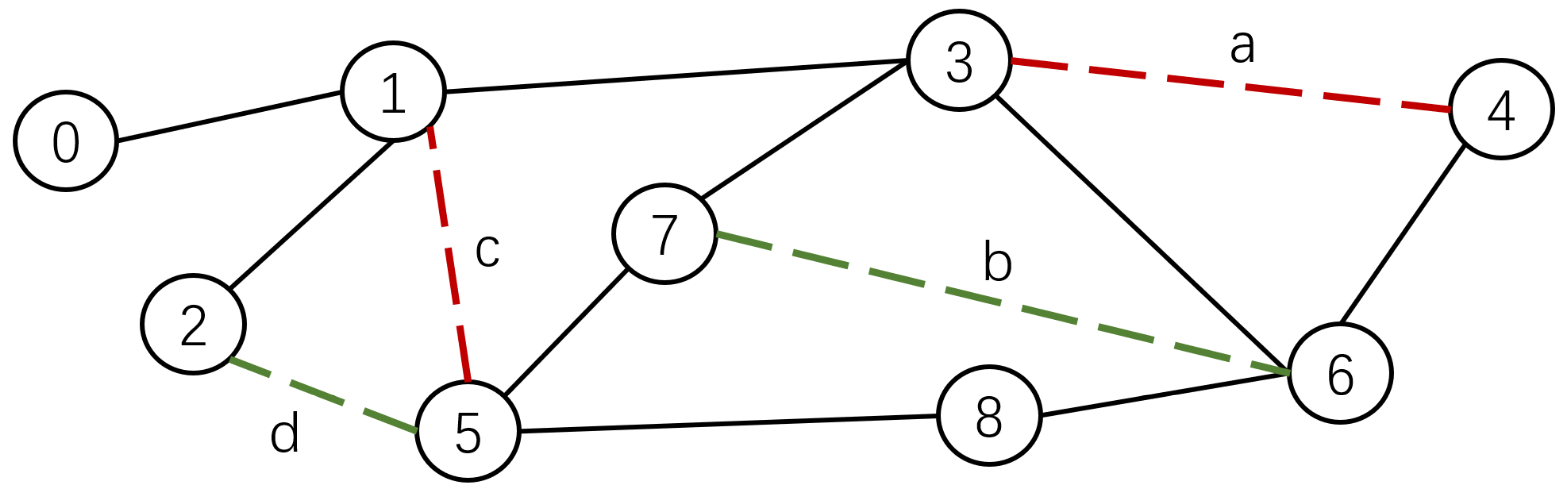


Figure 1

**Inputs:**

𝐺=(𝑉, 𝐸): the protein interaction network, 𝐸⊆(𝑉×𝑉), (𝑖,𝑗)∈𝐸 indicates there is an edge between gene 𝑖,𝑗 .

**Outputs:**

a low-dimensional vector of disease, 𝑉\_𝑑∈𝑅^𝑚, m is the dimension of representation space.

Using the training set and node features to train a classification model and predict the existence of edges in the test set. In figure 1, we are given a network with a certain fraction of edges removed (such as a, c), and we would like to predict these missing edges (such as a, b, c, d).

