Improved DeepWalk Algorithm Based on Preference Random Walk

Zhonglin Ye[1, 2], Haixing Zhao[1, 2*], Ke Zhang[1, 2],
Yu Zhu[1, 2], Yuzhi Xiao[1, 2], Zhaoyang Wang[1, 2]

[1] School of Computer, Qinghai Normal University, Xining 810800, China
[2] Key Laboratory of Tibetan Information Processing, Ministry of Education,
Xining 810008, China
h.x.zhao@163.com

Abstract Network representation learning based on neural network originates from language modeling based on neural network. These two types of tasks are then studied and applied along different paths. DeepWalk is the most classical network representation learning algorithm, which samples the next hop nodes of the walker with an equal probability method through the random walk strategy. Node2vec improves the random walk procedures, thus improving the performance of node2vec algorithm on various tasks. Therefore, we propose an improved DeepWalk algorithm based on preference random walk (PDW), which modifies the single undirected edge into two one-way directed edges in the network, and then gives each one-way directed edge a walk probability based on local random walk algorithm. In the procedures of acquiring walk sequences, the walk probability of the paths that have been walked will be attenuated according to the attenuation coefficient. For the last hop node of the current node in the walk sequences, an inhibition coefficient is set to prevent random walker from returning to the last node with a greater probability. In addition, we introduce the Alias sampling method in order to obtain the next hop node from the neighboring nodes of current node with a non-equal probability sampling. The experimental results show that the proposed PDW algorithm possesses a stable performance of network representation learning, the network node classification performance is better than that of the baseline algorithms used in this paper.

Keywords: network representation, network embedding, network representation learning, network data mining

1 Introduction

The researches on network structures have been conducting from the perspective of statistics methods [1]. However, there are fewer researches on network data

mining using machine learning algorithms [2]. The main reason is that the input of machine learning algorithm needs a large number of features, but the features in various networks is scarce in reality. DeepWalk [3], a network representation learning algorithm, uses the neural network to solve the related tasks of machine learning. Because DeepWalk algorithm is mainly used to learn the relationships between nodes in the network, and DeepWalk compresses this kind of relationships into the form of network representation vectors. DeepWalk algorithm inputs the relationships between nodes into neural network, and outputs the network representation vectors containing the features of node relationships, whose dimension can be regarded as the number of network structure attributes. For example, when the length of the network representation vector is 100, each dimension in the representation vectors represents a certain type of relationship factor between the current node and its neighboring nodes.

The network representation vector generated by DeepWalk algorithm is a low-dimensional, compressed, and distributed vector that contains the local structural features of the networks. Of course, the vectors generated by other network representation learning algorithms based on global information contain the global structural features of the networks [4-6]. The procedure of generating network presentation vectors can be considered as the processing of encoding network structural features, and the procedure can also be considered as the pre-processing of network structural features. Moreover, the pre-processed representation vectors can be inputted into the machine learning model to perform various tasks, such as, network node classification [7], link prediction [8], network visualization [9], recommendation system [10-11] and so on.

DeepWalk algorithm acquires random walk sequences on the network through random walk strategy. The random walk procedure completely adopts the random strategy, namely, DeepWalk randomly selects a node from the neighboring nodes of the current center node as the next hop node of the random walker. Node2vec algorithm [12] improves the random walk procedure of DeepWalk algorithm, which is a preference random walk in fact. Node2vec gives all neighboring nodes of the current center node a fixed walk probability, which consists of the probability of walking to last node, the probability of walking to next node that has no edge with the current center node (set the probability to 1) and the probability of walking to next node that has one edge with the current center node. Under the second kind of walk probability and the third kind of walk probability, there may exist several possible next hop nodes at the same time. Therefore, node2vec adopts the strategy of the equal probability random walk for these nodes with the equal walk probability. Meanwhile, node2vec adopts the strategy of non-equal probability random walk for these nodes with the non-equal walk probability, which is also called as the preference random walk. Node2vec controls the random walker to walk in the direction of the breadth or depth random walk by setting the size of the first type of probability and the third type of probability.

