




Graph Embedding Based on Characteristic of Rooted Subgraph Structure

Yan Liu^{1,2} , Xiaokun Zhang¹, Lian Liu³, and Gaojian Li¹

¹ PLA Strategic Support Force Information Engineering University, Zhengzhou 450001, China
ms.liuyan@foxmail.com

² State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China

³ Investigation Technology Center PLCCM, Beijing 100000, China

Abstract. Given the problem that currently distributed graph embedding models have not yet been effectively modeled of substructure similarity, biased-graph2vec, a graph embedding model based on structural characteristics of rooted subgraphs is proposed in this paper. This model, based on the distributed representation model of the graph, has modified its original random walk process and converted it to a random walk with weight bias based on structural similarity. The appropriate context is generated for all substructures. Based on preserving the tag features of the nodes and edges in the substructure, the representation of the substructure in the feature space depends more on the structural similarity itself. Biased-graph2vec calculates the graph representations with unsupervised algorithm and could build the model for both graphs and substructures via universal models, leaving complex feature engineering behind and has functional mobility. Meanwhile, this method models similar information among substructures, solving the problem that typical random walk strategies could not capture similarities of substructures with long distance. The experiments of graph classification are carried out on six open benchmark datasets. The comparison among our method, the graph kernel method, and the baseline method without considering the structural similarity of long-distance ions is made. Experiments show that the method this paper proposed has varying degrees inordinately improved the accuracy of classification tasks.

Keywords: Graph data · Network embedding · Graph embedding · Structural similarity · Graph classification

1 Introduction

Graph data is a data form widely exist in the field of biochemistry, social network & network security, in which tasks like the prediction of biochemical characteristics [1], community detection [2], malicious code detection [3], *etc.* often share a tight link with graph classifications and clusters. There is a need to put the original graph into representation as an eigenvector of fixed length to facilitate the application of mature classification and clustering algorithms in machine learning. Graph embedding, therefore, is the method that via studies upon how to maintain enough characteristics of the

graph while fitting it into the characteristic space, making the original graph being presented in vectors as integrated as possible, and eigenvectors after representation could lead to a better outcome in the following tasks of graph-processing.

The work related to the graph representation can be broadly divided into two categories:

(1) **Graph Kernel Method.** The Graph kernel method is a widely used method to measure the similarities between different graph structures. For ordinary kernel methods, the primary thought is to map a low-dimension vector x to a higher dimension that reproduces kernel Hilbert space using a nonlinear mapping function \emptyset . This way, nonlinear tasks that are relatively hard to calculate in lower dimensions can be solved in higher dimension Hilbert feature space through linear algorithms. Early Graph embedding methods mainly include the graph kernel methods [4–6] and dimensionality-reducing methods (multidimensional scaling (MDS) [7], IsoMap [8], locally linear embedding – LLE [9]).

Though graph kernel methods stand their crucial role in multiple graph-related tasks and are now still widely used, the method has its restrictions: 1) The high-dimensional feature representation obtained by the graph kernel method has some information redundancy, which leads to the high cost of calculation and memory. 2) This method needs a predefined kernel function, which relies on feature engineering to get practical features. As a result, making the model insufficient mobility. 3) In the graph kernel method, we generally regard substructures as atomic structures, ignoring the structural similarities among substructures.

(2) **Graph Embedding Method.** Graph embedding could be regarded as a unique form of network embedding in particular circumstances. Graph embedding learning and network embedding learning both aim to learn the low dimension vector presentations in the feature space. However, there is some difference between graph embedding learning and network embedding learning. Network embedding learning faces networks with rich node properties information like social networks. Graph embedding learning, simultaneously, faces network with rich graph-structural data like biochemical structures and code flows, which do not contain much information of node properties but contain rich information like node labels, edge labels, and weights. DeepWalk [10] is the first article to apply the word vector model, word2vec, and random walk to the field of network embedding. Later embedding models like LINE [11], node2vec [12], *etc.* are based on the representation learning model in the framework of the random walk. As word vector models widely used, new research based on distributed word vector models has come out in graph embedding field, e.g., subgraph2vec [3], graph2vec [13], GE-FSG [14], *etc.* These methods share similarities in their general frames. They decompose the graphs into their atom substructures by considering the graph as the document, considering the atom substructures as words in the document. The graph-embedding models can use the word vector model to learn the low dimension embeddings of each graph.

