Your Original and Relevant Course Project Title

Social Network Analysis for Computer Scientists — Course paper

Chenyu Shi
s3500063@umail.leidenuniv.nl
LIACS, Leiden University
Leiden, Netherlands

Shupei Li
s3430863@umail.leidenuniv.nl
LIACS, Leiden University
Leiden, Netherlands

ABSTRACT

KEYWORDS

node2vec, GCN, graph embeddings, social network analysis, network science

ACM Reference Format:
Chenyu Shi and Shupei Li. 2022. Your Original and Relevant Course Project Title: Social Network Analysis for Computer Scientists — Course paper. In *Proceedings of Social Network Analysis for Computer Scientists Course 2022 (SNACS '22).* ACM, New York, NY, USA, 3 pages.

1 INTRODUCTION

Graphs are mathematical objects that can model complex relationships on non-Eucildean space. They are widely used in multiple domains such as molecular structure modelling, social network analysis, recommender systems, etc. To leverage the information contained in graphs, it is essential to develop efficient techniques for representing graph-structured data numerically.

Traditional statistical and machine learning methods are desiged for extracting features from structured data on Eucildean space. For example, principal component analysis(PCA), uniform manifold approximation and projection (UMAP), and t-distributed stochastic neighbor embedding (T-SNE) are common techniques to reduce dimensions and capture features of data. Although these methods have achieved satisfactory performance on structured-data, they are hard to be generalized to graph-structured data, because they highly depend on properties of Eucildean space.

This challenge led to the development of techniques specifically for graph-based representation. There are two main types of these techniques: shallow embedding method and deep embedding method [6]. Shallow embedding methods use shallow encoder functions to map the original graph structure onto a Euclidean space and obtain the embedding matrix. If data is labelled, we can apply supervised learning algorithms, e.g. label propagation, to extract embeddings later used in a supervised task. However, labels are not available or only partly available in most cases, where we need unsupervised learning or semi-supervised learning to distill information about graph structure. These methods can be divided into distance-based method and outer product method further [6]. Generally, distance-based methods select a metric function that indicates distances between any pairs of nodes and optimize the function to generate embeddings. Representative distance-based methods include multi-dimensional scaling and laplacian eigenmaps. Outer product-based methods use matrix operations to evaluate the similarity between nodes. Most of early studies in graph embedding field adopt matrix factorization to reduce dimensionality of data while preserve the structure information [1]. Another mainstream outer product-based method is inspired by the development in natural language processing. Existing research generalizes the skip-gram word embedding framework to capture the graph embeddings, which has been proved to be efficient on many graph related tasks [7][8]. Node2vec [2], one of algorithms addressed in this paper, is also a variation of skip-gram-based method.

Node2vec is a semi-supervised algorithm whose goal is learning features from networks [2]. It transforms the graph embedding learning into a maximum likelihood optimization problem in a similar way to skip-gram architecture of word embedding learning. Likelihood calculation requires a clear definition of the neighborhood. Textual data has the intrinsic semantic order that can be naturally employed as word neighborhoods. However, graph-structured data has no explicit neighborhoods. Node2vec introduces the idea of the second-order random walk into graph neighborhood sampling strategy. The emphasis of node2vec model is easy to switch between breadth-first sampling (BFS) and depth-first sampling (DFS) by adjusting hyperparameters. Moreover, its computational complexity is less than classical BFS and DFS strategies. Because of its efficiency and great performance on graph embedding learning task, node2vec is an ideal choice among shallow embedding methods.

In recent years, a lot of studies have focused more on deep embedding method rather than the shallow one. Deep embedding method usually refers to algorithms that learn graph features via graph neural networks (GNN). The GNN is a class of artificial neural networks constructed for graph-structured data. Inspired by the success of convolutional neural networks (CNN) on grid data, many GNN architectures have been proposed to generalize the convolution operation on graphs. In this paper, we mainly focus on a method called graph convolutional networks (GCN) [4]. GCNs defines the graph convolution based on the graph Laplacian spectrum. It has achieved state-of-the-art performance on common graph related tasks, such as node classification, link prediction, etc.

We propose a novel method to extract embeddings from graphs in this paper. Our method is based on node2vec and GCNs. Motivated by the concept of meta learning, we regard the embeddings returned by node2vec as the meta information for GCNs. This prior knowledge helps to improve the quality of final graph embeddings, and therefore enhances the model performance in various tasks.

