Graph link prediction

Dmitry Denisenko, Aliya Janabekova,
Ilia Nasedkin, Ilya Osokin, Artem Pimkin
Team 10

December 23, 2018

Abstract

We have developed and implemented several approaches to the problem of graph link prediction. We started with simple heuristics methods, then continued with more complicated method called spectral graph partition, and finally attempted to apply *word2vec* method to the graph analysis via considering random graph walks as sentences. We made conclusions on the possible reasons behind the chosen methods' performance and applicability.

1 Introduction

1.1 Problem formulation

Given a sparse undirected graph and a pair of nodes, we are to evaluate if there should be an edge between them or not. The information about this edge could be considered as absent or damaged, depending on the specific problem formulation.

1.2 Applications

Predicting graph edges is a multipurposal problem, its common and well-known applications include advising friends in social networks, image inpainting, image segmentation and many more.

1.3 Related works survey

1. **DeepWalk**

   DeepWalk is a deep learning approach for representing vertices in a network. It represents social relations as elements of a continuous vector space, that are easy to examine with statistical models. DeepWalk uses local information from the surrounding of the vertex to learn. It was trained on YouTube, Flickr and other multilabel datasets. The important feature of DeepWalk is its scalability: it is easily parallelizable.

2. **Node2Vec**

Node2Vec is a framework for learning continuous feature representations for nodes in a network. It uses stochastic gradient descent akin to backpropogation on a single hidden-layer feedforward neural networks. The main idea is to map the nodes into low-dimensional space subject to maximizing the likehood of preserving network neighborhood of nodes.

3. **LINE**

   LINE is a method of optimizing complex objective function, desighed in a way that makes it possible to preserve both local and global network structures. Authors propose novel edge-sampling algorithm, that improves classical stochastic gradient descent in terms of effectiveness and efficiency.

2 Dataset

We decided to use vk.com friends graph. We used https://github.com/stleon/vk_friends to gather the data. We took small random subset for our experiments (128x128 1,0 adjacency matrix).

3 Naive approach

3.1 General assumptions

The main idea of the naive approach is to find meaningful heuristical patterns, characteristics and dependencies that allow one to distinguish if two vertices should be connected in the graph or not.

Since the considered data is a graph of social relations, we attempted to find heuristics basing on the properties of two people's friends, or, in terms of Graph theory, adjacent vertices.

Ideally, we are looking for a scalar function of two sets of friends' *id*. After the value of the function is obtained, it is compared with a threshold and if it is higher, we consider people as possible friends and vice versa.

The chosen models were roughly tuned by varying thresholds.

3.2 Heuristics

3.2.1 Common friends

The idea behind this is that if two people have many common friends, they are more likely to become friends themselves, than if they don't.

$\mathcal{N}(u)$ denotes friends (adjacent nodes) of the person (node) u.

The scalar function, representing Common friends feature, is the following:

$$s = |\mathcal{N}(u) \cap \mathcal{N}(v)|$$

3.2.2 Jackard's coefficient

Jackard's coefficient, also known as intersection over union, represents which relative part of the peoples' friends is common for them. In some sense it is a development of the previous score, because it does not depend on the absolute numbers of friends, but on the ratios.

Also Jackard's coefficient is naturally bounded by 0 and 1.

$$s = \frac{|\mathcal{N}(u) \cap \mathcal{N}(v)|}{|\mathcal{N}(u) \cup \mathcal{N}(v)|}$$

3.2.3 Adamic-Adar score

Adamic-Adar score represents, roughly speaking, how strong are the bounds between two people. If they share a friend with relatively small number of friends, it means that the person makes friends slowly and every friend connection of him means much. Adamic-Adar score formula implies that the possible measure of "strength" is an inverse of the logarithm of the friends number.

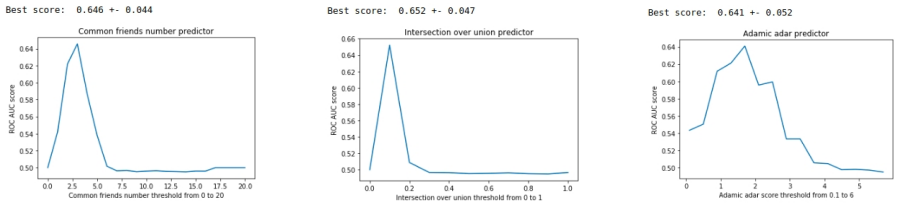
$$s = \sum_{t \in \mathcal{N}(u) \cap \mathcal{N}(v)} \frac{1}{\log |\mathcal{N}(t)|}$$

3.2.4 Preferential attachment

Preferential attachment is the simplest heuristic here, it basically relies on the assumption that more "social" people have higher probabilities of being friends. Such a general approach has its drawbacks, that led to the poor performance, see Discussion section for details.

3.3 Results

3.3.1 Successful



Common friends | Jackard's coefficient | Adamic-Adar score

Figure 1: Tuning the heuristical models

The best score for heuristic models is is 0.652 ± 0.047, achieved with Jackard's coefficient-based model.

3.3.2 Not successful

Preferential attachment score turned to be not applicable for this specific problem. The possible reason behind that is that it does not take into account any relations between groups of friends, only their sizes.

4 Spectral graph partition

4.1 Motivation

Heuristics can work well enough but this approach is not so universal and requires prior information about graph to fit better heuristic. Finding this prior information might be not so easy task, especially if the graph size is big, because it becomes difficult to visualize the graph and to find notable sub structures.