However, the graph embedding model fundamentally differs from the word embedding model. In the word vector model, it is not easy to capture the similarities between words in the very beginning. The similarity can only be obtained via the model's embedding results. If there are two certain random words marked as w_1 , w_2 , the model will not be able to measure their similarity without information of their context properly. In graph

embedding learning, the structural similarity between substructures is easy to measure. For example, given two substructures g_1, g_2 , even if there is no information about their context, the model can still capture their similarity by measuring characteristics such as edges, nodes, degrees, graph-kernels, and more. Existing graph embedding models leave this similarity information behind.

Subgraph (substructure), as a significant characteristic of graphs and networks, obtains a higher level of abstraction information than characteristics like nodes, edges, degrees, *etc.* Many recent kinds of research regard subgraphs as the atom substructures of graphs and learn their representation via distributed learning models. However, those graph embedding models ignore the similarities between subgraphs that exist in the very beginning. This paper proposes **biased-graph2vec**, which is based on structural characteristics of rooted subgraphs. The model learns the vector representation of both graphs and rooted subgraphs in the same vector space. The classified task has been tested in six base data sets, and the result shows the accuracy of graph classifications, compare to the baseline method, which has been varying degrees improved.

2 Problem Definition

Given a graph set Γ , graphs in Γ represented as $\{G_1, G_2, \dots\}$. The goal of graph embedding is to learn d -dimensional representation in the characteristic vector space of each graph G in Γ . In the learning process, it is vital that characteristic vectors reserve corresponding characteristics of labels and edges in substructures and context of the substructures. Moreover, the dimensionality d should be properly set to keep the memory and calculation cost of the representation matrix $\Phi \in R^{|\Gamma| \times d}$ low enough.

Graphs in the graph set Γ are defined as $G = (N, E, \lambda)$, in which N stands for the set of nodes, $E \subseteq (N \times N)$ represents the edge set. For that data used in graph embedding usually has relatively complete labels of edges and nodes, therefore in such a system, we define those graphs with label information as labeled graphs while those without defined as non-labeled graphs. In labeled graphs, there exist functions $\lambda : N \rightarrow \ell$, mapping each node to a corresponding character in the alphabet ℓ , in the same way, we define the edge-mapping function $\eta : E \rightarrow \varepsilon$.

For two given graphs, $G = (N, E, \lambda)$ and $G_{sg} = (N_{sg}, E_{sg}, \lambda_{sg})$, G_{sg} is the subgraph of G if and only if there exists an injective function $\mu : N_{sg} \rightarrow N$, which makes $(\mu(n_1), \mu(n_2)) \in E$ if and only if $(n_1, n_2) \in E_{sg}$.

3 Graph Embedding Model Based on Structural Characteristics of Rooted Graphs

According to descriptions in [15], the context of substructures only shows local characteristics of substructures. Therefore, if two substructures resemble each other but are far from each other in the context, it is almost impossible for existing distributed graph models to produce valid learning results. In a graph-related calculation, there is usually enough label information of nodes and edges, but apart from that, the similarities between substructures also mean they have similar characteristics. For example, in a malicious

code detection task, the function call of code segment A is different from malicious code segment B, but they share a similar process of execution. Therefore code segment A is highly suspectable, which means the representation of A in feature space resembles B more than a random one. In the existing graph distributed representation model, A and B as independent atomic structures possess different labels and contexts, making the final results of representation differ a lot from each other. Currently, existing distributed graph embedding models are incapable of capturing structural similarities between distanced structures like these. As in Sect. 3, by bringing the substructure's similarity into consideration, this paper proposed biased-Graph2vec to solve the problem.

3.1 The Frame of Biased-Graph2vec

The d -dimensional subgraph of the node n in graph g is defined as a subgraph containing every single node that could be reached from node n in d hops and edges between these nodes.

The core of the model is to use the similarities information among substructures to create a hierarchical structure from where random walks are performed. This paper uses graph sets in experiments, to learn a graph's embedding the process is the same.