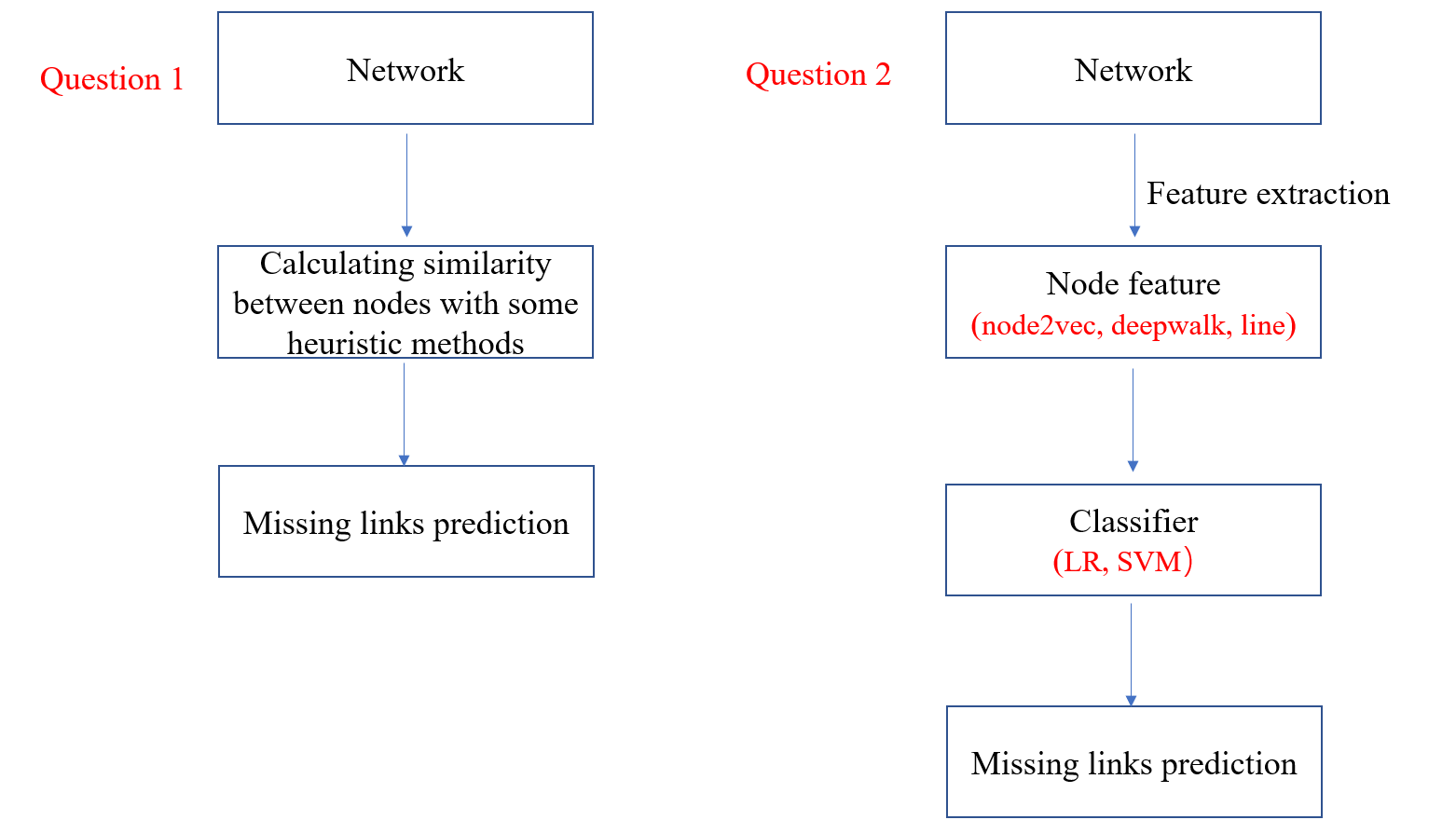


Figure 2

Figure 2 is the workflow of project.

**Question 1: Node similarity calculation**

Firstly, use some heuristic methods to calculate the similarity between nodes in the network such as the shortest distance, the common neighbors. Secondly, predict the missing links using node pair similarity score.

**Question 2: Node embedding and prediction**

* **Feature extraction**

Recently, methods which use the representation of network nodes in vector space have gained traction from the research community. We first learn features of each node in the network. Then we can train a classifier such as SVM, KNN to predict the missing links.

* We can construct the feature of each node based on the network structure such as the adjacency matrix.
* Dimension reduction-based methods:

We can obtain the adjacency matrix of the input network, then we can employ some dimension reduction-based methods such as PCA, SVD or autoencoder etc. to obtain the low-dimensional representation of each node.

[1]Wang D, Cui P, Zhu W. Structural Deep Network Embedding[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016:1225-1234.

* Random walk-based methods:

For node representation learning, random walk is exploited to capture structural relationships between vertices. By performing truncated random walks, an information network is transformed into a collection of vertex sequences, in which, the occurrence frequency of a vertex-context pair measures the structural distance between them

[2]Grover A, Leskovec J. node2vec: Scalable Feature Learning for Networks[J]. 2016, 2016:855-864.