PDW algorithm proposed in this paper is also a kind of the improved DeepWalk algorithm based on preference random walk. First, the PDW algorithm modifies the undirected network into a directed network, namely, a single undirected edge is

converted into two single directed edges. Secondly, the PDW algorithm sets the random walk probability into two categories, such as, the probability of returning the last node in the random walk sequence and the probability of walking to a non-previous node. Among them, the probability of returning the last node is the temporary walk probability, and its purpose is to prevent random walker from walking to the last node. Here, we adopt an inhibition coefficient to adjust this temporary walk probability. When the walker moves to the next node, the temporary walk probability is reset to the previous walk probability. Moreover, PDW reduces its walk probability through the attenuation coefficient for the edges (paths) between nodes that have already walked. Therefore, PDW algorithm controls the random walk procedure by the inhibition coefficient and the attenuation coefficient. Since there are two one-way directed edges between nodes in the network, thus, there exist two walk probabilities between the same pair of nodes. Thirdly, PDW algorithm needs to reset the attenuated walk probability in the network to the original walk probability between nodes after finishing a random procedure of one node. Finally, the Alias method [13] is introduced in PDW to achieve the node sampling of non-uniform probability. The experimental results of PDW algorithm show that its performance is better than that of DeepWalk algorithm and node2vec algorithm in the tasks of node classification on three kinds of citation network datasets.

2 Our Works

2.1 Preference Random Walk

DeepWalk algorithm obtains the node walk sequences through random walk strategy, which can be inputted into the CBOW or Skip-Gram model provided by Word2Vec [14-16] algorithm for training, so as to obtain the network representation vectors of the networks. Regarding the walk sequences, DeepWalk selects a neighboring node around the current node as the next hop node with the equal probability. In order to explain the random walk procedure in detail, we give a simple undirected graph example as shown in Fig. 1.

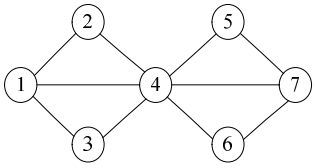

Fig. 1. Network example

As shown in Fig. 1, this graph has 7 nodes and 8 edges. Now, if the random walker is currently at the position of node 1, we set the random walk length as 5 and the number of random walks as 3. Consequently, there are three random walk sequences for node 1, such as, {1, 3, 4, 2, 1}, {1, 4, 3, 2, 1} and {1, 3, 4, 6, 7}. In a random walk,

the random walker can return to the last node of the random walk sequence. Moreover, the probability of walking to its neighboring node is the same.

The PDW algorithm proposed in this paper adopts the preference random walk. The preference random walk is defined in this paper as a random walk that tends to connect the node that has a stronger correlation with the current node. First, we show a simple procedure of preference random walk, which is mainly based on the undirected graph in Fig. 1. PDW algorithm transforms the undirected graph into weighted directed graph, and a single undirected edge is transformed into two one-way directed edges. The weight is the correlation degree between nodes, and Local Random Walk (LRW) [17-18] is used in this paper to measure the correlation degree between two nodes. The specific results are shown in Fig. 2 and Fig.3.

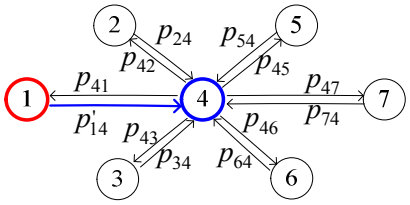

Fig. 2. Random walk on a weighted graph (from node 1 to node 4)

As shown in Fig. 2, the PDW algorithm proposed in this paper needs to attenuate the walk probability of the edge that has already walked when a random walker walk from node 1 to node 4, which is very consistent with the example in our daily life. For example, the probability of going somewhere for two times should be less than the probability of going there for one time. However, if the destination is a popular place, the probability of going there for two times should be greater than the probability of going somewhere else. Based on this assumption, it is very important to assign a reasonable weight to the undirected graph, and the proportion of attenuation is also very important. In Fig. 2, node 4 is regarded to be the node that has been walked when the walker finishes a walk procedure from node 1 to node 4, so the walk probability from node 1 to node 4 should be attenuated. Therefore, we give the new walk probability for this edge (path), i.e. $p'_{14} = p_{14} - p_{14} \cdot q$. For other nodes that have already walked, we give the attenuation equation of walk probability as follows:

$$p'_{uv} = p_{uv} - p_{uv} \cdot q. \tag{1}$$

In the equation (1), p_{uv} is the original random walk probability, p'_{uv} is the updated probability, and q is the attenuation coefficient of walk probability. The attenuation of the walk probability is initialized to the original probability value before random walk of each node, for example, we set the number of random walks as 10, the length of random walk as 40, the original walk probability needs to be reused to carry out the random walk of next node when the walk sampling of 400 nodes is obtained. In addition, the attenuation of the walk probability is unidirectional. For example, the value of the walk probability p_{14} will be changed, but the value of the

walk probability p_{41} will not be changed for random walk from node 1 to node 4 . When the random walker is at the position of node 4 during a random walk, the probability of walking to node 1 is the original and unattenuated p_{41}, which can be regarded as such understanding that the probability of walking the same path should be attenuation when we return from place *A* to place *B*, however, the probability of walking form place *B* to place *A* should not be attenuated when we return from place *C* to place *B*, but the probability of walking from place *B* to place *A* should be inhibited when we just walk from place *A* to place *B*. This is the main reason why we have transformed the undirected network into the two-way directed network.

Fig. 2 shows the example of the one-step random walk. After that, the random walker needs to walk from node 4. This procedure is somewhat different from the walk from node 1 to node 4, because the walk from node 4 needs to consider such case that the random walker returns to node 1. For example, we should exclude the places we have been to when we plan the next destination, but the places we have been to should still be selected again with a small probability. Therefore, we define that the random walker should return to the last node in walk sequence with a small probability. In order to explain this principle in more detail, we give the random walk procedure in Fig. 3.

In Fig. 3, there exist 6 next hop nodes when the random walker is at node 4, among which one random walk path can return to the last node 1 in the random walk sequence. However, we should avoid returning the last node of the walk sequence as mentioned above. Therefore, we define the walk probability attenuation coefficient as $1/p$ between the current node and the last node of the walk sequence, i.e. $p'_{41} = p_{41} - p_{41} \cdot (1/p)$. This attenuation is temporary attenuation, which only recomputes the walk probability of returning last node when the walker will select the next hop node, and it does not affect the original walk probability between nodes. When the walker walks from node 4 to node 7, we attenuate the walking probability from node 4 to node 7, i.e. $p'_{47} = p_{47} - p_{47} \cdot q$. In addition, the original walk probability from node 4 and node 1 is recovered, i.e. $p'_{41} = p_{41}$. Therefore, when the walker selects the next hop node, the temporary walk probability of returning last node is defined as:

$$p'_{xy} = p_{xy} - p_{xy} \cdot (1/p). \tag{2}$$

In the equation (2), p is the inhibition coefficient of returning the last node. p_{xy} is the original walk probability, p'_{xy} is the walk probability after attenuating, and p'_{xy} is only a probability that the walker returns to the last node of the sequence when the walker selects the next hop node. When the random walker walks to the next hop node, the inhibition is canceled, i.e. $p'_{xy} = p_{xy}$. And then use the equation (1) to carry out probability attenuation on the path that has been walked.

It should be noted that all attenuations of the above procedures only work on the walk procedure of a node. For example, the walk algorithm needs to walk for 10 times when the number of random walks is 10. After completing these 10 walks for the same node, the walk algorithm needs to perform another 10 walks for the next node. The walk probability between two nodes is reset to the original probability value

when the different nodes are selected to walk, which is calculated by LRW algorithm. The LRW algorithm is different from the global random walk in that it only considers the random walk procedure within a finite step, so it is very suitable to walk on large-scale networks. When a random walker starts from a node v_x in LRW, $f_{xy}(t)$ is defined as the probability that a random walker happens to meet with node v_y at a certain $t+1$. Then, the system evolution equation of the walk procedure is as follows:

$$f_x(t+1) = P^{\mathrm{T}} f_x(t), \quad t \geq 0, \tag{3}$$

where the markov transition probability matrix [18] of the network is defined as P, the element in P is computed as $P_{xy} = a_{xy} / k_x$. Assume that there is an edge between nodes v_x and v_y, then $a_{xy} = 1$, $a_{xy} = 0$ otherwise. k_x represents the degree of the node v_x. $f_x(0)$ is a vector with the size of $N \times 1$ where the value of x-th element is 1, and all other elements are 0. The initial network resource distribution is defined as $q_x = k_x / M$, and M is the total number of network edges. Then the similarity between nodes v_x and v_y in t steps is as follows:

$$s_{xy}(t) = q_x \cdot f_{xy}(t) + q_y \cdot f_{yx}(t). \tag{4}$$

In this section, the random walk probabilities between the current center node and its neighboring nodes are measured by $s_{xy}(15)$. In addition, the link prediction task between nodes is used to verify that the similarity weight between nodes for $t = 15$ is better than other values of t. This section mainly describes how to generate the initial probability of preference random walk between nodes, how to perform the probability attenuation in the random walk procedure, and how to temporarily attenuate the probability value of returning the last node. But we will be discussed in detail how to select one of the nodes to random walk based on the walk probability between the current node and its neighboring nodes in the following parts.

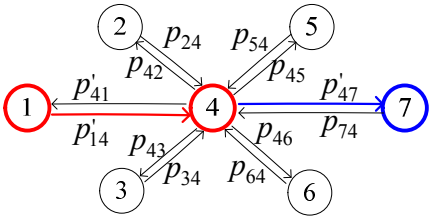

Fig. 3. Random walk on a weighted graph (from node 4 to node 7)

2.2 Non-equal Probability Node Selection

The node selection of non-equally probability refers to select one node from all neighboring nodes of the current node as the next hop node of the random walker

according to the given walk probability. First, there are two easy ways to realize it as follows.

The first kind of selection approach is based on random number. For example, the probabilities of four events are 0.1, 0.2, 0.3 and 0.4, respectively, and then an array of size of 100 is built and shuffled completely, where 10 elements in the array are set as the happening of event 1, 20 elements are set as the happening event 2, 30 elements are set to the happening events 3, and 40 elements is set as the happening events 4. After constructing the array, a random integer from 1 to 100 is generated randomly, and the element corresponding to the random integer in the array is the related event. The time complexity of this method is $O(1)$, but the accuracy is poor.

The second kind of selection approach is based on roulette method. For example, if the probability distributions of the four events are 0.1, 0.2, 0.3 and 0.4, consequently, the cumulative probabilities of the four events are 0.1, 0.3, 0.6 and 1, respectively. Then a random number between 0 and 1 is generated, and the event corresponding to the random number falling in the cumulative probability interval is the selected event. For example, the interval [0, 0.1] corresponds to the event 1, the interval [0.1, 0.3) corresponds to event 2, the interval [0.3, 0.6) corresponds to event 3, and the interval [0.6, 1] corresponds to event 4. If this selection algorithm adopts the binary search approach, the time complexity is $O(\log n)$. Although this selection algorithm is better than the first one in in practical applications, it requires to calculate the cumulative probability.

In this paper, the third kind of selection approach is introduced, which has a time complexity of $O(1)$, also has an excellent sampling performance. This algorithm is called as Alias method. In this method, the event occurrence probabilities 0.1, 0.2, 0.3 and 0.4 are normalized according to their mean value, the mean value is 1/4, and the probabilities of the final normalization are 2/5, 4/5, 6/5 and 8/5. The probability is illustrated as shown in Fig. 4.

