An Active Noise Correction Graph Embedding Method Based on Active Learning for Graph Noisy Data

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

[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China
565508802@qq.com, {dhguan,yuanweiwei}@nuaa.edu.cn, 18851870127@163.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

Abstract. In various scenarios of the real world, there are various graph data. Most graph structures are confronted with the problems of complex structure and large consumption of memory space. Graph embedding is an effective method to overcome such challenges, which converts graph structure into a low-dimensional dense vector space. In the real world, label acquisition is expensive, and there may be noise in the data. Therefore, it is important to find valuable noise nodes as much as possible to improve the performance of downstream task. In this paper, we propose a novel active sampling strategy for graph noisy data named Active Noise Correction Graph Embedding method (ANCGE). Given the label budget, the proposed method aims to use semi-supervised graph embedding algorithm to find valuable mislabeled nodes. ANCGE measures the value of noise nodes according to their representativeness and influence on the graph. The experimental results on three open datasets demonstrate the effectiveness of our method and its stability under different noise rates.

Keywords: Graph embedding · Noisy label · Active correction · Active learning

1 Introduction

In various scenarios of the real world, there are various network structures that represent the relationship between objects. For example, social graphs in social media networks, citation maps in research fields [3]. In computer science, the network structure is represented as a graph structure containing nodes and edges. However, most graph structures have the characteristics of huge structure and large space overhead, so the computational task is very heavy.

X. B. Zhai et al. (Eds.): MLICOM 2019, LNICST 294, pp. 419–433, 2019.
https://doi.org/10.1007/978-3-030-32388-2_36

Graph embedding is an effective method to solve this problem. Graph embedding transforms graph structure information into low-dimensional dense real vector and maps it to a low-dimensional latent space. Moreover, graph embedding can maximumly preserve the structure information and attributes of the graph, which be used for input of existing machine learning algorithms. Through graph embedding, various applications such as node classification, node recommendation, and link prediction can be performed on the graph [2].

According to whether the graph nodes used for training include labels and whether they are completely included, the graph embedding algorithm can be divided into three categories: unsupervised, semi-supervised and supervised. In the real world, the acquisition of labels is expensive, and it is generally difficult to obtain labels for all nodes in the graph structure. In order to make use of the information of the node label without too much cost, a small number of node labels will be acquired in practice. Therefore, for a scenario where such a small number of nodes have labels and most of the nodes are unlabeled, a semi-supervised graph embedding algorithm [9,21] can be employed.

But in real-world scenarios, the label data in the graph structure is not always correct. There may be a certain amount of noise in the labeled graph data. Using these noisy node labels for training can affect the performance of graph embedding and classifiers, which in turn reduces the accuracy of node classification. Therefore, it is very important to solve the classifier degradation under noisy scenes in the node label [11]. However, using domain experts to modify all noise labels is not only time-consuming, but also economically expensive. Therefore, given the label budget, it is very important to select the nodes that need expert marking in the labeled nodes to maximally find the noise nodes that have the greatest impact on the graph structure and improve the performance of the node classification. Active learning [15] can solve this problem very well.

In this paper, we propose an active corrective graph embedding method (ANCGE) based on active learning for graph noisy data. Given the label budget, we use semi-supervised graph embedding algorithm to find more noise in labeled nodes by active learning. And when the active corrective graph embedding method chooses the noise nodes which need to be corrected, the noise nodes which have large amount of representativeness and great influence on the graph are selected as far as possible by considering the structure of the graph.

The specific details of our method are as follows: Firstly, the semi-supervised graph embedding algorithm GCN [9] is used to train and generate graph embedding from noisy labeled graph data. Secondly, graph embedding is used to detect noise in graph data, and the noise probability scores of all nodes in graph data are calculated. Given the label budget, the node most likely to be labeled incorrectly is selected. Thirdly, by calculating the graph centrality score [1] in the representative query criteria of active learning, we can find out the nodes with the greatest amount of information from the selected nodes with high probability of noise, and then give these nodes to the domain experts for correction. Finally, the active corrected nodes are added to the labeled node set, and then

the training set is updated. The updated data is used to retrain the graph embedding, and then the node classifier is trained.

The above is a complete label correction and classifier training process, and then we iteratively proceed with the above steps. As the process progresses, graph embed-ding will produce more and more accurate node embedding, which will provide more information for the training of labeled nodes, and the performance of classifier will be improved very well.

