Multi-order Proximity Graph Structure Embedding

Wang Zhang[1,2], Lei Jiang[1], Huailiang Peng[1], Qiong Dai[1], and Xu Bai[1(✉)]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{zhangwang,jianglei,penghuailiang,daiqiong,baixu}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

Abstract. Graph embedding methods convert the flexible graph structure into low-dimensional representations while maintaining the graph structure information. Most existing methods focus on learning low- or high-order graph information, and cause loss of information during the embedding process. We instead propose a new method that can learn low and high order graph information simultaneously. The method fuses structure-preserving model with random walk sampling, which learns multi-order graph structure information more efficiently. Our method also utilizes distance-based weighted negative samples to improve the representations learning. The experimental results indicate that our proposed method provides very competitive results on the node classification, node clustering and graph reconstruction tasks for four benchmark datasets, BlogCatalog, PPI, Wikipedia and email-Eu-core.

Keywords: Deep learning · Unsupervised learning · Graph embedding

1 Introduction

Graph structure widely exists in many scenarios of real world. For example, the connection between people in social networks, the interaction of proteins in organisms, and the communication between IP addresses in communication networks, etc. In the past few decades, many effective graph analysis methods have been proposed for many applications, such as node classification [2], node clustering [8], link prediction [14] and so on. Graph structure is very flexible and complex, many existing graph analytics methods may suffer the high computation and space cost, and graph embedding is an effective solution for the above problems [3]. Various graph embedding methods usually try to learn a vector representation for each node in the graph, thus it is easy to process these vectors using traditional analysis methods.

H. Gao and X. Wang (Eds.): CollaborateCom 2021, LNICST 407, pp. 416–431, 2021.
https://doi.org/10.1007/978-3-030-92638-0_25

Recently, there are many studies on learning representations from graph data. For example, DeepWalk model [19] treats nodes as words, random walk sequences as sentences, and embedded vectors through the skipgram model [16]. Node2vec [9] uses a biased random walk, which is considered as an improved DeepWalk model that combines DFS and BFS. Another famous model, SDNE [27], takes advantage of an autoencoder structure to optimize first and second order proximity. Although the above methods are very effective, there are still some disadvantages. DeepWalk tends to learn the context information of nodes but ignores the neighborhood information. Node2vec has been improved, but the above situation still exists. SDNE focuses on first and second order proximity but neglects higher order proximity. Thus, the methods might not learn multi-order graph structure information very efficiently. Figure 1 shows embedding examples of the same Karate club graph.

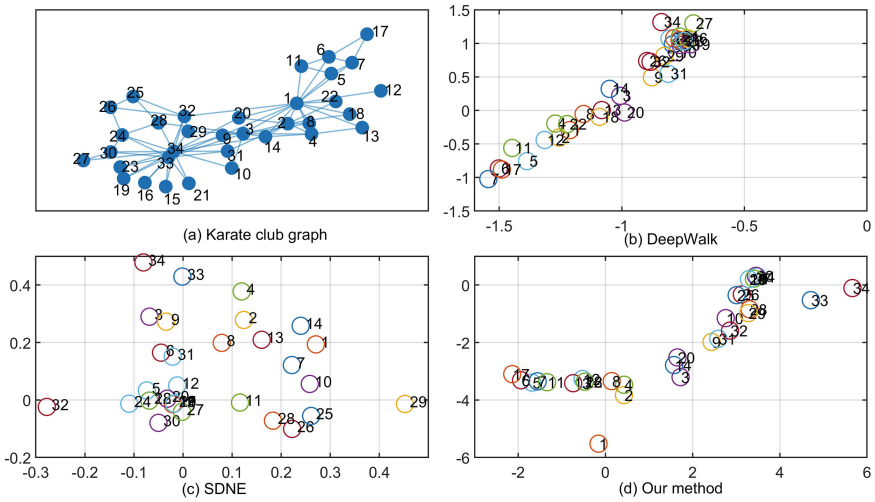

Fig. 1. Examples of three embedding methods on Karate club graph. It can be seen from the figure that the DeepWalk method effectively maintains the community structure, but ignores the low-order structure information within the community (e.g. node 2 and node 12 are very close). SDNE method ignores the community structure information, and the embedding representations are very scattered. Our method maintains not only community information, but also low-order structure.