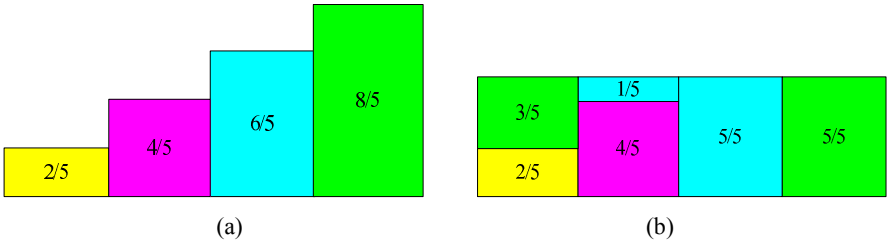

Fig. 4. An example of Alias sampling

In Fig. 4, the Alias method first converts the normalized probabilities to the form as Fig. 4 (a). Then, Fig. 4 (a) is converted to a 1×4 rectangle as shown in Fig. 4 (b), where the area of the rectangle is 4. The Alias method contains at most two events per column as shown in Fig. 4 (b). This rectangle is also known as the Alias table. There are mainly two arrays in this table, one is the probability array $prabs$, where each element is the area percentage (probability value) of the event i corresponding to the column i, namely, $prabs$ = {0.1, 0.2, 0.3, 0.4}. Another array saves the tags

that do not belong to the event i. This array is defined as $alias$ = {4, 3, null, null}. Alias method then generates two random integers, the first random integer determines the column to be used, and the second random number is between 0 and 1. If the second random number is less than $prabs[i]$, the subscript of array $prabs$ is sampled and returned. If the second random number is greater than $prabs[i]$, $alias[i]$ is sampled and returned. The result of the above Alias method is an integer rather than a probability value.

After the PDW algorithm samples the next hop node of the walker based on the above procedures, the context node pairs can be subsequently constructed, and then they are inputted into the neural network model provided by DeepWalk algorithm for node relationship modeling. The PDW algorithm proposed in this section only improves random walk procedure of DeepWalk, but PDW algorithm adopts multiple efficient methods to obtain the appropriate next hop nodes, so that the random walk sequences can better reflect the structural features of the networks and other information.

3 Experimental Results and Analysis

3.1 Experimental Settings

We adopt DeepWalk, LINE [19], HARP [20], DeepWalk+NEU [21], GraRep (K = 3) [22] and node2vec as the baseline algorithms. The baseline algorithms selected in this section set the length of network representation vector as 100 in the task of network node classification. In addition, DeepWalk and node2vec algorithms need to set the random walk number and random walk length. Therefore, in the network node classification experiment, the number of random walks is set to 10, and the length of random walk is set to 40. Node2vec sets the breadth-first random walk parameter to 0.5 and the depth-first random walk parameter to 0.25. In this parameter combination, node2vec is more inclined to obtain nodes by the depth-first random walk. We repeat each experiment for 10 times and take the average accuracy as the result of network node classification using LIBLINEAR [23] classifier. In addition, DeepWalk, HARP (DeepWalk), node2vec and PDW use CBOW model to model the relationships between nodes, use negative sampling to optimize its training speed, and set the negative sampling size as 5.

3.2 Results and Analysis

We verify the performance of PDW algorithm and various comparison algorithms on Citeseer, Cora and DBLP datasets [24]. In order to make a more detailed comparison and analysis under different training set proportions, we extract 10%, 20%, ..., 90% nodes of the dataset as the training set, the rest of the dataset is the testing set. In the PDW algorithm, p is the inhibition coefficient of returning the last node of the walk sequence. q is the attenuation coefficient of random walk probability. The specific accuracy results of network node classification are shown in table 1, 2 and 3.

Table 1. Classification performance on Citeseer dataset (%)

Algorithm name	10%	20%	30%	40%	50%	60%	70%	80%	90%	Average
DeepWalk	47.6	50.2	51.9	52.3	53.7	53.2	53.8	53.9	54.6	52.3
LINE	41.2	44.6	47.9	49.2	52.2	53.5	53.9	53.3	53.9	49.5
HARP (DeepWalk)	48.9	50.3	50.8	50.7	51.3	51.3	50.3	51.8	53.0	50.9
DeepWalk + NEU	48.5	51.2	52.5	53.9	53.5	54.7	54.6	54.4	55.9	53.2
GraRep (K = 3)	45.1	51.0	53.4	54.2	54.9	55.8	55.5	55.2	54.2	53.2
node2vec	50.8	52.6	54.3	54.5	55.7	56.2	55.6	56.2	56.6	54.7
PDW (p = 5, q = 0.05)	53.2	55.2	55.7	56.3	57.3	57.9	58.0	58.0	57.7	56.6
PDW (p = 10, q = 0.05)	52.9	55.0	55.7	56.7	56.3	56.9	57.4	57.4	57.7	56.2
PDW (p = 20, q = 0.1)	53.7	54.8	55.4	55.9	56.1	57.0	56.7	57.9	57.2	56.1

