Exercise 4 – Link Prediction

①	②	③	④	∑
2	2	2	4	10

Preliminaries

In this exercise[1], we will empirically verify the friendship paradox which was proved in the lecture, and implement some link prediction algorithms. In addition to the normal tasks (Task 1 to 4), you may get 3 points from accomplishing the extra task (Task 5). However you cannot get more than 10 points in total. Please submit your solution before June 26, 2017 – 23:59.

1 The Friendship Paradox (2 Point)

It is sometimes said that most people have less friends than their friends, on average. We will verify this apparent paradox empirically using the Facebook social network containing 1.3 million friendships between 63,000 people[2].

For each user u in the network, compute the number of friends $d(u)$.

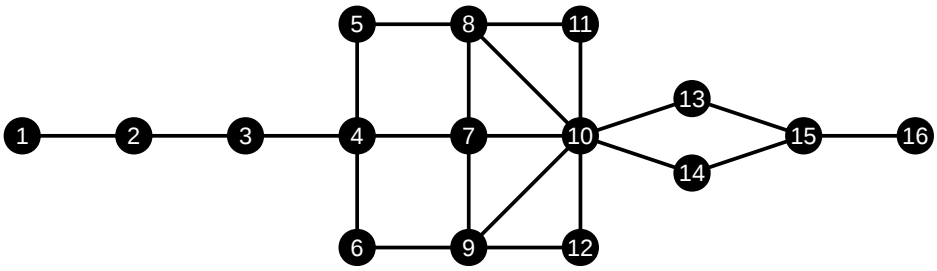
For each user u in the network, compute the average number of friends the friends of u have. Call this value $f(u)$.

Compare values d and f for all users. For how many users is $d(u) < f(u)$? For how many users is $d(u) = f(u)$? For how many users is $d(u) > f(u)$. Explain the results.

Compute the average number of friends $\bar{d}$ in network. Compute the average $\bar{f}$ over the average number of friends that friends have. Compare the values $\bar{d}$ and $\bar{f}$. Explain the results.

2 Friend of a Friend (2 Points)

Consider the following network (The sparse representation is available on the website[3]):



Compute the friend-of-a-friend link prediction matrix in two different ways:

- Using the square of the adjacency matrix

[1] This exercise is based on the exercise by Jérôme Kunegis.
[2] http://konect.uni-koblenz.de/networks/facebook-wosn-links
[3] https://west.uni-koblenz.de/sites/default/files/studying/courses/ss17/ntds/prediction-graph.txt

- Using the square of the eigenvalues in the eigenvalue decomposition $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$

Are both resulting matrices identical? Why?

How many common friends do the users 7 and 10 have? How many common friends do the users 7 and 15 have?

For user number 7, compute friend recommendations based on the friend-of-a-friend recommender. The friend recommendations should be a list of other users along with their link prediction scores. The list should be sorted by decreasing link prediction score. The list should not contain user 7 himself and his neighbors. Which user is at the top of this recommendation list?

3 Matrix Exponential (2 Points)

Use the eigenvalue decomposition of the above mentioned network's adjacency matrix $\mathbf{A}$ to compute the link prediction function $e^{0.1\mathbf{A}}$.

Compare the link prediction values of the pairs $(1, 2)$, $(1, 3)$ and $(1, 4)$. Which node pair has the highest link prediction score? Which has the lowest? Why?

Consider the node pairs $(7, 1)$ and $(7, 16)$. Do both pairs have the same link prediction score? If yes, why; and if not, why not?

Compute the friend recommendations for user number 7 based on the matrix exponential $e^{0.1\mathbf{A}}$, using the same rules as in the previous exercise. Compare the ranking of the recommendations for the friend-of-a-friend link prediction function with the matrix exponential. Explain your observations.

4 Sparse Decomposition (4 Points)

The function `eig()` is not scalable to large matrices. Instead, use the function `eigs()` to compute the sparse eigenvalue decomposition, as follows:

```
[U Lambda] = eigs(A, k);
```

This will compute a partial eigenvalue decomposition, which contains only the k largest absolute eigenvalues and their corresponding eigenvectors. Thus, the matrix $\mathbf{U}$ has size $n \times k$ and the matrix $\mathbf{\Lambda}$ has size $k \times k$. The matrix $\mathbf{\Lambda}$ is diagonal. However, the matrix $\mathbf{U}$ is not orthogonal anymore (all orthogonal matrices are square), but is still fulfills $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}$ (but *not* $\mathbf{U}\mathbf{U}^{\mathrm{T}} = \mathbf{I}$). The resulting decomposition $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$ is approximate, and can be used to compute the matrix exponential and other functions in an approximate way.

Compute the sparse eigenvalue decomposition with $k = 20$ eigenvalues for arXiv collaboration network[4]. You can ignore the edge timestamps included in the datasets; they are included in the fourth column of the `out.*` file (the third column contains only ones).

Compute the top-10 collaboration recommendations based on the link prediction function $e^{0.1\mathbf{A}}$ for users number 1, 22 and 333.

[4] http://konect.uni-koblenz.de/networks/ca-cit-HepTh

5 Algebraic Graph Theory (Extra[5])

Let $G = (V, E)$ be an undirected and unweighted graph without multiple edges. Let $\mathbf{A} \in \{0,1\}^{|V|\times|V|}$ be its adjacency matrix, and $\mathbf{D}$ its degree matrix. Recall that $\mathbf{D}$ is the diagonal $|V| \times |V|$ matrix defined by $\mathbf{D}_{uu} = \sum_{v\in V} \mathbf{A}_{uv}$ for all u.

Prove that the all-ones vector $\mathbf{1} \in \mathbb{R}^{|V|}$ defined by $\mathbf{1}_u = 1$ for all u is an eigenvector of the matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ (also known as the *Laplacian matrix*). What is its corresponding eigenvalue?

Tips

- The function `diag` will, when applied to a matrix, returns its diagonal as a vector; and when given a vector, return the corresponding diagonal matrix.
- The function `diag`, when applied to a vector `v`, will return a dense diagonal matrix. To return a sparse diagonal matrix, use `spdiags(v, [0], n, n)`.
- All adjacency matrices considered in this exercise are symmetric. If you get complex numbers as results, your input matrix may not be symmetric.
- The function `eigs` supports a display parameter `opts.disp` to monitor the decomposition algorithm.

[5] You may get 3 points from accomplishing this task. However you cannot get more than 10 points in total.