Noise Reduction in Network Embedding

Cong Li[1,2], Donghai Guan[1,2(✉)], Zhiyuan Cui[1,2], Weiwei Yuan[1,2], Asad Masood Khattak[3], and Muhammad Fahim[4]

[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
l8851870127@163.com, {dhguan,yuanweiwei}@nuaa.edu.cn, 565508802@qq.com

[2] Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210016, China

[3] College of Technological Innovation, Zayed University, Dubai, United Arab Emirates
Asad.Khattak@zu.ac.ae

[4] Institute of Information System, Innopolis University, Innopolis, Russia
m.fahim@innopolis.ru

Abstract. Network Embedding aims to learn latent representations and effectively preserves structure of network and information of vertices. Recently, networks with rich side information such as vertex's label and links between vertices have attracted significant interest due to its wide applications such as node classification and link prediction. It's well known that, in real world applications, network always contains mislabeled vertices and edges, which will cause the embedding preserves mistake information. However, current semi-supervised graph embedding algorithms assume the vertex label is ground-truth. Manually relabel all mislabeled vertices is always inapplicable, therefore, how to effective reduce noise so as to maximize the graph analysis task performance is extremely important. In this paper, we focus on reducing label noise ratio in dataset to obtain more reasonable embedding. We proposed two methods for any semi-supervised network embedding algorithm to tackle it: first approach uses a model to identify potential noise vertices and correct them, second approach uses two voting strategy to precisely relabel vertex. To the best of our knowledge, we are the first to tackle this issue in network embedding. Our experiments are conducted on three public data sets.

Keywords: Network embedding · Noise identification · Voting

1 Introduction

Networks naturally exist in a widely diversity of real world scenarios, e.g., citation paper in research areas, social network such as Face Book, Wei Bo. Directly analysis these networks may suffer the high computation and space cost [1]. One fundamental and effective solution is graph embedding. With such kind of vertex representations, the graph analytic tasks can be conducted efficiently in both time and space.

X. B. Zhai et al. (Eds.): MLICOM 2019, LNICST 294, pp. 109–120, 2019.
https://doi.org/10.1007/978-3-030-32388-2_10

A network can be regarded as a graph $G = (V_U, V_L, E)$. V_U represent unlabeled vertex set, and V_L represent labeled vertex set in a network, and E represent edge set, which is the relationship among the vertices.

Nowadays network embedding method can be divided into two categories based on whether the vertex's label is considered in the embedding procedure. In this paper, we focus on the networks containing labelled vertices. There are GCN [2], which proposes a multi-layer Graph Convolutional Network for semi-supervised node classification on graph and LANE [3] is also proposed to incorporate the label information into the attributed network embedding. However, all these methods based on a hypothesis that the labeled data is given ground-truth label, which is impossible in real word applications. Take user recommend system as an example, we recommend some users to a target user for they have same label, which can represent their age, gender, interest, and they are linked by n-hop. Obviously, mislabeled data impact the performance of a recommend system. Mislabeled data can be divided into two classes, mislabeled vertex and mislabeled edge [4].

In this work, we focus on mislabeled vertex, leaving the latter to future work. Given a network, our goal is to design a framework which can efficiently relabel mislabeled data to obtain more reasonable embedding. There are two main issues. First, we need to identify mislabeled vertex as candidates for relabel. Second, after we relabel the vertex, how do we use relabel data sets. In other words, how do we combine relabel methods with network embedding framework.

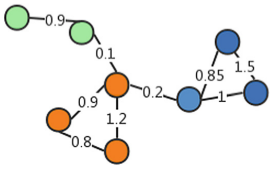

Fig. 1. Weight between vertices

In this paper, we proposed an effective framework tackles above mentioned issues. For identify and correct, we use the cost of misclassification to estimate potential noise vertex and correct. And, we consider a popular noise correct method, majority voting, to relabel vertex. Considered characteristic of network, after re-label process, we also propose a label propagation process [5, 6], to gain more labeled vertex. The intuition behind propagation process is the following: two vertices are more similar if they are connected by an edge with larger weight named first-order proximity. Same color circles belong to one class (see Fig. 1). In the context of our work, we use cosine similarity to represent weight of an edge. As the process progresses, our framework will gain more and more vertices with reasonable label. Meanwhile, with the more reasonable labeled vertices, the network embedding framework able to train more reasonable vertex embedding.