Table 2. Classification performance on Cora dataset (%)

Algorithm name	10%	20%	30%	40%	50%	60%	70%	80%	90%	Average
DeepWalk	67.6	72.1	74.5	75.1	76.7	76.7	77.4	78.1	77.7	75.1
LINE	64.3	68.4	70.1	71.3	73.3	75.8	75.6	77.7	79.5	68.8
HARP (DeepWalk)	65.6	68.5	70.8	71.0	70.9	70.8	71.2	72.8	72.9	70.5
DeepWalk + NEU	69.3	74.7	76.1	77.3	77.8	78.6	78.8	79.4	79.1	76.8
GraRep (K = 3)	72.6	77.3	78.3	79.4	79.4	80.3	80.3	80.7	79.9	78.7
node2vec	69.3	73.2	74.1	75.6	76.1	76.6	76.5	77.5	77.4	75.2
PDW (p = 5, q = 0.05)	75.7	78.1	79.4	80.2	80.5	81.0	80.6	80.9	81.0	79.7
PDW (p = 10, q = 0.05)	76.0	78.8	79.5	80.0	80.3	80.4	81.8	81.6	81.6	80.0
PDW (p = 20, q = 0.1)	76.5	78.6	79.9	79.9	80.2	80.8	81.4	81.7	81.1	80.0

Table 3. Classification performance on DBLP dataset (%)

Algorithm name	10%	20%	30%	40%	50%	60%	70%	80%	90%	Average
DeepWalk	76.7	79.5	80.8	81.2	82.1	81.6	82.6	83.2	82.6	81.2
LINE	73.3	75.2	76.9	77.4	78.1	78.7	78.9	80.1	80.5	77.7
HARP (DeepWalk)	78.7	80.1	80.8	81.1	80.8	81.3	81.1	81.0	81.8	80.7
DeepWalk + NEU	80.9	81.6	82.1	83.8	83.9	83.8	83.9	84.5	84.0	83.2
GraRep (K = 3)	81.6	83.1	84.3	84.1	84.0	84.4	85.2	85.5	85.1	84.2
node2vec	83.2	83.1	83.3	83.6	84.8	84.9	84.3	84.8	84.8	84.1
PDW (p = 5, q = 0.05)	82.9	83.3	83.8	83.9	84.3	84.1	85.0	84.4	84.6	84.0
PDW (p = 10, q = 0.05)	82.6	83.3	83.8	84.2	84.2	84.5	84.4	85.3	85.3	84.2
PDW (p = 20, q = 0.1)	82.7	83.5	84.0	84.8	84.6	84.2	84.1	85.9	85.1	84.3

Based on the results in Tables 2, 3 and 4, we have the following observations:

(a) PDW algorithm improves the random walk procedure of DeepWalk algorithm. Therefore, the accuracy improvement of PDW algorithm is 7.13% at least and 8.1% at most on the Citeseer dataset. In Cora dataset, PDW algorithm has an accuracy improvement of 6.13% at least and 6.52% at most. In the DBLP dataset, PDW

algorithm has an accuracy improvement of 3.54% at least and 3.89% at most. These results show that the improvement of PDW algorithm based on DeepWalk is effective.

(b) Node2vec algorithm also improves the random walk procedure of DeepWalk algorithm. Therefore, the improved idea is the same with PDW algorithm. Experimental results show that PDW algorithm has an accuracy improvement of 2.47% at least and 3.40% at most compared with node2vec algorithm on the Citeseer dataset. In Cora dataset, PDW algorithm is improved by 6.1% at least and 6.45% at most. In DBLP data set, PDW algorithm has an accuracy improvement of 0.24% at most. These results show that although both PDW algorithm and node2vec algorithm improves on the random walk procedures of DeepWalk algorithm.