To reduce loss of information in graph embedding, first, we introduce random walk sampling to structure-preserving model. Structure-preserving model can learn the low-order information of graph structure and generate embedding vectors containing the above information. Then, we sample the context of nodes in graph by random walk and force embedding vectors of nodes to learn their contextual information. Second, empirically, for a node, other nodes at different hop have different effects on it. Therefore, samples are weighted according to

their distance from the starting node during negative sampling. Each node will stay away from the nodes that are relatively far away. The experimental results indicate the superiority of our proposed method. Our main contributions are listed as follows:

- We present a new method that fuses structure-preserving model with random walk sampling for learning multi-order graph structure information.
- We introduce distance-based weighted negative samples, which can improve the ability of learning representations.
- The proposed model is evaluated on four benchmark datasets and experimental results show that our method provides competitive results on the node classification, node clustering and graph reconstruction tasks.

2 Related Work

There are the researches on graph embedding, and we briefly introduce the representative studies here.

Matrix Decomposition. These methods use matrices to store the relationships between nodes and obtain the embedding representation of nodes by matrix decomposition. For example, GraRep [4] uses SVD to decompose each k-step transition probability matrix to obtain the node representations, then concatenates all these k-step representations and gets final embedding vectors. It integrates global structural information into the learning process. HOPE proposes a directed graph embedding method for learning asymmetric transitivity [17]. Different from other studies, it learns two vectors for each node to preserve asymmetric relationship between nodes. M-NMF [28] proposes a unified framework of representation learning model based on NMF and community detection model based on modularity, in which the representation of nodes can retain both microscopic and community structure. AROPE [32] is an arbitrary order proximity preserved graph embedding method. It is used to solve the problem that the existing methods can only learn fixed order proximity.

Random Walk. These methods convert graph to node sequences with graph properties by random walk sampling, then apply skipgram [16] or other model on the sampled sequences to get representations. DeepWalk [19] is one of the most famous work among them. It uses random walk to sample node path in the graph to capture co-occurrence information between nodes, and then apply skipgram model to learn the vector representation of nodes. Different from DeepWalk, Node2vec [9] uses a biased random walk with return parameter p and in-out parameter q, which combines DFS and BFS. To make full use of the auxiliary information in graph, TriDNR [18] uses information from three parties: graph structure, node content, and node labels to jointly learn node representations.

Deep Learning. Due to the superiority of deep learning methods, these models are adopted into the tasks of graph embedding. SDNE [27], take advantage of an autoencoder structure, that aims to minimize reconstruction errors between encoder and decoder, for optimizing first and second order proximity. It can preserve both the local and global graph structure effectively even though graph is very sparse. GCN [13] is a new neural network model designed for embedding graph data through local first-order approximation of spectral graph convolution. It can utilize information from graph structure and node features to learn node representations. GAT [26], a neural network framework that works on graph structured data, uses masked self-attentional layers to aggregate node neighborhood information to learn its representation. Many previous approaches are transductive, [10] proposed an inductive framework that can generalize to unseen nodes by aggregating local neighborhood information of a node.

Others. Line [23], optimize first and second order proximity by maximizing edge reconstruction probability through edge sampling. It can be extended to large-scale networks and suitable for directed, undirected, weighted graphs. Verse [24] method learns node representations that preserve the distributions of any selected similarity measure among nodes by optimizing Kullback-Leibler divergence from the given similarity distribution to embeddings. NodeSketch [29] proposes a computationally efficient graph embedding technique that can preserve the proximity of higher-order nodes through recursive sketching.

In general, most existing methods focus on learning low- or high-order graph information, and how to effectively learn multi-order graph information is still a valuable question. Inspired by previous work, we fuse structure-preserving model with random walk sampling, in which embeddings preserve low-level (first- and second-order) and high-level (community structure) graph information.

3 Proposed Method

3.1 Notation

We define a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. We use A to denote the adjacency matrix. For an unweighted graph, if there exists an edge from v_i to v_j, $A_{i,j}$ equals one, and $A_{i,j}$ equals zero otherwise. For a weighted graph, the value of $A_{i,j}$ is the weight of the edge between nodes i and j. $A_{i,:}$ represents the i-th row of the adjacency matrix, and $A_{:,j}$ stands for the j-th column of the adjacency matrix. Graph embedding aims to learn a mapping function $f : V \rightarrow Z \in \mathbb{R}^{N \times d}$ from nodes to the learning representations, where N is equal to the number of nodes, d is the specified number of dimensions of our embedding representations and $d << |V|$, Z is an embedding representation, and $\mathbf{z}_i$ is the embedding representation of node i. We define $N(u)$ as the neighborhood of node u. Let $N_{RW}(u)$ denote the context nodes of node u obtained by random walk sampling. $N_{neg}(u)$ is defined as the negative samples of node u.