The chiefly contributions of this paper are follows:

(1) We are the first to tackle noise issue in network. Our framework can optimize the network embedding's performance by accurately identify and relabel mislabeled vertex.
(2) We study the impact of label noise and edge noise on network embedding's performance. Combing our method and network embedding, we effective eliminate the impact of label noise.
(3) We verify the effectiveness of our method by conducting experiments on three public citation datasets. And the results prove our method can improve embedding's performance in task node classification.

The rest of the paper is organized as follows: Sect. 2 introduces the related work of network embedding and noise relabel. Our proposed method will be introduced in Sect. 3. Section 4 introduce datasets and show experiments and results. Section 5 summarizes our paper, and we discuss our future work.

2 Related Works

In this paper, we focus on identifying mislabeled vertex and relabeling them so as to improve the network embedding's performance in task of node classification. In this section, we review the literature in two relevant areas: network embedding and noise reduction.

2.1 Network Embedding

Many complex applications take the form of networks. Most exist analysis methods for network embedding are high computation and space cost. Early study [7] aims to preserve the network's structure information, such as first-order, second-order and high-order proximities by matrix factorization. In series of matrix factorization models, Singular Value Decomposition (SVD) is commonly used in network embedding due to its optimality for low-rank approximation [8] These methods always have a poor performance when vertex's number grow up. Inspired by the recent success of natural language processing(NLP)some researchers [9, 10] start to embed network use a random walk or biased walk to sample paths from networks, and then apply skip-gram [11] on these walks to preserve network's information. Obviously, methods mentioned before only consider network topology. In nowadays networks always are accompanied rich side information [8], such as vertex label, signed link between vertices. With vertex label, network embedding can train in a semi-supervised manner. On the one hand, GCN [2], considers the problem of classifying nodes (such as documents) in a graph (such as a citation network), where labels are only available for a small subset of vertices. On the other hand, SNE [12, 16], which exploits the network structure and user attributes simultaneously for network representation learning. In this paper, we adopt GCN as an example embedding framework.

GCN considers a multi-layer Graph Convolutional Network (GCN) with the following layer-wise propagation rule:

$$\mathrm{f}(H^{(l)}, A) = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{1}$$

Where W is a weight matrix for the l-th neural network layer and σ (·) is a non-linear activation function like the ReLU. And $\hat{\mathrm{A}} = \mathrm{A} + 1$, where I is the identity matrix and D is the diagonal node degree matrix of $\hat{\mathrm{A}}$. A is symmetric adjacency matrix (binary or weighted). $H^0 = X$, which is a matrix of node feature vectors X_i. Then the forward of GCN take the simple form:

$$Z = f(X, A) = soft\max\left(\hat{A}\mathrm{ReLU}\left(\hat{A}XW^{(0)}\right)W^{(1)}\right) \tag{2}$$

For a semi-supervised multiclass classification, GCN evaluate the cross-entropy error over all labeled examples:

$$Loss = -\sum_{l\in y_L}\sum_{f=1}^{F} Y_{lf}\ln Z_{lf} \tag{3}$$

Where y_L is set of node indices that have labels.

2.2 Noise Reduction

Since ground-truth labelled data is often expensive to obtain and manually label all training data is inapplicable [13]. In real applications, labelled data always contains mislabeled data. A popular noise reduction method is that relabel vertex by its nearest neighbors which can calculate by similarity function such as cosine similarity:

$$similarity = \cos(\theta) = \frac{A \bullet B}{\|A\|\|B\|} \tag{4}$$

where A, B representative vector. • is dot product. Another noise data relabel framework aims to find and relabel mislabeled data, which usually consists of two primary processes: a classification system and voting system, and k-fold cross-validation [15]. For classification system, the majority work is build classifiers based on current features (embedding) and predict label for all labelled data. Then voting system votes a label for a target data. Based on the vote strategy, there are two voting strategies: majority vote and consensus vote [17]. Majority voting choose a label, which are predicted by more than half classifiers. Consensus voting choose label which are predicted by all the classifiers [15].