The rest of the paper is organized as follows. Section 2 reviews related works on graph embedding learning methods. We illustrate

SNACS '22, Master CS, Fall 2022, Leiden, the Netherlands

basic task notations and framework in Section 3. And then we describe approaches in detail in Section 4. Section 5 is an introduction to five open source data sets we use in the project. After that, we present our experimental set-up and results in Section 6. The paper ends with a conclusion section.

2 RELATED WORK

3 PRELIMINARIES

3.1 Feature Learning Framework

Feature learning in networks is regarded as a a maximum likelihood optimization problem[2]. We represent the given network as $G = (V, E)$, where V and E are nodes set and edges set respectively. Feature Learning task aims to find a projection $f : V \rightarrow R^d$, such that f optimizes the following objective function:

$$\max_f \sum_{u \in V} log\ Pr(N_s(u)|f(u))$$

Projection f allocates each node an embedding vector with length d. In other words, projection f can be formulated as a matrix of size $|V| \times d$. $N_s(u) \subset V$ defines *network neighborhood* for a node u with neighborhood sampling strategy S. Please be aware, this *network neighborhood* is not equivalent to the commonly used concept *local neighborhood* which only is determined by the graph structure. On the contrary, *network neighborhood* is generated based on both network structure and sampling strategy. Therefore, the above formula aims to maximize the log-probability of observing a network neighborhood $N_s(u)$ for a node u conditioned on its feature embedding representation, given by projection f.

To simplify the above optimization formula and task, two standard assumptions are made. One is conditional independence, which means observing a neighborhood node is independent of observing any other neighborhood node. Then according to the property of independent event, the formula in the objective function above can be written in the form of multiplication:

$$Pr(N_s(u)|f(u)) = \prod_{n_i \in N_s(u)}^{N} Pr(n_i|f(u))$$

The other is symmetry in feature space, which means each source node and its neighborhood node have a symmetric effect over each other in feature space. In the reference [2], the conditional likelihood of each *(source node, neighbor node)* pair in the network neighborhood is formulated as a softmax function:

$$Pr(n_i|f(u)) = \frac{exp(f(n_i) \cdot f(u))}{\sum_{v \in V} exp(f(v) \cdot f(u))}$$

Finally, the objective function can be rewritten using the above two assumptions in the form of:

$$\max_f \sum_{u \in V} [-log \sum_{u \in V} exp(f(u) \cdot f(v)) + \sum_{n_i \in N_s(u)} f(n_i) \cdot f(u)]$$

The first half of this formula $-log \sum_{u \in V} exp(f(u) \cdot f(v))$ is very expensive in computing. Therefore, a negative sampling method is applied to calculate it approximately [5]. For this objective function, stochastic gradient decent is applied to optimize it and obtain the projection f.

With a good designed network neighborhood sampling strategy S, the above optimization process could help to find projection f which makes the nodes embedding equipped with homophily and structural equivalence information [3], which could be taken advantage of in the downstream task.

The sampling strategy S is the most important part of feature learning framework. In this paper, second order random walk, the key point of $Node2Vec$, is applied as the sampling strategy. Before we dive into the details of $Node2Vec$ and second order random walk in section 4.1, let us have a glance to two commonly used downstream tasks, nodes classification and link prediction, which can be used to measure the performance of the nodes embedding generated by $Node2Vec$.

3.2 Downstream Task

Nodes embedding is widely used in many common tasks of networks, such as nodes classification and link prediction. These tasks, which take advantage of information from nodes embedding, are called downstream tasks. Therefore, the best way to examine whether the nodes embedding generated by Feature Learning Framework, such as $Node2Vec$, is to check the performance of nodes embedding on such downstream task.

In this paper, two downstream tasks are used.

Nodes Classification Given a graph $G = (E, V)$ with category labels of part of nodes, nodes classification task aims to classify the rest of nodes without labels into their category correctly.

Link prediction Given a subgraph $G' = (V', E)$ of the entire graph $G = (V, E)$, where $V' \subset V$ and the rest of edges are missing. Link prediction task aims to predict these missing edges correctly.

Nodes classification and link prediction are two basic but very important tasks in network analysis area. Therefore, it's of much importance to examine $Node2Vec$ whether works well in these two tasks.