3.2 Structure Preserving Model

We adopt the structure in SDNE [27], which can preserve low order graph structure effectively. Our main model is an autoencoder structure composed of an encoder and a decoder. The encoder consists of multiple fully connected layer and performs non-linear mapping. The input of k-th layer is defined as $\mathbf{x}_i^k$, and the output $\mathbf{y}_i^k$ is calculated as follows:

$$\begin{aligned} \mathbf{x}_i^0 &= A_{:,i}, \quad \mathbf{x}_i^k = \mathbf{y}_i^{k-1} \\ \mathbf{y}_i^k &= \mathbf{W}^k \mathbf{x}_i^k + \mathbf{b}^k, \quad k = 1, ..., K \end{aligned} \tag{1}$$

The output of the encoder is $\mathbf{y}_i^K$, we take it as embedding representation $\mathbf{z}_i$ of the node i, and then restore it to the input of encoder through decoder. The calculation process of decoder is symmetric to encoder. Therefore, the autoencoder learns the second order proximity of graph by reducing the reconstruction loss. The reconstruction loss is defined as:

$$\mathcal{L}_{2nd} = \|(\hat{\mathbf{x}}_i - \mathbf{x}_i) \odot \mathbf{a}^i\|_2^2 \tag{2}$$

where $\hat{\mathbf{x}}_i$ is the output of decoder, $\mathbf{x}_i$ is equal to $A_{:,i}$, $\odot$ is the Hadamard product, and $\mathbf{a}_j^i = \beta$ if $A_{i,j} > 0$. The main function of the Hadamard product is to reconstruct the non-zero elements first. The loss of second order proximity guarantees that nodes with similar neighbors will be close to each other in embedding space.

The model also needs to learn first-order proximity, which means that if a pair of nodes i and j are connected, they should be close to each other. To preserve first-order proximity, a graph-based loss function is defined as:

$$\mathcal{L}_{1st} = -\frac{1}{|N(i)|} \sum_{j \in N(i)} log(\sigma(\mathbf{z}_i^\top \mathbf{z}_j)) \tag{3}$$

where σ is the sigmoid function ($\sigma(x) = \frac{1}{1+e^{-x}}$). The loss function of first order proximity also encourages nearby nodes to have similar representations in embedding space.

3.3 Random Walk Sampling

Although the structure-preserving model can effectively learn low-order graph structure information, it ignores high-order graph structure information. A graph can be represented as a set of information flows that contain community structure by random walk sampling [20]. We first randomly select a starting node, and uniformly sample a neighbor node of the last visited node until the maximum length is reached. The objective function of random walk sampling is as follows:

$$\begin{aligned} min_\Phi \; & -logP(v_{i-\Delta}, ..., v_{i-1}, v_{i+1}, ..., v_{i+\Delta} | v_i) \\ &= -log \prod_{-\Delta \le j \le \Delta} P(v_{i+j} | v_i) \\ &= -log \prod_{-\Delta \le j \le \Delta} \frac{exp(\Phi^\top(v_{i+j})\Phi(v_i))}{\sum_{k=1}^{|V|} exp(\Phi^\top(v_k)\Phi(v_i))} \end{aligned} \tag{4}$$

where the goal is to maximize the co-occurrence probability between nodes. However, due to softmax calculation is expensive, we do not use hierarchical softmax in DeepWalk, but use negative sampling. To preserve community proximity, loss function is defined as:

$$\mathcal{L}_{rw} = -\frac{1}{|N_{RW}(i)|} \sum_{j \in N_{RW}(i)} log(\sigma(\mathbf{z}_i^\top \mathbf{z}_j)) \tag{5}$$