In network embedding scenario, our work is distinct different from the most noise relabel frameworks in two ways. First the features (embedding) trained by classifiers will be updated after iteration in our framework, which is constant in traditionally relabel framework. Second, as we mentioned before at Sect. 1, propagation process

able to generate more reasonable labelled vertices. The detailed framework is discussed in next section.

3 Proposed Method

Given a network dataset with mislabeled vertices, we propose methods to identify and relabel vertex to optimize network embedding's performance. In our framework, we relabel the vertices after the training of GCN. The two processes, noise reduction and graph embedding, reinforce each other. Next, we introduce detailed of our proposed method in two subsections: relabel, propagate. The main algorithm's steps of our proposed method are illustrated as bellows.

Algorithm 1 : Noise Reduction In Network Embedding

Input: Network data with mislabeled vertices

Output: Cleaned label set and reasonable vertex's embedding

1: **for** $i = 0 -> n$

2: **repeat**

3: Obtain vertex embedding on current network data by GCN

4: Relabel labelled vertex

5: Propagate label to vertex's neighbor

6: **until** node performance is steady or exit in step 4

7: **end for**

The detail of step 4, step 5: relabel labelled vertex and propagate label will discuss in next section. We proposed two ways to implement step 4.

Noted that our method can combine any semi-supervised network embedding framework.

3.1 Relabel

We propose two methods to implement relabel: Identify and Correct, Vote.

In this work, each labelled vertex v (y, x), where x is vertex's embedding and y is vertex's label in current iteration. We use $1 - P(y \mid x)$ to estimate potentially mislabeled vertex. The conditional probability $P(y \mid x)$ is probability that vertex is a sample of class 'y'. At each iteration, we only choose the most likely mislabeled vertex to relabel. Although in our framework more iterations mean more time consuming, we can reduce noise ratio and improve network embedding's performance.

The intuition behind using this strategy is the following: if there is a robust classifier, the conditional probability can successfully represent which class a vertex should be in current iteration. Generally speaking, on the one hand, if $P(y \mid x)$ is great than 0.9,

it may be safe to label vertex as y, on the other hand, if $P(y \mid x)$ is less than 0.1, it may be safe to assume the vertex is mislabeled.

For example, let us consider a three classification problem. {1, −1, 0} is class set. There is a vertex is labelled as '1'. And a trained classifier calculates the conditional probability for each class is $P(1 \mid x) = 0.1$, $P(-1 \mid x) = 0.2$, $P(0 \mid x) = 0.7$. As we mentioned above, this vertex is regarded as a mislabeled vertex (identify), and we can simple assume this vertex should be class '0' (correct). Algorithm 2 gives a detailed process of Identify and Correct.

Algorithm 2: Identify and Correct

Input: Vertex embedding set X with noise label set Y

Output: Cleaned label set

1: Train a base-classifier on X

2: Compute M, an ordered set of potentially mislabeled vertex

3: **if** $M \neq \{\ \}$

4: a. Select top n vertex

5: b. Correct label base on conditional probability P (y | x)

6: **exit**

There are two shortcomings about Identify and Correct. First, one classifier may not effective identify mislabeled vertex, especially as our method process. In on hand, it means we only identify portion of noise. In other hand, this method will only able to identify mislabeled vertices which are easy to identify. Second, considering we can't have enough absolutely clean labelled data to generate a robust classifier, which means in correct step the performance is not powerful as we expected.

As we mentioned above, in order to reduce noise effect in one classifier, we propose other method: Vote. For vote, we process majority vote process and consensus vote on labelled data set. The detailed showed in Algorithm 3.

We score all vertices for each class. The score can be defined as follows:

$$Score_{v,y} = \sum_{i}^{n} \theta_i \tag{5}$$

where $\theta_i = 1$ when $f_i(X_v) = y$, otherwise $\theta_i = 0$. X_v, y are embedding and label for vertex v. f_i is one of classifiers and n is number of classifiers. In majority vote, for each vertex, if there is a score of one class is not less than half of n, we relabel vertex by this class. In consensus vote, we only relabel vertex which score for a class is equal n.