The contributions of this paper are summarized as below:

- We propose an active corrective graph embedding method on graph noise data, which can effectively find and correct the noise in graph data, and then optimize the performance of graph embedding.
- We combine the noisy possibility with the graph centrality to find out the nodes which are not only noisy but also have great influence on the graph structure. In the case of a small number of label requests, the performance of the classifier can be improved as much as possible.

The rest of this paper is organized in the following way. Section 2 reviews the literature related to graph embedding, noise data learning and active learning. In Sect. 3, we describe the framework of the proposed active noise correction graph embedding algorithm, as well as the noise detection correction graph embedding algorithm (NDCGE) and active learning correction graph embedding algorithm (ALCGE). In Sect. 4, the experimental results are analyzed. Finally, we summarize the paper in Sect. 5.

2 Related Work

2.1 Graph Embedding

Graph embedding converts the graph data into a low dimensional space in which the graph structural information and graph properties are preserved.

Matrix factorization is an early algorithm for graph embedding. It based graph embedding represent graph property in the form of a matrix and factorize this matrix to obtain node embedding. There are several common algorithms of matrix factorization based graph embedding, such as LLE, Laplacian Eigenmaps and GraRep [5,19].

After that, embedding techniques using random walks on graphs to obtain node embeddings have been proposed: DeepWalk [13] and node2vec [6] are two examples. Deepwalk is an online graph embedding algorithm based on local information of nodes. Given a network, it first generates random walks to get the sequence of nodes. DeepWalk regards each node as a word and the sequence of nodes as a sentence, and then uses Skip-Gram [10], a typical training word vector model, to train the embedding of nodes. Node2vec model is an extension of DeepWalk. By improving the random walk strategy of DeepWalk model, node2vec can generate higher quality sequence of nodes.

In addition, LINE [17] is also a network representation learning model for large-scale networks. LINE defines the first-order proximity and the second-order

proximity between nodes in social networks. SDNE [20] introduces typical deep neural networks into graph embedding for the first time. It jointly optimizes the first-order proximity and second-order proximity, which solves the challenge of high nonlinearity.

But this method can be computationally expensive for large sparse graphs. Graph Convolutional Network (GCN) solves this problem by defining a convolution operator on graph. It is an extensible semi-supervised learning method based on graph structured data. And it has good classifier performance in node classification and other applications. So in this paper, we use GCN as a graph embedding framework.

2.2 Noisy Data Learning

In the real world, the acquired data will inevitably be affected by noise. There are many methods to deal with noise in the existing literature.

In the past, when researchers dealt with noise, one was to label an example with multiple non-expert labels. The EM algorithm of Dempster et al. [4] is a popular procedure for finding maximum likelihood estimates of parameters where the model depends on unobserved latent variables. EM algorithm is shown to provide a slow but sure way of obtaining maximum likelihood estimates of the parameters of interest. In another study in the field of natural language processing, Snow et al. [16] concluded that multiple inexpensive tags might be preferable to an expert.

In addition, in the literature [8,18], the method uses classifier prediction to detect noise. It trains the classifier with labeled noise data, and then deletes instances of error classification. However, this method deletes a large number of instances, which will reduce a lot of training data and degrade the performance of the classifier. How-ever, the above methods can't effectively find noise nodes in noisy graph data and make use of noise label.

Therefore, in our method, we can iteratively find the noise nodes from the graph data, and then correct the noise data. And we can train the classifier from the corrected data to improve the performance of the classifier.

2.3 Active Learning

Given the label budget, active learning can find those nodes with large amount of information to be corrected by experts, and then improve the performance of the classifier. According to the query strategy, active learning algorithms can be divided into three categories: the heterogeneity based, the performance based and the representativeness based [15].

There are three sub-categories of active learning based on heterogeneity. Uncertainty Sampling is probably the simplest and most commonly used query framework. In this framework, an active learner queries the instances about which it is least certain how to label. Another, more theoretically-motivated query selection framework is query-by-committee (QBC) algorithm. Each committee member needs to vote on the label being queried and then use the instance

they most disagree as the most informative query. The last method is Expected Model Change. Its main idea is to label the most different instances from the current known models. If we know the label of the current model, it will bring the greatest change to the current model.

Performance-based active learning method includes two sub- categories. One method is expected error reduction and it uses the remaining unlabeled instances to estimate the expected future error of the model, and uses the smallest expected future error to query the instances. The other method is variance reduction and this method indirectly reduces generalization error by minimizing output variance.