The biased-graph2vec mainly consists of two parts. The first part is to traverse all nodes to produce substructures related to every node, construct a 2-layer structure to capture similarity information among substructures, then use the biased random walk algorithm to perform a random walk in the two-layer graph structure to obtain contexts containing structural characteristics of the subgraphs. The second part is just like other graph embedding models, to fit all substructures into the word embedding model doc2vec. Treat substructures as a word in the word embedding model, the graph as the word sequence and substructures of the graph as words. Doc2vec model is applied to acquire the representation of the graph and substructures in the feature space of a lower dimension.

According to the specific task, the selection subgraphs is flexible choosing from subgraphs like frequent subgraph, rooted subgraph, ego graph, *etc.* Biased-graph2vec, in this paper, selects rooted subgraphs as its substructures. Advantages of rooted subgraphs over the other options are as followed:

- (1) The computation cost is much less than the frequent subgraph mining algorithm.
- (2) Rooted subgraphs, compared with other characteristics like nodes and edges, possess a higher level of abstraction, possibly containing more information on the graph. Once applied to the word vector model, for it is generated from node traversal, different rooted subgraphs share a similar order of magnitudes. The more valuable information the subgraph contains, the better the graph representation will be.
- (3) The nonlinear substructure could better acquire the graph's characteristics in normal tasks than linear substructures. Weisfeiler-Lehman (WL) kernel [6] algorithm, for example, is the kind of algorithm based on characteristics of rooted graphs that appears to be more effective in both experiments and applications than other linear graph-kernel algorithms like random walk kernel algorithm and shortest path kernel algorithm.

The processing procedure of biased-graph2vec is shown in the Fig. 1. First, generate the set of rooted subgraphs, then calculate the structural similarities among substructures, build a model of the biased random walk from structural similarities, and the model of the document is built based on that model, finally output the low dimension representations vectors of substructures and graphs.

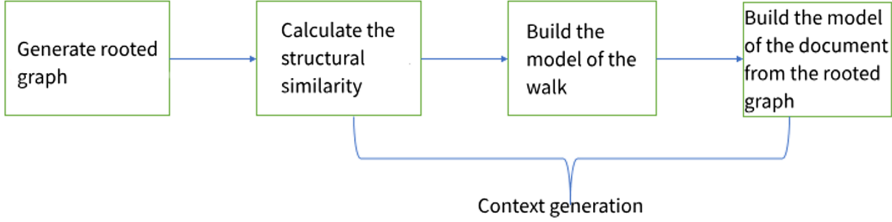


Fig. 1. A processing procedure of biased-graph2vec

3.2 Rooted Subgraphs Generation

Graph set denoted as $D = \{d_1, d_2, \dots, d_N\}$, generate rooted subgraph $sg_v^{(h)}$ for every node $v \in d_i$ of every single graph $d_i \in D$, h denotes the depth of currently rooted subgraph. After h iterations, the result set contains all the root subgraphs of nodes v less than or equal to h -order neighbor nodes. The procedure of generating rooted subgraph refers to the labeling process of the Weisfeiler-Leman (WL) kernel [6] algorithm, the WL relabeling process, which is shown in Algorithm 1. The input of the algorithm is current node v , graph to be extracted G , hyper-parameter h depth of extraction. The process of extracting rooted subgraphs is recursive, in which h controls the of recursive depth the extraction, meaning that in the end, the rooted graph set of node v contains rooted graphs ordered from 0 to h (when h equals 0 the function returns to the current node). The larger h is, the more rooted subgraphs are extracted, and the more information about adjacent structures are contained, the more subsequent computations cost will be. The rooted subgraph generation algorithm is shown in Fig. 2.

3.3 Context Generation

After recursion of all nodes of each graph in graph set D , the set of all rooted subgraphs in the graph set is generated and denoted as G_{sg} . Also, the neighboring rooted graphs of the targeted subgraph are obtained in the subgraph generating process. For example, if h is 3, the subgraph $sg_v^{(1)}$ has the context of $sg_v^{(0)}, sg_v^{(1)}, sg_v^{(2)}$ that represent local information of $sg_v^{(1)}$. According to WL relabeling process, it is able to represent every rooted subgraph in the form of unambiguous unique strings, which make up the vocabulary of rooted subgraphs, $Vocab_{sg}$.

The context obtained above only represents the local feature of the rooted subgraph. To capture the similarity between subgraphs far away and expand the range of the random walk, the context generating process should meet the requirements below:

Algorithm 1: GetWLSubgraph($v; G; h$)

Input: v : current node.
 $G = (V, E, \lambda)$: graph to perform subgraph extraction.
 h : extraction depth of rooted graph.