We divide labelled data into n groups and select n − 1 groups data as the training samples and the rest as the test samples. We train a set of classifiers based on current training samples and predict label of vertex. But unlikely traditional process, we not only predict target vertex, we also predict its fist-order neighbors as references to help

us decide a final label to relabel target vertex. Because they have a significant odds share same label. The detail is showed in Algorithm 3.

Using above methods, at a specific iteration, we gain some reasonable labelled vertex. Before retrain embedding we propagate the label to its neighbors, which will be discussed in next section.

Algorithm 3: Vote

Input: vertex embedding X with noise label Y

Output: Cleaned label

```
Divided data to train and test sample
Train a set of classifiers on train sample, named C
Initialize a vote pool, name P
for vertex in test sample:
    a. Predict vertex's label and add results to P
    b. Find vertex's neighbors, named N
    for neighbor in N:
        if neighbor labelled:
            Add neighbor's label to P
        else:
            Predict and add results to P
    end for
    Vote a label for vertex by P
end for
```

3.2 Propagate Label and Retrain Embedding

Propagating label is a trade-off between more labelled vertices and more noise labelled vertices. Generally speaking, in one hand, propagating label means labeled vertex set is enlarged. In other, there are odds we propagate wrong label to its neighbors. In order to reduce mislabeled vertex by propagating, two criterions are proposed. First-order neighbor and high similarity between vertices. Then propagation set of V_i can be defined as follows:

$$S = \left\{ V_i, V_j \,|\, f\left(E_i, E_j\right) \geq \alpha \right\} \tag{6}$$

where V_j is first-order of V_i. f can be any similarity function, such as inner product or the cosine similarity. A larger similarity implies that the two vertices may have a higher propensity to be linked [7] in data set. In this paper, we use cosine similarity to estimate similarity between vertices. And α is a threshold, which will provide lower bound for

similarity. At every iteration of label propagation, vertex adopts the label shared most by its first-hop neighbors T_i. Hence,

$$C_i = \arg\max_C \{j \in T_i \mid C_i = C\} \tag{7}$$

Then we obtain more reasonable embedding by network embedding framework to relabel more mislabeled vertex at next iteration.

4 Dataset and Experiments

4.1 Dataset

In this paper, we use three public citation networks dataset: Citeseer, Cora and Pumbed, and more detail network statistics showed in Table 1. Each of dataset contains sparse bag-of-words feature vectors for each document, which are linked by citation links. The unweight and undirected links are treated edges in a network.

Table 1. Data statistics

Dataset	Nodes	Edges	Classes	Features	Label rate
Cora	2,708	5,429	7	1,433	0.052
Citeseer	3,327	4,732	6	3,703	0.036
Pumbed	19,717	44,338	3	500	0.003

4.2 Experiments

The experiments are conducted on above mentioned datasets. First, we verify noise impact. Then we use our framework to identify and correct mislabeled vertex to improve network embedding's performance. All experiments are evaluated by reasonable metrics. More detail discussion and results in following section.

In this paper, we evaluate the network embedding by embedding's performance in node classification, and use accuracy to measure the classification performance.

4.2.1 Noise Impact

We verify noise impact on three different types noise: label noise, edge noise and label with edge noise.

We randomly add noise into dataset from noise rate 0 to 0.4. As we expected, no matter what type of noise, higher noise ratio produces lower performance in node classification task (see Fig. 2). Specially, because edge noise will change the structure of network, its impact is greater than label noise. When label noise ratio increase, the data set Pubmed still has acceptable performance in node classification (see Fig. 2). But, if there are label and edge noise exists in data set, its performance significant decline (see Figs. 3 and 4).