[3]Perozzi B, Alrfou R, Skiena S. DeepWalk: online learning of social representations[J]. 2014:701-710.

* **Link prediction**

After obtaining the low-dimensional vector of each node, we can train a classifier (SVM, KNN; sklearn package in python) to predict whether a link in the test set exists or not.

The evaluation metrics is AUC value.

**Data set and source code**

Data file:

* **interactome.txt** : the input network (positive samples in training set)
* **train\_neg.txt** (negative samples in training set)
* **test\_pos.txt**: the missing links (positive samples in test set)
* **test\_neg.txt** (negative samples in test set)

Source code: (python 3.6, networkx package; sklearn package; numpy package; pandas package)

* project.py :
* Loading data

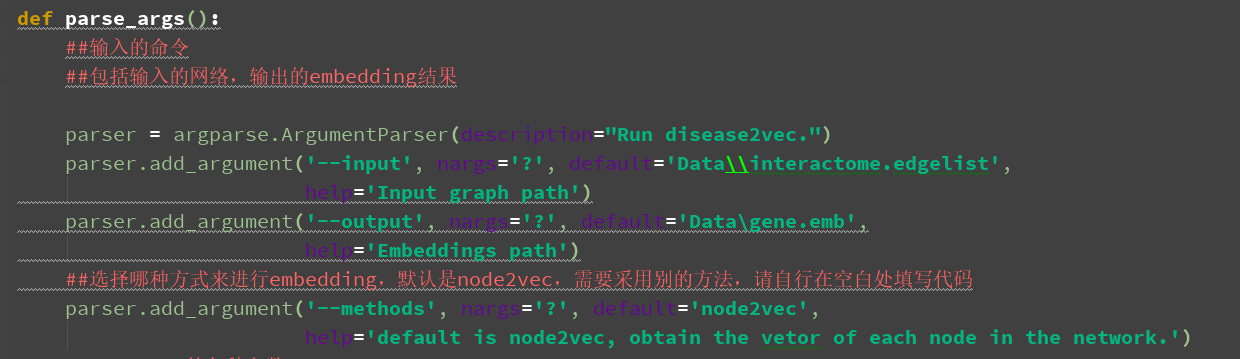


Figure 3

* Question 1: calculating the similarity between nodes directly based on some metrics such as the shortest distance.

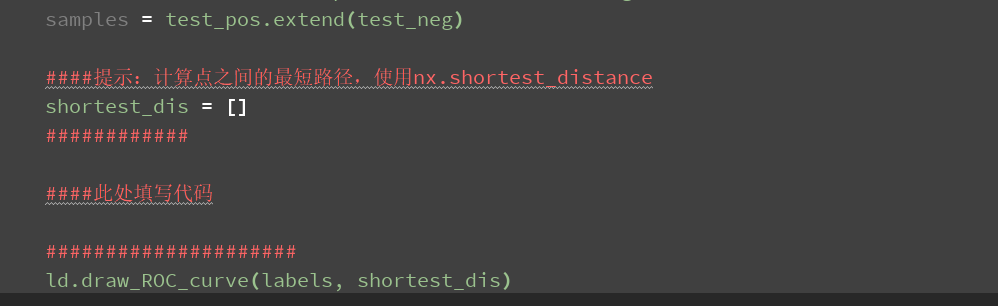


Figure 4

* Question 2: learning embeddings (The network is the input and node embedding is the output.)