Output: $sg_v^{(h)}$: rooted subgraph set of the current node with extraction depth h .

```

1: function GetWLSubgraph( $v; G; h$ )
2:    $sg_v^{(h)} = \{ \}$ 
3:   if  $h = 0$  then
4:      $sg_v^{(h)} \leftarrow \lambda(v)$ 
5:   else
6:      $Neighbor_v := \{v' | (v, v') \in E\}$ 
7:      $M_v^{(h)} := \{GetWLSubgraph(v', G, h - 1) | v' \in Neighbor_v\}$ 
8:      $sg_v^{(h)} := sg_v^{(h)} \cup GetWLSubgraph(v, G, h - 1) \oplus sort(M_v^{(h)})$ 
9:   end if
10:  return  $sg_v^{(h)}$ 
11: end function

```

Fig. 2. Process of GetWLSubgraph algorithm

For rooted subgraphs $sg_1, sg_2 \in \mathbb{G}_{sg}$, representations in feature space are correspondingly marked as $e(sg_1), e(sg_2)$. The distance of $e(sg_1), e(sg_2)$ embedding in vector space should reflect not only the local context's similarity but also the structure similarity of sg_1, sg_2 themselves.

Structural Similarity Calculation. Depending on the specific situation, a standard similarity measure can be used to measure structural similarity such as node similarity, edge similarity, graph kernel, *etc.* Considering the calculation cost, for that the degree of the node reveals a structural similarity to some degree, this section calculates the similarities among each node's degree sequence of rooted subgraphs to infer the structural similarity of the rooted subgraphs.

For a given subgraph $sg \in \Gamma_{sg}$, its ordered sequence of nodes' degree is marked as $s(V_{sg})$, and V_{sg} stands for the node-set of subgraph sg . Due to the sequence length could be inequality, this section applies the Dynamic Time Warping (DTW) method to the calculation. In this method, all elements in two sequences will be correspondingly lined up individually, making the sum of aligned sequences' distance reduced to the least. Let sequences A, B represent the sequences of degrees of nodes, this section uses the formula below to calculate the distance between $a \in A, b \in B$:

$$d(a, b) = \frac{\max(a, b)}{\min(a, b)} - 1 \quad (1)$$

in which $\max(a, b)$ is the maximum of two node degrees and $\min(a, b)$ is the opposite. Formula (1) makes the distance clear to zero when the sequences are identical and the d in the formula becomes more sensitive towards the difference between a, b when a, b are smaller.

Calculation of structural similarities between two subgraphs sg_1 and sg_2 can be converted to the issue of calculating the distance of degree sequence of nodes from those two corresponding subgraphs. As defined in the DTW algorithm, the distance can be turned into an optimization problem. The sequence distance obtained by the DTW algorithm is the distance of structural similarity of rooted subgraphs sg_1, sg_2 , which are denoted as $f(sg_1, sg_2)$.

Random Walk Process. Compared to structural similarities among rooted subgraphs, the similarity between two nodes is not meaningful, so in context generation, biased-graph2vec chooses a hyperparameter to avoid the calculation.

To generate enough context for single nodes and capture long distanced rooted subgraph's structural similarities, biased-graph2vec uses a two-layer network structure where context is captured through a cross-layer random walk. The two-layer walk model uses the similarity between the degree sequences of nodes to measure the structural similarity of nodes, and controls the walk jump probability through the similarity. The random walk process is shown in Fig. 3.

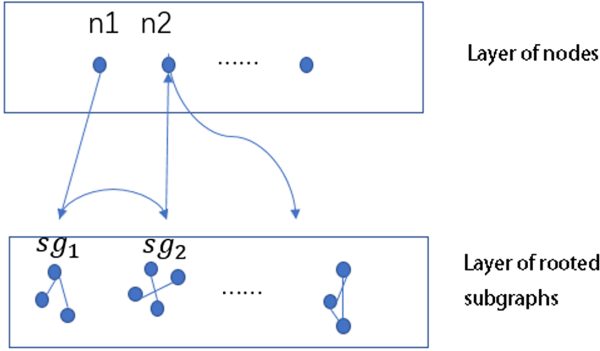


Fig. 3. The sketch map of the random walk process

The generated sequence of the random walk process acquired in Fig. 3 is denoted as $(n1, sg_1, sg_2, n2 \dots)$. The first layer contains all nodes from each graph in the graph set \mathbb{G} , while the second layer includes all rooted subgraphs. The structural similarities of nodes are represented via the calculation of similarities of nodes' rooted subgraphs. In the random walk process, each node in the node layer and each rooted subgraph in the subgraph layer will be start point and do the random walk process. The random walk process is performed to generate a fixed-length sequence and repeat the process multiple times. The number of walks performed from every start point and the length of each step will be taken as hyper-parameters to control the scale of context generated. The random walk process can be described in three conditions:

1. Jump from the node layer to rooted subgraph layer

The transition probability of skipping from node layer $layer_n$ to rooted subgraph layer $layer_{sg}$ is 1. For node v of the node layer as the current node, the destination of skipping will be chosen from all rooted graph sets that include node v . As defined in

formula (2), the likelihood of skipping is inversely proportional to the sum of the node's degree sequence of the currently rooted subgraph. If the number of nodes in a subgraph is smaller, or the degree of nodes is smaller, the structure of the root subgraph is simpler, that is, the sum of the sequence elements of the sequence composed of the degree of each node is smaller, then the probability of skipping to a rooted subgraph that contains node v and relatively simple in structure is more significant and vice versa. The probability of skipping from the node v of the node layer to rooted subgraph sg of the rooted subgraph layer can be calculated from the formula:

$$P_{layer_n \rightarrow layer_{sg}}(v, sg) = \frac{e^{-sum(s(V_{sg}))}}{M} \quad (2)$$

The $sum(s(V_{sg}))$ denotes the sum of the elements in the degree sequence of the current node. The normalizing factor M is defined as:

$$M = \sum_{sg' \in \Gamma_{sg}, sg' \neq sg, v \in sg'} e^{-sum(s(V_{sg'}))} \quad (3)$$

2. Jump from rooted subgraph layer to rooted subgraph layer

The structure of the rooted subgraph layer could be regarded as an undirected weighted graph, in which weight represents the transition probability in the random walk process. It is defined by structural similarities. In rooted subgraph layer, the transition probability is defined as followed:

$$p(sg_1, sg_2) = \frac{e^{-f(sg_1, sg_2)}}{Z} \quad (4)$$

Z is the normalizing factor defined as:

$$Z = \sum_{sg \in \Gamma_{sg}, sg \neq sg_1} e^{-f(sg_1, sg)} \quad (5)$$

The more similarity sg_1, sg_2 share, the smaller the similarity distance $f(sg_1, sg_2)$ will be, and synchronously, the probability of jump probability $p(sg_1, sg_2)$ grows.

3. Jump from rooted subgraph layer to node layer

The likelihood of skipping from rooted subgraph layer to the node layer is the hyper-parameter q . The skipping destination is a random node from all nodes contained in the current subgraph. For each node in the current subgraph, the probability is equal.

The context acquired from the random walk above and context from the process of rooted subgraph generation are merged as the context of biased-graph2vec.

Model Construction Based on Rooted Subgraphs. Every rooted subgraph in the rooted subgraph set is correspondingly fitted into word2vec as a word while graph set $D = \{d_1, d_2, \dots, d_N\}$ as the document. Applying the word2vec model and representation in a lower dimension of a graph and rooted subgraph in the same vector space is obtained. For that, the vocabulary size of the rooted subgraph is often relatively large. Applying negative sampling technology could effectively reduce the amount of

calculation. Biased-graph2vec uses SGD (Stochastic Gradient Descent) to optimize the parameters of the model.

The low dimension representation acquired could be applied to the following tasks. Moreover, it is convenient to use the traditional machine learning algorithm. For example, in graph classification tasks, we can directly feed the acquired vectors into classifying algorithms like SVM. In graph clustering tasks, the vectors obtained can be used as the input of clustering algorithms like K-means.

4 Experiments

4.1 Experimental Settings

Experimental Data. The six open benchmark data sets used in the experiment from the field of biochemistry are Enzymes, Mutag, Nci1, Nci109, Proteins, and Ptc. Table 1 below shows the statistics of the data sets. The six data sets are all multi classification data, and their classification standards range from different protein structures to whether they cause cancer to experimental mice, including multiple classification standards in structure and function. In this experiment, the graph representation ability of the model is proved by the classification task experiment under different classification standards on six datasets.