(c) Compared with DeepWalk algorithm, the improvement rate of classification tasks of PDW algorithm gradually decreases with the increasing growth of the network density. Compared with node2vec algorithm, PDW algorithm has the highest performance improvement on Cora dataset, and the classification performance of PDW algorithm and node2vec algorithm is almost the same on dense DBLP dataset. Node2vec algorithm achieves poor node classification performance on Cora dataset. Consequently, the PDW algorithm proposed in this paper shows a stable network node classification performance on Citeseer, Cora and DBLP datasets. Because DBLP is a dense dataset, so various algorithms almost achieve same classification performace on DBLP dataset.

(d) PDW algorithm sets the inhibition coefficient p of returning last node as 5, 10, 20, the attenuation coefficient q as 0.05, 0.1. Experimental results show that the different parameter combinations have little influence on the classification performance of PDW algorithm.

3.3 Parameter Sensitive Analysis

Three parameters are mainly set in the PDW algorithm, namely, the PDW algorithm sets the inhibition coefficient p of returning the last node, the attenuation coefficient q for the walked paths, and the network representation vector length k. In addition, we also discuss the effect of random walk length. The specific results are shown in Fig. 5.

4 Conclusion

The main works of the proposed PDW algorithm are to improve the random walk procedures of DeepWalk algorithm. Specifically, the improvement is to convert the original undirected graph into a weighted two-way directed graph, and the weight of the network is initialized by the local random walk algorithm. Random walker performs a random walk in a weighted bidirectional directed network. During the random walk, PDW algorithm attenuates the walk probabilities of the edges that have been walked, and inhibits the probabilities of returning the last nodes in the random walk sequences. For the feasibility of the above improvements, we conduct the network node classification evaluation on Citeseer, Cora, DBLP datasets. The experimental results show that the PDW algorithm can efficiently obtain the next hop

nodes of random walker, the generated node sequences are inputted to the shallow neural network to train and model the relationships between network nodes. Consequently, the network node classification performance of PDW algorithm is superior to that of DeepWalk algorithm and other baseline algorithms. In addition, the classification performance of PDW and node2vec algorithms is compared in detail, because these two algorithms are improved based on the random walk strategy. On DBLP dataset, the classification performance of PDW algorithm is almost the same as that of node2vec algorithm.

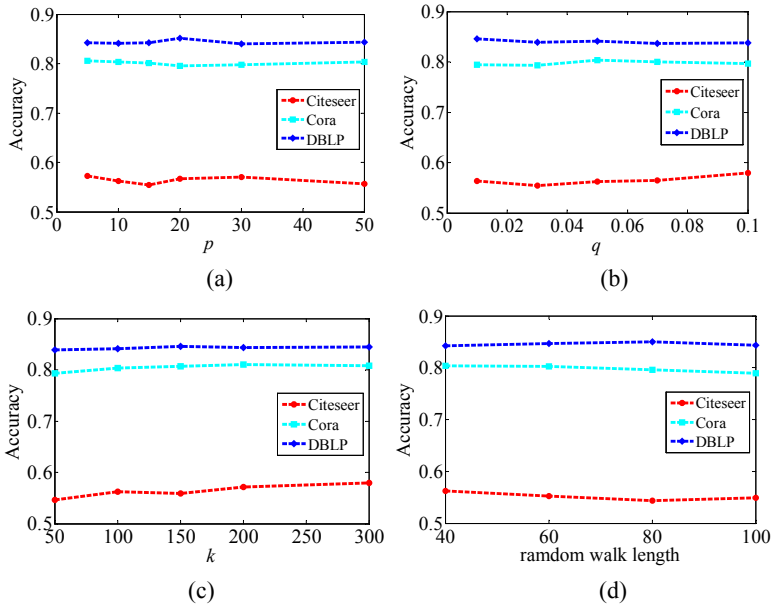

Fig. 5. Influence analysis of all parameters

Acknowledgement

This project is supported by NSFC (No. 11661069, 61663041 and 61763041).