3.4 Negative Sampling

In order to avoid the calculation of softmax, we introduce positive and negative samples. And we need to sample several nodes as negative samples for learning the ground truth effectively. Thus, we define a noise distribution $P_n(u)$ to pick negative samples for node u. Since many graphs are scale-free (the degree distribution of a graph follows the power law, as shown in Fig. 2), the noise distribution $P_n(u)$ we use here is actually a uniform distribution. And for each node, negative samples with different hops should have different weights. It is just like if the nodes in the graph are farther apart, then they should be "thrown" farther away in the embedding space. Therefore, we use the shortest distance between nodes as the weight of negative samples. The negative sample loss is defined as follows:

$$\mathcal{L}_n = -\frac{1}{|N_{neg}(i)|} \sum_{v_n \sim P_n(v_i)} s_{v_n} log(\sigma(-\mathbf{z}_i^\top \mathbf{z}_{v_n})) \tag{6}$$

where s_{v_n} is the distance between the negative sample v_n and the node v_i.

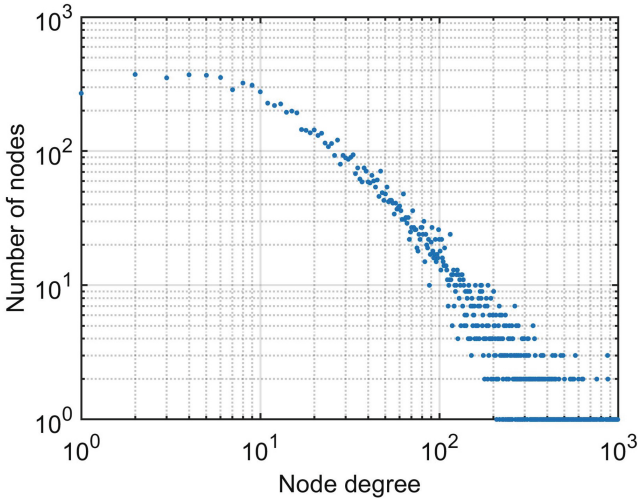

Fig. 2. Power law distribution of nodes in BlogCatalog dataset.

3.5 Algorithm

Objective Function. To learn low- or high-order graph information, and the final objective function is:

$$\mathcal{L} = \eta\mathcal{L}_{2nd} + \alpha\mathcal{L}_{1st} + \gamma\mathcal{L}_{rw} + \delta\mathcal{L}_{n} + \nu\mathcal{L}_{reg} \tag{7}$$

where $\mathcal{L}_{reg}$ is an L2-norm regularizer. Our proposed model only use one autoencoder structure with parameters $\boldsymbol{\theta}$ and does not introduce other structures. And we only adopt random walk strategy and negative sampling strategy. Our goal is to get reasonable parameters $\boldsymbol{\theta}$ by minimizing loss $\mathcal{L}$, and it is a weighted linear combination of per objective losses. Therefore, we introduce an efficient multi-objective optimization algorithm, MGDA-UB [22], to optimize $\mathcal{L}$. Because tuning these weights manually is very difficult. According to MGDA-UB, the resulting optimization problem is:

$$\min_{\alpha_1,\ldots,\alpha_T} \left\{ \left\| \sum_{t=1}^{T} \alpha_t \nabla_{\mathbf{Z}} \mathcal{L}_t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^{ta}) \right\|_2^2 \middle| \forall t, \sum_{t=1}^{T} \alpha_t = 1, \alpha_t \geq 0 \right\} \tag{8}$$

where $\boldsymbol{\theta}^{sh}$ are shared parameters, $\boldsymbol{\theta}^{ta}$ are task-specific parameters, $\mathbf{Z}$ denote the representations. In fact, the above optimization problem is equivalent to find a minimum-norm point in the convex hull of the set of input points [22]. MGDA-UB is computationally efficient, which requires only a single backward pass. After computing the gradient $\nabla_{\mathbf{Z}} \mathcal{L}_t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^{ta})$, the method uses Frank-Wolfe solver [12] to solve the optimization problem.