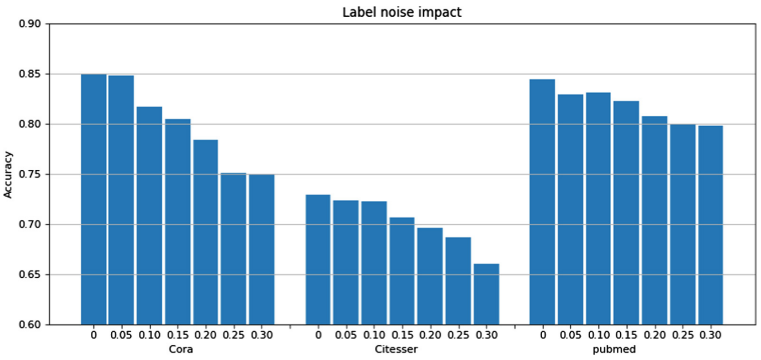

Fig. 2. Label noise impact

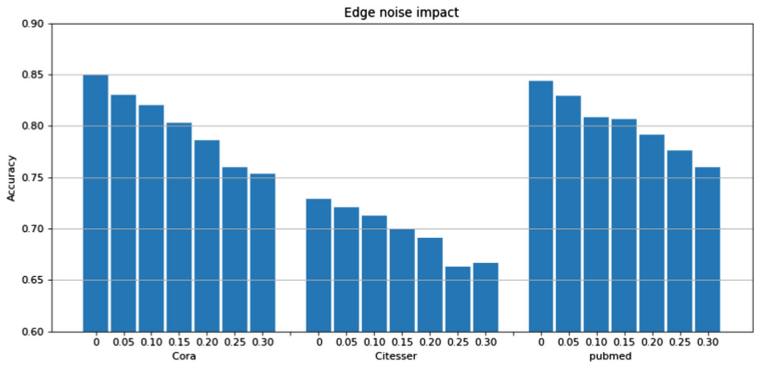

Fig. 3. Edge noise impact

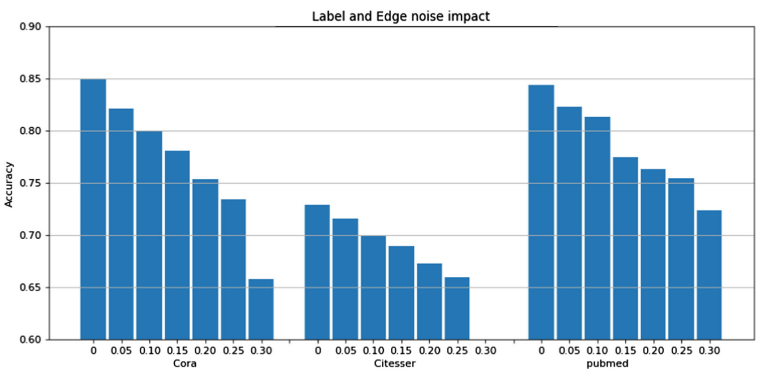

Fig. 4. Label and edge noise impact

4.2.2 Relabel

In our experiments, we initially add noise ratio of 0.1, 0.2, 0.3, 0.4, into dataset. We conduct our method to data set and in each noise ratio we both can improve network embedding's performance in node classification task.

As results showed in Table 2, embedding performance in node classification is improved at each noise ratio. Specifically, compared to the accuracy with specific noise, our method improves the node classification accuracy by 2% to 6%.

Table 2. Perform accuracy on dataset Cora

Noise ratio	Method		
	Correct	Majority vote	Consensus vote
0.1(0.803)	**0.815**	**0.822**	**0.826**
0.2(0.786)	**0.790**	**0.803**	**0.798**
0.3(0.724)	**0.767**	**0.745**	**0.749**
0.4(0.679)	**0.716**	**0.703**	**0.733**

Among three methods, correct performance lower than vote method. There is a straight forward understanding and guess and we will verify at our future work. In a citation networks dataset paper is linked by same or similar discipline. We still take a three classification problem as an example. A class set is {Computer Science, Mathematics, Physics}. As we know, mathematics is the basic discipline of computer science, especially in machine learning area. Assume a computer science paper is mislabeled as Physics. After a vote process, if it is labelled as Mathematics. For data set this vertex (paper) is still mislabeled, but for label information a vertex containing is become more reasonable, which leads to gaining a more robust classifier and a better performance in node classification. And this also explains why the noise ratio is not significant reduced and the performance is improved (Table 3).