The Representative-Based active learning method [7] can select representative unlabeled instances to query their labels. It can explore unknown areas of data and then avoid non-representativeness of query labels. So in our method, we use the representative query strategy to select nodes with large amount of information. And we evaluate the representativeness by calculating the center score of PageRank centrality [14] for each node.

3 Proposed Approach

In the first two sections of this chapter, we first propose two label correction algorithms: noise detection correction graph embedding algorithm and active learning correction graph embedding algorithm. In the third section, we integrate the two algorithms and propose an active noise correction graph embedding algorithm based on active learning. Next, we will introduce the details of three noise label correction algorithms.

3.1 Noise Detection Correction Graph Embedding Algorithm

Algorithm 1 gives the relevant input and specific flow of the noise detection correction graph embedding algorithm. The algorithm can effectively detect the noise nodes in the labeled graph data, and iteratively correct the most likely noise nodes. The performance of the classifier can be greatly improved by using the updated label set to relearn the classifier.

The details of the algorithm are as follows: Firstly, we input graph embedding $X = \{X_1, X_2, \ldots, X_n\}$ of noisy labeled graph data generated by GCN algorithm and label correction budget B given by the algorithm. The first step is to train a classifier based on graph embedding X. Then set up a node set C to store corrective labels.

For a graph dataset with k categories of labels, the conditional probabilities of each label are $P_1(y|x), P_2(y|x), \cdots, P_k(y|x)$. Conditional probability $P(y|x)$ is the probability that the label of a node is y when the graph embedding is X. For the calculation of conditional probability $P(y|x)$, we adopt one-against-one SVM multi-classification method.

So in the noise correction algorithm, the third step is to find the corresponding label y in the label matrix Y. By calculating the conditional probability

Algorithm 1. Noise Detection Correction Graph Embedding(NDCGE)

Input: Graph embedding X, label budget B

Output: Updated labeled set L and retrained classifier

1. Train a base-classifier on X
2. Set of nodes with corrective labels $C = \{\}$
3. Compute the possibility of Label y, $P(y|x)$
4. Compute node noise probability $1 - P(y|x)$
5. Generate an ordered set M of potentially mislabeled nodes, and $M \bigcap C = \{\}$
6. Generate M_n, the top n nodes from M for label correction
7. Generate the labels $y = \{y_1, y_2, \ldots, y_n\}$, and update Y
8. $C = C \cup M_n$
9. Retrain the classifier
10. While $M \neq \{\}$ and the number of C is less than B : Repeat steps 3–9

of the label y under embedding X_i generated by GCN algorithm, we can get the possibility that the label of the node v_i is y. We can calculate the noise probability of the node by formula $1 - P(y|x)$.

Then the fifth step is to generate a node set M sorted according to the noise possibility, and ensure that the intersection of M and C is empty. Then we select M_n, the top n nodes about the noise possibility from M for label correction. The seventh step of the algorithm is to generate the updated label set $y = \{y_1, y_2, \cdots, y_n\}$, and then update the label matrix Y of the label graph node set G_L. Next, update the node set C that used to store the corrective label and add M_n to C. Finally, retrain the classifier about graph embedding. When node set M is not empty or the number of node set C is less than label budget B, repeat steps 3–9 and iterate the algorithm until it stops.

Given the label budget, the noise detection correction algorithm can iteratively and effectively correct the noise in the graph data, and relearn the classifier according to the updated label to improve the performance of the classifier. However, the algorithm only considers the influence of noise nodes in the data set, and does not consider the structure of the graph data itself. The situation of graph nodes themselves is different, and the information and representation of each node are different. Therefore, in the next section, we propose an active learning correction graph embedding method based on active learning.