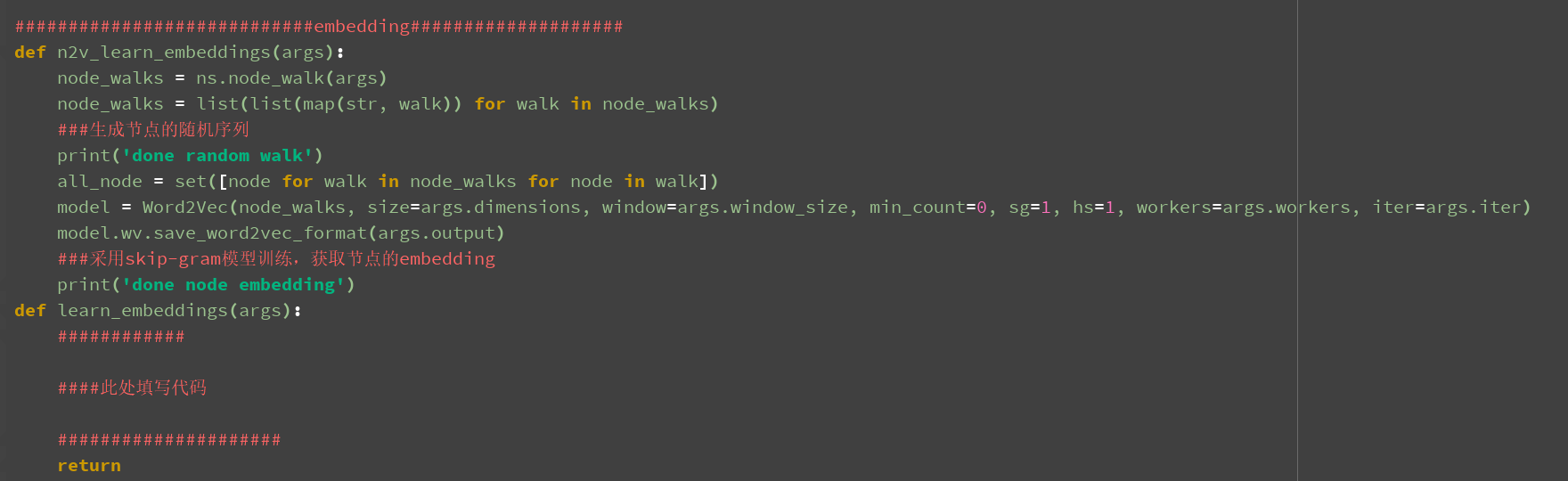


Figure 5

* Training classifier and drawing ROC curve

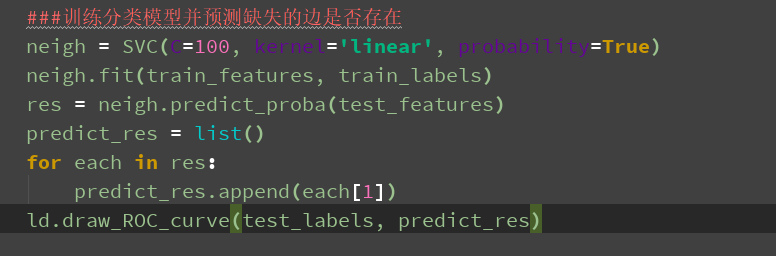


Figure 6

* Load\_data.py:
* loading training set and test set;
* drawing ROC curve
* Node\_sequence.py & node2vec.py:
* The source code of node2vec method.

Running code, the output is a ROC curve figure (figure 8 is an example of the ROC curve) and the AUC value!!!

Figure 7

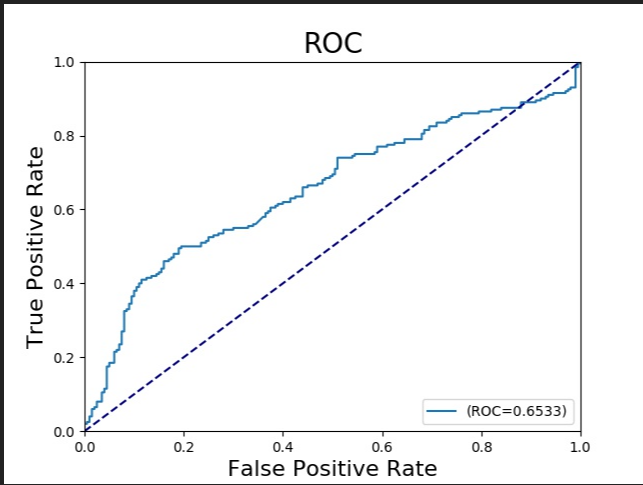


Figure 8

**Submission (IMPORTANT THINGS! READ THREE TIMES ☺ )**

* Submitting your code and a report ( **MAX 4 pages, 12pt font**), through E-learning system in a zip file with the name “pj2-prediction-student name-student No.”.
* Presentation.
* **Project 2 is due on Nov. 30, 2018.**