Table 1. Statistics of the graph sets

Data sets	Enzymes	Mutag	Nci1	Nci109	Proteins	Ptc
Number of samples	600	188	4110	4127	1113	344
Average degree	33.5	17.9	29.8	29.6	39.1	25.5
Graph label	7	2	2	2	2	2
Node label	44	7	37	38	3	19

The Enzyme contains proteinic structures from enzymes of 6 kinds, moreover, 100 protein structures per kind. Mutag contains 188 structures of compounds classified and separately labeled according to their capability of inducing a certain kind of bacteria mutation. Ptc contains 344 compound structures classified and labeled by the fact of whether they are carcinogenic to mice. Protein has 1113 amino acids' second-level structures. Nci1 and Nci109 contain compounds related to cancer cell researching, which respectively have the sample numbers of 4110 and 4127.

Base Line Methods. The baseline methods apply the subgraph2vec, graph2vec algorithms, and kernel method WL, which have been mentioned in content-related sections of the paper.

Subgraph2vec [3] aims at the disadvantage that the substructure of the graph kernel method is entirely independent. It takes graph's substructures as words in the text while

the graph as the document then applies a random walk and word embedding model to learn the representation of subgraphs. The algorithm applies the labeling process of the WL method to generate rooted subgraphs. Meanwhile, it modified the skip-gram model of the word2vec algorithm to make its inputting parameters able to suit vectors of indeterminate length.

Graph2vec [13] universally learns the representation of both graphs and substructures in the same characteristic space. The algorithm has two stages, first of which is to generate the rooted subgraphs, the second is to embed the vector of every rooted subgraph via doc2vec. The difference between the algorithm and subgraph2vec is that the model used in the second stage is different.

Weisfeiler-Lehman (WL) kernel [6] designs graph kernel algorithm from characteristics of the rooted subgraph. Compared with others, this is a better way to capture the characteristics of the graph itself.

In the experiment, the dimension of the vectors is uniformly set to 256. The length of the random walk sequence of biased-graph2vec and graph2vec is fixed to 10. The rooted subgraphs generated with the WL method have a max depth of 2, and the hyper-parameter of biased-graph2vec, q , is 0.3.

Evaluation Methodology. This section evaluates the effectiveness of the model with classification tasks. We use the accuracy rate as an evaluation indicator, which is to classify the nodes from their representations and evaluate the accuracy of the result. Process of calculation is as followed:

The vector representation of each rooted subgraph and node is obtained by modeling 90% of the data while verifying the representation in the remaining 10%. This paper applies SVM to classify the nodes of the remaining 10% to get accuracy. The formula of calculation of the accuracy is:

$$acc = \frac{TP + TN}{P + N} \times 100\% \quad (6)$$

P is the number of positive examples, while the N means the number of negative cases. Similarly, TP is the number of the positive examples that have been correctly classified and TN is the number of negative examples in their supposed places.

Graph Classification. The experiment was repeated ten times because some of the data sets were relatively small. In this paper, the average accuracy and standard deviation are taken as the evaluation index of the classification effect.

The parameters in the experiment are chosen via the grid-search method. The length of the random walk sequence is 15, and the walk has been performed five times. The probability of skipping in biased-graph2vec, q , is 0.1, and the depth of acquiring rooted subgraphs is 3.

Many results of the experiment are shown in Table 2.

In the multi-classification tasks, it is evident that the WL kernel method and the subgraph2vec have a rather poor performance. Probably for that, the lack of a unified model building of graphs and subgraphs leads to their vector representation emerging in different vector spaces.

Table 2. Accuracy of the experiment of graph classification

Data sets	Ptc	Proteins	Nci109	Nci1	Mutag	Enzymes
Biased-graph2vec	70.86 ± 2.40	77.17 ± 4.40	71.82 ± 1.35	72.02 ± 2.33	88.42 ± 6.25	77.50 ± 6.90
WL	58.78 ± 4.91	76.07 ± 1.16	69.93 ± 3.58	70.03 ± 1.35	67.11 ± 8.32	32.78 ± 8.34
Subgraph2vec	55.92 ± 9.01	75.18 ± 1.47	70.07 ± 3.21	71.14 ± 2.94	65.79 ± 8.89	44.00 ± 6.3
Graph2vec	68.10 ± 7.96	76.25 ± 4.75	67.19 ± 1.61	70.60 ± 1.60	86.47 ± 7.3	72.92 ± 4.52

In subgraph2vec, the output of the model is the representation of the substructure in the feature space; a similarity matrix of substructures is needed to get the representation of the graph at its final stage. Based on model doc2vec, biased-graph2vec uniformly does the model construction of both substructures and graphs. In this multi-classification experiment, doc2vec-based methods perform better.