3.2 Active Learning Correction Graph Embedding Algorithm

Our algorithm inputs are not independent identically distributed data but connected graph structures. So we should consider edge-to-edge connections between nodes when using active learning strategies. Therefore, we adopt a representativeness-based AL query criteria [7]. We use graph centrality $\phi_{(centrality)}$ as a representativeness measurement. This method uses graph structure to measure the representativeness of nodes by calculating the graph

centrality score of nodes. Graph centrality is first proposed in [12]. The existing graph centrality methods include classical methods and eigenvector-based methods. Because the eigenvector-based method is superior to other methods, we use PageRank centrality [14] to calculate graph centrality. Algorithm formula of PageRank is as follows:

$$x = (I - \alpha AD^{-1})^{-1} I = D(D - \alpha A)^{-1} I \tag{1}$$

Where A is the adjacency matrix of the graph and D is the diagonal matrix of the graph. There is a range of values for A, which is less than the reciprocal of the maximum eigenvalue of AD^{-1}. For undirected graphs, the value of α is 1, and for a directed graph, the value depends on the calculation. The calculation formula for the PageRank centrality of the candidate node v_i to be calculated is:

$$\phi_{centrality}(v_i) = \rho \sum_j A_{ij} \frac{\phi_{centrality}(v_j)}{\sum_k A_{jk}} + \frac{1-\rho}{N} \tag{2}$$

where ρ is the damping parameter.

Algorithm 2. Active Learning Correction Graph Embedding(ALCGE)

Input: Graph embedding X, label budget B

Output: Updated labeled set L and retrained classifier

1. Train a base-classifier on X
2. Set of nodes with corrective labels $C = \{\}$
3. Compute the possibility of Label $y, P(y|x)$
4. Compute node noise probability $1 - P(y|x)$
5. Generate an ordered set M of node graph centrality, and $M \cap C = \{\}$
6. Generate M_m, the top m nodes from M for label correction
7. Generate the labels $y = \{y_1, y_2, \cdots, y_m\}$, and update Y
8. $C = C \cup M_m$
9. Retrain the classifier
10. While $M \neq \{\}$ and the number of C is less than B : Repeat steps 3–9

Algorithm 2 gives the relevant input and specific flow of the active learning correction graph embedding algorithm. Its steps are basically similar to those of Algorithm 1, except that steps 5–8. Then the fifth step is to generate a node set M sorted according to node graph centrality, and ensure that the intersection of M and C is empty. Then we select M_m, the top m nodes about the node graph centrality from M for label correction. The seventh step of the algorithm is to generate the updated label set $y = \{y_1, y_2, \cdots, y_m\}$, and then update the label matrix Y of the label graph node set G_L. Next, update the node set C that used to store the corrective label and add M_m to C.

Given the label budget, the active learning correction algorithm can iteratively and effectively correct the large amount of information and representative nodes in the graph data, and relearn the classifier according to the updated label to improve the performance of the classifier. However, contrary to Algorithm 1, Algorithm 2 only considers the structure of the data itself, and does not consider the noise problem in the graph data set. In the next section, we propose an active noise correction graph embedding method based on active learning by combining the advantages of the two algorithms.

3.3 Active Noise Correction Graph Embedding Framework

Algorithm 3 gives the relevant input and specific flow of the active noise correction graph embedding algorithm. It is the fusion of Algorithm 1 and Algorithm 2, and it is the same as step 1–6 of Algorithm 1.

Algorithm 3. Active Noise Correction Graph Embedding(ANCGE)

Input: Graph embedding X, label budget B

Output: Updated labeled set L and retrained classifier

1. Train a base-classifier on X
2. Set of nodes with corrective labels $C = \{\}$
3. Compute the possibility of Label $y, P(y|x)$
4. Compute node noise probability $1 - P(y|x)$
5. Generate an ordered set M of potentially mislabeled nodes, and $M \cap C = \{\}$
6. Generate M_n , the top n nodes from M for label correction
7. Generate an ordered set N_n of node graph centrality with n nodes
8. Generate N_m, the top m nodes from N_n for label correction, and $m < n$
9. Generate the labels $y = \{y_1, y_2, \cdots, y_m\}$, and update Y
10. $C = C \cup N_m$
11. Retrain the classifier
12. While $M \neq \{\}$ and the number of C is less than B : Repeat steps 3–11

Then the seventh step is to generate a node set N_n sorted according to the graph centrality of n nodes. Then we select N_m , the top m nodes about the node graph centrality from N_n for label correction. In addition, make sure that $mislessthan$n. The ninth step of the algorithm is to generate the updated label set $y = \{y_1, y_2, \cdots, y_m\}$, and then update the label matrix Y of the label graph node set G_L. Next, update the node set C that used to store the corrective label and add N_m to C. Finally, retrain the classifier about graph embedding. When node set M is not empty or the number of node set C is less than label budget B, repeat steps 3–11 and iterate the algorithm until it stops.