Spectral graph partition requires only the adjacency matrix of the graph, hence it's more universal than heuristics and can be widely used at least like the baseline for further research of the methods for graph link prediction. Another reason of using spectral graph partition comes from its effectiveness in various domains and the availability of a huge number of existing linear algebra packages to help solve the problem.

Let's consider the undirected graph of friends $G(V, E)$ (matrix consisting of 1 and 0) where 1 means presence of a friendship and 0 otherwise. The idea of the method is to find vector-format representation of peoples involvement in different groups (affiliations). One key observation is that when people belong to the same affiliation, they tend to connect to each other. For example, people of the same department interact with each other more frequently than any two random people in a network.

4.2 Method

As was shown in [2] spectral partition comes down to the eigenvalue decomposition problem

$$\min_S \ Tr(S^T \widetilde{L} S)$$

$$S^T S = I$$

where $\widetilde{L}$ is the normalized Laplacian defined below

$$\widetilde{L} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} A D^{-1/2}$$

A is the adjacency matrix of the graph G

The optimal solution of S corresponds to the first k eigenvectors of the normalized graph Laplacian $\widetilde{L}$ with the smallest eigenvalues.

Now, when we have matrix S, each row of such matrix can be considered as the features of each **node**. Next we need to find **edge** features and train the classifier (telling us the probability of edge presence between nodes) on these features.

We decided to use the same approach as in the paper [1]. Given two nodes u and v, we define a binary operator $\cdot$ over the corresponding feature vectors $f(u)$ and $f(v)$ in order to generate a representation $g(u, v)$ such that $g : V \times V \to R^d$. As a binary operator we chose Weighted-L1 operator defined as:

$$||f(u) \cdot f(v)||_i = |f_i(u) - f_i(v)|$$

The decision to use this operator was also based on the results shown in [1] for spectral partition method.

4.3 Implementation

To implement this approach we decided to use standard python scientific stack libraries (numpy and scipy).

4.4 Results

To assess the results we used the following pipeline:

1. Find node embedding
2. Get all connected pairs of nodes and create the random subset of the same size of not connected pairs of vertices
3. Transform pairs of nodes features to edges features for both positive and negative class
4. Gather the best hyper parameters for the chosen classifier (in our case SVM) using cross validation based on the ROC-AUC score

Finally, we achieved 0.887 ± 0.023 (mean cross-val score) using this method and our dataset.

5 Word2vec reduction approach

5.1 Motivation

Our approach that we have tested was reducing of the stated problem to the word embedding problem (that we have already solved in HA). The main idea was the following: what if we consider each node as a word and a goal that we want to achieve is the linear relationship between the similar nodes as it is for word embedding problem. So, considering each node as a word we have to obtain the text which structure will fit our problem statement. To obtain that, we considered random sentences composition from a random walks through the graph. To enrich the context of each node we considered not a direct random walk (using edges from current vertex), but random walk with a possibility $p_{ij} = f(l_{i,j})$, where l_{ij} is the length of the shortest path between vertices i and j. Then word2vec supposed to be run on these sentences (such an approach, as we understood later is pretty similar to DeepWalk with weighted window, though). Similar as for previous paragraph, after obtaining node embedding, we construct edge embedding with Weighted-L_1 operator (it has to be a good idea since word2vec approach linear relationships between similar nodes as it was said above) to further provide an edge classifier.

5.2 Implementation

We will not repeat here the word2vec method (to avoid straight HA copying), but will list hyperparameters of our method to be varied here. So, we have pretty sparse graph G with n nodes, m edges and we can't guarantee its connectivity. We are trying to build the text of S sentences of length l using random walk transition function $p_{i_j} = f(l_{i,j}), l_{i,j} = \overline{0,n} \cup \emptyset$. Actually, we tried cut end normalized Poisson distribution (cut to maximum path length from current

vertex, then normalized to sum of 1) with 0 for absence of path appended. Also we tried different normalized decreasing functions like $l_{i,j}^{-\lambda}$, etc.

5.3 Results

By the way, as it was said during our presentation, with this method we did not obtain any good results that differ from 0.5 roc-auc significantly. It may be due to implementational error, or caused by approach in this statement invalidity.

6 Results comparison, discussion

Spectral partition showed better results with a huge gap (0.652 for heuristics compared with 0.88 for spectral partition). But actually there are some crucial points which must be considered in future work:

1. We used very small graph, so our approaches must be further tested on bigger structures
2. Spectral graph partition is good, but if we use bigger graphs, then eigenvalue decomposition takes a lot of time ($O(n^3)$ complexity (this is one of the reasons why we used such small graph). Hence online methods (like node2vec) may be more preferrable

7 References, GitHub link

7.1 References

1. Node2vec: Scalable Feature Learning for Networks
2. Leveraging social media networks for classification
3. Deep Graph Kernels
4. LINE: Large-scale Information Network Embedding
5. DeepWalk
6. Subgraph2vec

7.2 GitHub link

Here

8 Personal contribution

- Dmitry Denisenko: spectral graph partition
- Aliya Janabekova: heuristic models implementation
- Ilia Nasedkin: data gathering, spectral graph partition
- Ilya Osokin: heuristic models testing baseline
- Artem Pimkin: w2vec adaptation, data gathering