In the experiment, biased-graph2vec is more effective than the other baseline methods in six data sets. It proves that compared with graph kernel method and method, ignoring the similarity among long distanced substructures, biased-graph2vec is more effective.

5 Summary

This paper proposed biased-graph2vec, the graph embedding model based on structural characteristics of rooted subgraphs, improved based on the vulnerability that existing projects have not done practical acquirement of the structural similarity of substructures. By building a suitable walking model for the substructures, biased random walks are performed to generate a moderate context. After that, the low dimensional representations of graphs and substructures are acquired via word embedding model. The experiment of graph classification proves the effectiveness of the model.

Directions of future research:

- (1) For that compared with other graph embedding models, the biased-graph2vec model adds an additional part, that is, context generation, which leads to more computing work in the model. Next, we will further study how to reduce the computing cost.
- (2) In the next step, we will also find out whether other effective fine-grained structures will not increase the calculation amount and can capture the similarity of substructures more precisely.

Acknowledgments. This work was supported by the National Natural Science Foundation of China (U1636219, 61602508, 61772549, U1736214, 61572052, U1804263, 61872448) and Plan for Scientific Innovation Talent of Henan Province (No. 2018JR0018).

References

1. Airola, A., Pyysalo, S., Björne, J., Pahikkala, T., Ginter, F., Salakoski, T.: All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC Bioinformatics* 9(S11), S2 (2008). <https://doi.org/10.1186/1471-2105-9-S11-S2>

2. Lancichinetti, A., Fortunato, S.: Community detection algorithms: a comparative analysis. *Phys. Rev. E* **80**(5), 056117 (2009)
3. Narayanan, A., Chandramohan, M., Chen, L., Liu, Y., Saminathan, S.: subgraph2vec: learning distributed representations of rooted sub-graphs from large graphs. arXiv preprint [arXiv:1606.08928](https://arxiv.org/abs/1606.08928) (2016)
4. Borgwardt, K.M., Kriegel, H.P.: Shortest-path kernels on graphs. In: Fifth IEEE International Conference on Data Mining, pp. 74–81. IEEE (2005)
5. Vishwanathan, S.V.N., Schraudolph, N.N., Kondor, R., Borgwardt, K.M.: Graph kernels. *J. Mach. Learn. Res.* **11**(Apr), 1201–1242 (2010)
6. Shervashidze, N., Schweitzer, P., Leeuwen, E.J.V., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-lehman graph kernels. *J. Mach. Learn. Res.* **12**(Sep), 2539–2561 (2011)
7. Kruskal, J.B.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika* **29**(1), 1–27 (1964). <https://doi.org/10.1007/BF02289565>
8. Balasubramanian, M., Schwartz, E.L.: The isomap algorithm and topological stability. *Science* **295**(5552), 7 (2002)
9. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. *Science* **290**(5500), 2323–2326 (2000)
10. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710. ACM, New York (2014)
11. 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, pp. 1067–1077. International World Wide Web Conferences Steering Committee, Florence (2015)
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, San Francisco (2016)
13. Prieto, L.P., Rodríguez-Triana, M.J., Kusmin, M., Laanpere, M.: graph2vec: learning distributed representations of graphs. arXiv preprint [arXiv:1707.05005](https://arxiv.org/abs/1707.05005) (2017)
14. Nguyen, D., Luo, W., Nguyen, T.D., Venkatesh, S., Phung, D.: Learning graph representation via frequent subgraphs. In: Proceedings of the 2018 SIAM International Conference on Data Mining, pp. 306–314. Society for Industrial and Applied Mathematics, San Diego (2018)
15. Ribeiro, L.F., Savarese, P.H., Figueiredo, D.R.: struc2vec: learning node representations from structural identity. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 385–394. ACM, Halifax (2017)