Given the label budget, the active noise correction algorithm can iteratively and effectively correct the noise nodes of graph data, and select those noise

nodes with large amount of information and great influence on graph. Moreover, ANCGE can iteratively correct the selected nodes and relearn the classifier according to the updated labels to improve the performance of the classifier.

4 Experiments

The purpose of our design experiment is to: (1) prove the performance of our proposed active noise correction graph embedding model; (2) verify the stability of ANCGE model under different noise rates. We first introduce our experimental setup, and then analyze the experimental results.

4.1 Datasets

All experiments were conducted on three public citation network datasets, Cora, Citeseer and Pubmed [9]. Dataset statistics are summarized in Table 1. Each dataset contains a sparse bag-of-words feature vector of a document and a list of citation links between documents. Each document has a class label. And nodes are documents and edges are citation links. Label rate denotes the number of labeled nodes that are used for training divided by the total number of nodes in each dataset.

Table 1. Dataset statistics

Datasets	Nodes	Edges	Classes	Features	Label rate
Cora	2708	5429	7	1433	0.185
Citeseer	3327	4732	6	3703	0.150
Pubmed	19717	44338	3	500	0.025

4.2 Experimental Settings

For each dataset, we use 500 nodes for training, 500 nodes for validation and 1000 nodes for node classification testing. We randomly extract 500 nodes from non-training and non-testing nodes and use them as validation sets for all experiments to ensure the stability of performance in experiments.

For each data set, we set up 500 labeled nodes. The labeling rates of the three data sets are 0.185, 0.150 and 0.025. In addition, we use 10% label budget B, that is, which means B is 50 nodes. We divide label budget B into five groups, correcting 10 nodes each time, and iterate five times for each algorithm. In the experiment, we can simulate data sets in the real world by randomly modifying data labels and artificially increasing noise.

4.3 Evaluation Metrics

Node classification is a common task to evaluate the performance of graph embedding algorithms. Therefore, in this paper, we use node classification to verify the performance of the algorithm, and use the accuracy of node classification as an evaluation metrics.

4.4 Comparison of Label Correction Algorithm

In this section, we compare the advantages and disadvantages of three label correction methods and one random correction graph embedding method (RCGE) for node classification on three data sets. For each data set, the experiment set up 500 labeled nodes and 10% noise rate. The label budget is 50 nodes, which are iterated five times. The experimental results are shown in Figs. 1, 2 and 3. We use Random Label Correction Graph Embedding (RCGE) as the baseline. Given the label budget, it randomly selects the correct nodes from the data set each time, and iteratively corrects the nodes. The other steps are the same as those of other algorithms except the strategy of selecting nodes.