The downstream tasks also need a model, such as a classifier like Logistic Regression Classifier or Graph Neural Network. Determining which downstream model is better is not our focus in this paper. In this paper, we mainly care about whether the nodes embedding generated from $Node2Vec$ can make downstream model work better. Therefore, in experiments section, we will experimentally show the performance improvement gained by applying GCN with nodes embedding from *node2vec* comparing to GCN without nodes embedding.

4 APPROACH

4.1 Node2vec

In section 3.1, we discussed feature learning framework. And node2vec is such a feature learning model with special designed sampling strategy S. There are two classic search algorithms, BFS and DFS, can be naively used as network neighborhood sampling strategy S. However, according to the reference [2], directly applying BFS as network neighborhood sampling strategy leads the nodes embedding to fully learn structural equivalence of networks. On the contrary, directly applying DFS as network neighborhood sampling strategy leads the nodes embedding to fully learn homophily of networks. Since structural equivalence and homophily are both important similarities for nodes [5], it's definitely not a good way

to throw away one of them. Therefore, we should use a method to combine the features of BFS and DFS, and make the sampling strategy capture these two similarities at the same time.

Inspired by random walk, the reference [2] comes up a second order bias random walk method, which combines the features from both BFS and DFS, as the network neighborhood sampling strategy of *node2vec*.

The basic random walk procedure with length l is formulated as following [2]. Given a source node u and let Let c_i denote the i-th node in the walk, starting with $c_0 = u$. Nodes c_i are generated by the following distribution:

$$p(c_i = x | c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & if\ (v,x) \in E \\ 0 & otherwise \end{cases}$$

where π_{vx} is the unnormalized transition probability between nodes v and x, and Z is the normalizing constant. In weighted graph, π_{vx} is usually set to the edge weight between node v and x, namely w_{vx}. In unweighted graph, $w_{vx} = 1$, that's to say, the next nodes to walk is generated with equal possibility in the local neighborhood. However, the basic random walk is still not enough to capture both structural equivalence and homophily similarities.

Therefore, a second order bias random walk method is proposed by the reference [2]. Consider a baisc random walk process just traversed edge (t, v) and now resides at node v. Now the walk process wants to decide the nodes to visit in the next step. So the unnormalized transition probability $\pi_{v,x}$ on edge (v, x) will be calculated. Particularly, the unnormalized transition probability is set as $\pi_{v,x} = \alpha_{pq}(t, x) \cdot w_{vx}$, where

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & if\ d_{tx} = 0 \\ 1 & if\ d_{tx} = 1 \\ \frac{1}{q} & if\ d_{tx} = 2 \end{cases}$$

p and q are two key hyperparameters introduced in this method, and d_{tx} represents the smallest distance between node t and node x. Notice that for each two step, d_{tx} can only be three possible values: 0, 1 and 2. When $d_{tx} = 0$, it means the possible next step node x will lead the process to walk back to node t again. When $d_{tx} = 1$, it means the possible next step node x will lead the process to walk to the nodes which are the other direct local neighbor of node t. When $d_{tx} = 2$, it means the possible next step node x will lead the process to walk to the new node and get rid of node t's ego network. Hyperparameters p and q are called return parameter and in-out parameter. p controls the possibility for the walk immediately step back to the previous node, and q controls the possibility for the walk to go out of the previous node's ego network. That's easy to find, when applying small p and large q, the walk process will prefer staying in the previous node's ego network, which means the walk process is more similar to BFS. On the contrary, when applying large p and small q, the walk process will prefer stepping out of the previous node's ego network, which means the walk process is more similar to DFS. Therefore, by well tuning p and q, we can obtain a method that can take advantage of the features from both DFS and BFS, and then can capture both structural equivalence and homophily similarities.

Up to now, the second order bias random walk has been formulated. Them, we should apply it in the features learning framework as the network neighborhood sampling strategy.

5 DATA

6 EXPERIMENTS

7 CONCLUSION

ACKNOWLEDGMENTS

REFERENCES

[1] HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. 2018. A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications. *IEEE Transactions on Knowledge and Data Engineering* 30, 9 (2018), 1616–1637. https://doi.org/10.1109/TKDE.2018.2807452

[2] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16).*

[3] Peter D Hoff, Adrian E Raftery, and Mark S Handcock. 2002. Latent space approaches to social network analysis. *Journal of the american Statistical association* 97, 460, 1090–1098.

[4] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations.*

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems* 26.

[6] Kevin P. Murphy. 2022. *Probabilistic Machine Learning: An Introduction.* MIT Press.

[7] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14).*

[8] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-Scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web (WWW '15).*