Optimization. To get the weights, we need to calculate the partial derivative $\nabla_{\mathbf{Z}} \mathcal{L}_t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^{ta})$. At first, we calculate the gradient $\nabla_{\mathbf{Z}} \mathcal{L}_{2nd}$, the detailed mathematical form is shown as follows:

$$\mathcal{L}_{2nd} = \|\beta(\hat{\mathbf{X}} - \mathbf{X})\|_2^2 \tag{9}$$

$$\frac{\partial \mathcal{L}_{2nd}}{\partial \hat{\mathbf{X}}} = 2\beta^2(\hat{\mathbf{X}} - \mathbf{X}) = \mathbf{H} \tag{10}$$

$$\mathrm{d}\mathcal{L}_{2nd} = \mathrm{tr}((\frac{\partial \mathcal{L}_{2nd}}{\partial \hat{\mathbf{X}}})^{\top} \mathrm{d}\hat{\mathbf{X}}) \tag{11}$$

The decoder here uses three fully connected layers. It can be phrased as follows:

$$\hat{\mathbf{X}} = \mathbf{W}_3\sigma_2(\mathbf{W}_2\sigma_1(\mathbf{W}_1\mathbf{Z} + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3 \tag{12}$$

To simplify the expression of the formula, it can be rephrased as:

$$\begin{aligned} &\mathbf{h}_2 = \mathbf{W}_1\mathbf{Z} + \mathbf{b}_1, \mathbf{g}_1 = \sigma_1(\mathbf{h}_2) \\ &\mathbf{h}_1 = \mathbf{W}_2\mathbf{g}_1 + \mathbf{b}_2, \mathbf{g}_2 = \sigma_2(\mathbf{h}_1) \\ &\hat{\mathbf{X}} = \mathbf{W}_3\mathbf{g}_2 + \mathbf{b}_3 \end{aligned} \tag{13}$$

Then the differential form of $\mathcal{L}_{2nd}$ can be expressed as follows:

$$\begin{aligned}
\mathrm{d}\mathcal{L}_{2nd} &= \mathrm{tr}((\frac{\partial \mathcal{L}_{2nd}}{\partial \hat{\mathbf{X}}})^\top \mathrm{d}\hat{\mathbf{X}}) = \mathrm{tr}(\mathbf{H}^\top \mathbf{W}_3 \mathrm{d}\mathbf{g}_2) \\
&= \mathrm{tr}(\mathbf{H}^\top \mathbf{W}_3(\sigma_2^{'}(\mathbf{h}_1) \odot (\mathbf{W}_2 \mathrm{d}\mathbf{g}_1))) \\
&= \mathrm{tr}(\underbrace{(\mathbf{W}_3^\top \mathbf{H} \odot \sigma_2^{'}(\mathbf{h}_1))^\top \mathbf{W}_2}_{\mathbf{H}_1} \mathrm{d}\mathbf{g}_1) \\
&= \mathrm{tr}(\mathbf{H}_1(\sigma_1^{'}(\mathbf{h}_2) \odot (\mathbf{W}_1 \mathrm{d}\mathbf{Z}))) \\
&= \mathrm{tr}(\underbrace{(\mathbf{H}_1^\top \odot \sigma_1^{'}(\mathbf{h}_2))^\top \mathbf{W}_1}_{\mathbf{H}_2} \mathrm{d}\mathbf{Z}) \\
\frac{\partial \mathcal{L}_{2nd}}{\partial \mathbf{Z}} &= \mathbf{H}_2^\top
\end{aligned} \tag{14}$$

Then we can calculate the gradient $\nabla_{\mathbf{Z}} \mathcal{L}_{1st}$. From Eq. 3, it can be phrased as follows:

$$\frac{\partial \mathcal{L}_{1st}}{\partial \mathbf{z}_i} = (-\frac{1}{|N(i)|} \sum_{j \in N(i)} \sigma_3^{'}(\mathbf{z}_i^\top \mathbf{z}_j)\mathbf{z}_j^\top)^\top \tag{15}$$

Similarly, it is easy to calculate the gradient $\nabla_{\mathbf{Z}} \mathcal{L}_{rw}$ and $\nabla_{\mathbf{Z}} \mathcal{L}_{neg}$. And the above σ_1, σ_2 and σ_3 are element-wise functions. After obtaining these partial derivatives, we can make use of MGDA-UB method to get the weights of above objectives. Then the parameters $\boldsymbol{\theta}$ can be optimized by using stochastic gradient descent. We present the our algorithm as follows (see Algorithm 1).