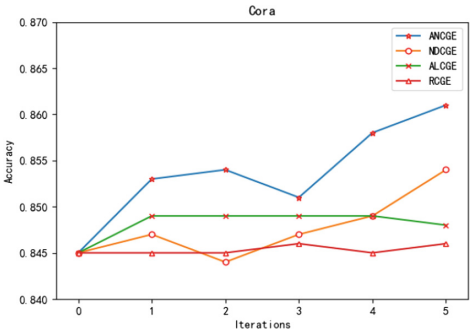

Fig. 1. Comparison of label correction algorithms on dataset Cora

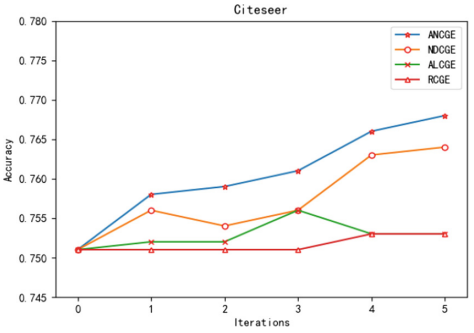

Fig. 2. Comparison of label correction algorithms on dataset Citeseer

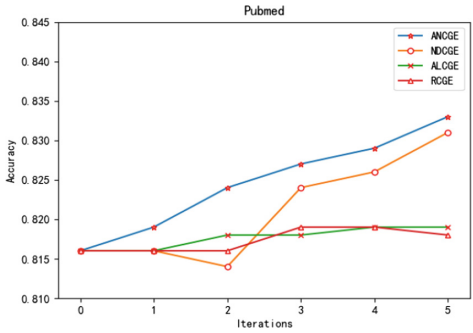

Fig. 3. Comparison of label correction algorithms on dataset Pubmed

Through three experimental results, we can draw the following conclusions: on the whole, with the increase of iteration times, the performance of classifier on each data set has been improved almost. For the three data sets, the performance is respectively improved by 1.6%, 1.7% and 1.7%. But in the process of iteration, the algorithm will fluctuate slightly. Sometimes performance degrades, but eventually performance improves. Specifically, RCGE and ALCGE are difficult to improve the performance of node classification. ALCGE is slightly better than RCGE. The reason for the poor performance of ALCGE is that it only considers the nodes which have great influence on the graph structure, and does not consider more noise selection. Compared with RCGE and ALCGE, NDCGE can effectively improve the performance of node classification. However, in data sets Cora and Pubmed, NDCGE is not as effective as ALCGE at the beginning of iteration. But after several iterations, NDCGE speeds up and surpasses ALCGE. However, the best performance is ANCGE, which is superior to all other algorithms in node classification performance on all data sets.

So it can be seen that active learning can't significantly improve the performance of classifiers, but it can bring positive impact and make NDCGE better. Therefore, in noisy data, priority of correcting is given to the node where the noise possibility is high. ANCGE, which combines active learning and noise detection, can effectively improve the performance of classifier. The above experimental results prove the effectiveness of ANCGE algorithm.

4.5 Stability of ANCGE Model

In this section, we compare the performance of ANCGE model on three data sets with different noise rates. For each data set, we respectively take 10%, 20% and 30% noise rates. In this experiment, 500 labeled nodes are set up. The label budget is 50 nodes, which are iterated five times. The experimental results are shown in Figs. 4, 5 and 6.

Three experimental results show that ANCGE can greatly improve the performance of node classification under different noise rates. For different data sets, the performance improvement range is different between different noise rates. Table 2 is the performance improvement rate of each data set after five iterations. As can be seen from the table, in general, the greater the noise rate, the greater the performance improvement. For dataset Pubmed, the performance improvement at 20% noise rate is better than the other two noise rates. Therefore, the above experimental results verify the stability of ANCGE model under different noise rates.

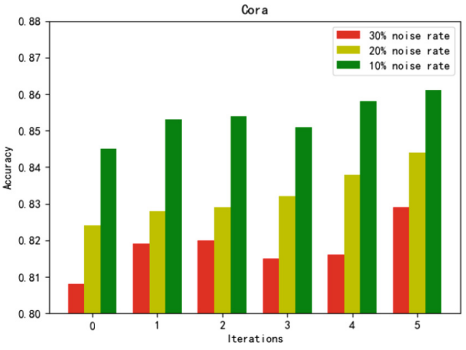

Fig. 4. Accuracy of ANCGE under different noise rates on dataset Cora

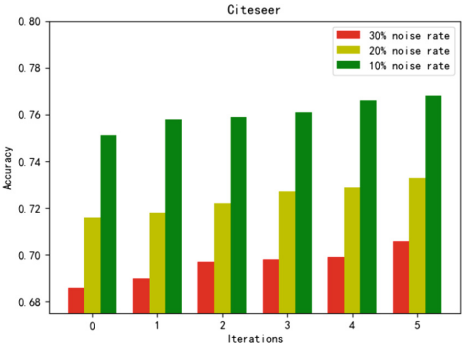

Fig. 5. Accuracy of ANCGE under different noise rates on dataset Citeseer

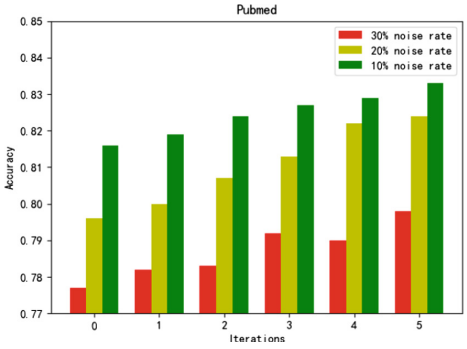

Fig. 6. Accuracy of ANCGE under different noise rates on dataset Pubmed

Table 2. Performance improvement rate of ANCGE

Noise rate	10%	20%	30%
Cora	1.6%	2.0	2.1%
Citeser	1.7%	1.7%	2.0%
Pubmed	1.7%	2.8%	2.1%

5 Conclusion

In this paper, we propose an active correction graph embedding method (ANCGE) based on active learning for image noise data. Given the label budget, it can improve the performance of the classifier by actively learning to find those noisy nodes with large amount of information and great influence on the graph. The experimental results on three open datasets demonstrate the effectiveness of our method and its stability under different noise rates.