References

1. Vedanayaki, M.: A Study of Data Mining and Social Network Analysis. Journal of Testing & Evaluation 40(7), 663-681 (2014).
2. Bhat, S.Y., Abulaish M.: Analysis and mining of online social networks: emerging trends and challenges. Wiley Interdisciplinary Reviews Data Mining & Knowledge Discovery 3(6), 408-444(2013).
3. Perozzi, B., Al-Rfou, R., Skiena, S., et al.: Deepwalk: online learning of social representations. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701-710. ACM, New York (2014).
4. Tu,C.C., Zhang, W.C., Liu, Z.Y., et al.: Max-margin deepwalk: discriminative learning of network representation. In: International Joint Conference on Artificial Intelligence, pp. 3889-3895. AAAI, Palo Alto (2015)
5. Yang, C., Liu, Z.Y., Zhao, D., et al.: Network representation learning with rich text

information. In: International Conference on Artificial Intelligence, pp. 2111-2117. AAAI, Palo Alto (2015).
6. Wang, D., Cui, P., Zhu, W., et al.: Structural deep network embedding. In: The 22nd ACM SIGKDD international conference on Knowledge Discovery and Data Mining, pp. 1225-1234. ACM, New York (2016).
7. Zhang, D., Jie, Y., Zhu, X., et al.: Network representation learning: a survey. IEEE Transactions on Big Data PP(99), 1-1 (2017).
8. Li, W.J.: Predictive network representation learning for link prediction. http://101.96.10.64/www4.comp.polyu.edu.hk/~csztwang/paper/pnrl.pdf.
9. Doncheva, N.T., Morris, J.H., Gorodkin, J., et al.: Cytoscape StringApp: Network Analysis and Visualization of Proteomics Data. Journal of Proteome Research 18(2), 623-632 (2019).
10. Zhao, W.X., Huang, J., Wen, J.R., et al.: Learning distributed representations for recommender systems with a network embedding approach. In: Asia Information Retrieval Symposium, pp. 224-236. Springer, Heidelberg (2016).
11. Yu, X., Ren, X., Sun, Y.Z., et al.: Personalized entity recommendation: a heterogeneous information network approach. In: ACM International Conference on Web Search and Data Mining, pp. 283-292. ACM, New York (2014).
12. 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, pp. 855-864.: ACM, New York (2016).
13. Ohata, F., Kondou, K., Inoue, K., et al.: Alias analysis method for object-oriented programs using alias flow graphs. Systems and Computers in Japan 35(4), 49-59 (2004).
14. Lai, S., Liu, K., He, S., et al.: How to Generate a Good Word Embedding. IEEE Intelligent Systems 31(6), 5-14 (2016).
15. Mikolov, T., Sutskever, I., Chen, K., et al.: Distributed representations of words and phrases and their compositionality. In: Twenty-seventh Conference on Neural Information Processing System, pp. 3111-3119. MIT, Massachusetts (2013).
16. Mikolov, T., Corrado, G., Chen, K., et al.: Efficient estimation of word representations in vector space. https://arxiv.org/abs/1301.3781.
17. Kingman, J.F.C.: Markov transition probabilities. Zeitschrift Für Wahrscheinlichkeitstheorie Und Verwandte Gebiete 7(4), 248-270 (1967).
18. Liu, W., Lü, L.Y.: Link prediction based on local random walk. Europhysics Letter 89(5): 58007 (2010).
19. Tang, J., Qu, M., Wang, M., et al.: LINE: large-scale information network embedding. In: International Conference on World Wide Web, pp. 1067-1077. Springer, Berlin (2014).
20. Chen, H., Perozzi, B., Hu, Y., et al.: HARP: hierarchical representation learning for networks. https://arxiv.org/abs/1706.07845.
21. Yang, C., Sun, M., Liu, Z., et al.: Fast network embedding enhancement via high order proximity approximation. In: International Joint Conference on Artificial Intelligence, pp. 3894-3900. Morgan Kaufmann, San Francisco (2017).
22. Cao, S., Lu, W., Xu, Q., et al.: GraRep: learning graph representations with global structural information. In: Conference on information and knowledge management, pp. 891-900. ACM, New York (2015).
23. Fan, R.E., Chang, K.W., Hsieh, C.J., et al.: LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research 9(9): 1871-1874 (2008).
24. Ye, Z.L., Zhao, H.X., Zhang, K., et al.: Text-associated max-margin DeepWalk. In: CCF Big Data 2018, pp. 301-321. Springer, Heidelberg (2018).