Table 3. Performance accuracy on dataset Citeseer

Noise ratio	Method		
	Correct	Majority vote	Consensus vote
0.1(0.721)	**0.727**	**0.742**	**0.727**
0.2(0.707)	**0.711**	**0.714**	**0.710**
0.3(0.658)	**0.676**	**0.682**	**0.673**
0.4(0.602)	**0.628**	**0.687**	**0.627**

5 Summary and Future Work

In this paper we propose a framework to tackle noise label in network embedding area. Our framework incorporates network embedding and noise reduction, which reduces noise ratio of dataset to obtain more reasonable embedding by iterative relabeling

vertex and retraining embedding. Unlike the traditional noise reduction algorithms, our method handles the data with rich side information: label, and edge learnt representations (vertex embedding), and it is carefully designed to exploit the usefulness brought by these two characteristics. First, to exploit the edge information, a label propagation process is considered in first-hop neighbors. Second, the noise reduction method and graph embedding process are run together in iteration manner. Relabeling the label and retraining embedding at each iteration process. For experiments, we first verify the noise impact with different noise ratio and noise type over three public citation networks. Then we have evaluated our proposed method over two public citation network dataset which performance is significant decline when label noise grows. Experiments show that our framework is effectiveness.

For future work, in one hand, we will focus on reducing networks structure's impact on network embedding. Further-more, we will also focus on tackle there are edge and label noise issue by combining the method for edge and label. In other hand, the guess we method at Sect. 4, we aim to give a mathematical proof and verify on more dataset.

Acknowledgements. This research was supported by Natural Science Foundation of China (Grant no. 61572252, 61672284). Meanwhile, this research work was supported by Zayed University Research Cluster Award \# R18038.

References

1. Cai, H.Y., Zheng, V.W., Chang, K.: A comprehensive survey of graph embedding: problems, techniques and applications. IEEE Trans. Knowl. Data Eng. **30**, 1616–1637 (2018)
2. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional network. In: 5th International Conference on Learning Representations (2017)
3. Huang, X., Li, J., Hu, X.: Label informed attributed network embedding. In: Tenth ACM International Conference on Web Search and Data Mining. ACM (2017)
4. Li, Z., Fang, X., Sheng, O.R.L.: A survey of link recommendation for social networks: methods, theoretical foundations, and future research directions. Social Science Electronic Publishing (2015)
5. Zhang, X.K., Ren, J., Song, C., et al.: Label propagation algorithm for community detection based on node importance and label influence. Phys. Lett. A **381**, 2691–2698 (2017). https://www.sciencedirect.com/science/article/abs/pii/S0375960117305868
6. Wang, F., Zhang, C.: Label propagation through linear neighborhoods. IEEE Trans. Knowl. Data Eng. **20**(1), 55–67 (2007)
7. Ahmed, A., Shervashidze, N., Narayanamurthy, S., et al.: Distributed large-scale natural graph factorization (2013)
8. Peng, C., Xiao, W., Jian, P., et al.: A survey on network embedding. IEEE Trans. Knowl. Data Eng. 1 (2018)
9. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2014)
10. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2016)

11. Mikolov, T., Chen, K., Corrado, G., et al.: Efficient estimation of word representations in vector space. Computer Science (2013)
12. Yuan, S., Wu, X., Xiang, Y.: SNE: signed network embedding. In: Kim, J., Shim, K., Cao, L., Lee, J.-G., Lin, X., Moon, Y.-S. (eds.) PAKDD 2017. LNCS (LNAI), vol. 10235, pp. 183–195. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57529-2_15
13. Cai, H., Zheng, V.W., Chang, C.C.: Active learning for graph embedding (2017)
14. Frenay, B., Verleysen, M.: Classification in the presence of label noise: a survey. IEEE Trans. Neural Netw. Learn. Syst. **25**(5), 845–869 (2014)
15. Chen, K., Guan, D., Yuan, W., et al.: A novel feature selection-based sequential ensemble learning method for class noise detection in high-dimensional data. In: International Conference on Advanced Data Mining and Applications (2018)
16. Wang, S., Aggarwal, C., Tang, J., et al.: Attributed signed network embedding. In: CIKM 2017 (2017)
17. Ruta, D., Gabrys, B.: Classifier selection for majority voting. Inf. Fusion **6**(1), 63–81 (2005)