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. Bloch, F., Jackson, M.O., Tebaldi, P.: Centrality measures in networks. Available at SSRN 2749124 (2017)
2. Cai, H., Zheng, V.W., Chang, K.C.: A comprehensive survey of graph embedding: problems, techniques, and applications. IEEE Trans. Knowl. Data Eng. **30**(9), 1616–1637 (2018). https://doi.org/10.1109/TKDE.2018.2807452
3. Cai, H., Zheng, V.W., Chang, K.C.: Active learning for graph embedding. CoRR abs/1705.05085 (2017). http://arxiv.org/abs/1705.05085

4. Dawid, A.P., Skene, A.M.: Maximum likelihood estimation of observer error-rates using the EM algorithm. J. Roy. Stat. Soc.: Ser. C (Appl. Stat.) **28**(1), 20–28 (1979)
5. Goyal, P., Ferrara, E.: Graph embedding techniques, applications, and performance: a survey. Knowl.-Based Syst. **151**, 78–94 (2018). https://doi.org/10.1016/j.knosys.2018.03.022
6. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 13–17 August 2016, San Francisco, CA, USA, pp. 855–864 (2016). https://doi.org/10.1145/2939672.2939754
7. Huang, S., Jin, R., Zhou, Z.: Active learning by querying informative and representative examples. IEEE Trans. Pattern Anal. Mach. Intell. **36**(10), 1936–1949 (2014). https://doi.org/10.1109/TPAMI.2014.2307881
8. Jeatrakul, P., Wong, K.W., Fung, C.C.: Data cleaning for classification using misclassification analysis. JACIII **14**(3), 297–302 (2010). https://doi.org/10.20965/jaciii.2010.p0297
9. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, ICLR 2017, 24–26 April 2017, Toulon, France, Conference Track Proceedings (2017). https://openreview.net/forum?id=SJU4ayYgl
10. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: 1st International Conference on Learning Representations, ICLR 2013, 2–4 May 2013, Scottsdale, Arizona, USA, Workshop Track Proceedings (2013). http://arxiv.org/abs/1301.3781
11. Nallapati, R., Surdeanu, M., Manning, C.: Corractive learning: learning from noisy data through human interaction. In: IJCAI Workshop on Intelligence and Interaction. Citeseer (2009)
12. Newman, M.: Networks: An Introduction. Oxford University Press, Oxford (2010)
13. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014, 24–27 August 2014, New York, NY, USA, pp. 701–710 (2014). https://doi.org/10.1145/2623330.2623732
14. Rodriguez, M.A.: Grammar-based random walkers in semantic networks. Knowl.-Based Syst. **21**(7), 727–739 (2008). https://doi.org/10.1016/j.knosys.2008.03.030
15. Settles, B.: Active learning literature survey. Technical report, University of Wisconsin-Madison, Department of Computer Sciences (2009)
16. Snow, R., O'Connor, B., Jurafsky, D., Ng, A.Y.: Cheap and fast–but is it good?: evaluating non-expert annotations for natural language tasks. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 254–263. Association for Computational Linguistics (2008)
17. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: LINE: large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web, WWW 2015, 18–22 May 2015, Florence, Italy, pp. 1067–1077 (2015). https://doi.org/10.1145/2736277.2741093
18. Thongkam, J., Xu, G., Zhang, Y., Huang, F.: Support vector machine for outlier detection in breast cancer survivability prediction. In: Ishikawa, Y., et al. (eds.) APWeb 2008. LNCS, vol. 4977, pp. 99–109. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89376-9_10
19. Tu, C., Yang, C., Liu, Z., Sun, M.: Network representation learning: an overview. SCIENTIA SINICA Informationis **47**(8), 980–996 (2017)

20. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 13–17 August 2016, San Francisco, CA, USA, pp. 1225–1234 (2016). https://doi.org/10.1145/2939672.2939753
21. Yang, Z., Cohen, W.W., Salakhutdinov, R.: Revisiting semi-supervised learning with graph embeddings. In: Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, 19–24 June 2016, New York City, NY, USA, pp. 40–48 (2016). http://proceedings.mlr.press/v48/yanga16.html