Algorithm 1: Multi-order graph embedding algorithm

Input: graph $G = (V, E)$, adjacency matrix A
Output: embedding representations Z

while *Parameter convergence* **do**
  Walks = RandomWalkSampling(G);
  Select nodes from V as a minibatch B;
  $\mathbf{Z}_B, \hat{\mathbf{X}}_B$ = Autoencoder($\mathbf{X}_B$;$\boldsymbol{\theta}$);
  foreach $v \in B$ **do**
    Select the context nodes of node v as $N_{RW}(v)$ from Walks;
    Select the neighbors of node v as $N(v)$;
    Select the negative samples of node v as $N_{neg}(v)$;
  end
  $\alpha_1, ..., \alpha_T$ = MGDA-UB($\nabla_{\mathbf{Z}_B} \mathcal{L}_1, ..., \nabla_{\mathbf{Z}_B} \mathcal{L}_T$);
  Update parameters $\boldsymbol{\theta}$ with stochastic gradient descent based on $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}$;
end
$\mathbf{Z}, \hat{\mathbf{X}}$ = Autoencoder($\mathbf{X}$;$\boldsymbol{\theta}$);
return embedding representations $\mathbf{Z}$;

4 Evaluation

4.1 Experimental Setup

Datasets. In order to evaluate the effectiveness of our method, we use four public graph datasets, including one social network, BlogCatalog,[1] one words co-occurrence network, Wikipedia,[2] a subgraph of the Protein-Protein Interactions network (Homo Sapiens),[3] and an email communication network between institution members (the core)[4] for node classification, graph reconstruction and node clustering tasks. The statistics of the above datasets can be summarized in Table 1.

Table 1. Statistics of the datasets

Datasets	#Node	#Edge	#Category	Note
BlogCatalog	10312	333983	39	Multi-label
Wikipedia	4777	92295	40	Multi-label
PPI	3890	37845	50	Multi-label
email-Eu-core	1005	25571	42	Single-label

Evaluation Metrics. We validated our method in three tasks. Specifically, we use Hamming Loss [7] in multi-label node classification task, which is often used in multi-label tasks [30,31]. And two well known F1 measures, Macro-F1 and Micro-F1 [25], were used to evaluate the performance in single-label node classification task. In graph reconstruction, we adopted Mean Average Precision (MAP) [27]. In node clustering, Normalized Mutual Information (NMI) [5] is often used in finding non-overlapping communities. However, we here used an improved Normalized Mutual Information (MGH-NMI) [15] to detect overlapping communities.

Baselines. In our experiments, we mainly verify the ability to maintain the structure information of undirected graphs. Thus, we compare our method with the following state-of-the-art methods: DeepWalk [19], Node2vec [9], GraRep [4], Line [23], SDNE [27], AROPE [32], M-NMF [28] and NodeSketch [29]. For the first five methods, we use the open source tool, OpenNE,[5] for experiments. The implementation of AROPE used in our experiments is available at https://github.com/ZW-ZHANG/AROPE. For M-NMF and NodeSketch, we use the open source tool, Karate Club [21] in our experiments.

[1] http://socialcomputing.asu.edu/datasets/BlogCatalog3.
[2] http://snap.stanford.edu/node2vec/POS.mat.
[3] http://snap.stanford.edu/node2vec/.
[4] http://snap.stanford.edu/data/email-Eu-core.html.
[5] https://github.com/thunlp/OpenNE.

4.2 Experimental Results

Node Classification. We used the multi-layer perceptron [11] without activation for multi-label node classification and use linear support vector machine [6] for single-label node classification. We also adopted 10-fold cross validation in this experiment. M-NMF and NodeSketch methods in Karate Club tool can not run on the disconnected graphs. Therefore, there are no experimental results on PPI and email-Eu-core. The experimental results of node classification are listed in Table 2. Our method provides competitive results in multi-label and single-label node classification tasks. And it shows that our method with MGDA-UB is always superior to the other methods. This also demonstrates that dynamic weights of multi-objective can help optimize the effect of node classification. SDNE or Line mainly preserves first and second order proximity of graph. Thus, DeepWalk or Node2vec always perform better than SDNE or Line, which indicates that the high-order graph structure information needs to be effectively learned. Overall, our method effectively learns the high-order graph structure information and provides competitive results in node classification tasks.

Table 2. Experimental results of node classification task.

	BlogCatalog	Wikipedia	PPI	email-Eu-core	
	Hamming loss	Hamming loss	Hamming loss	Macro-F1	Micro-F1
DeepWalk	0.0385 ± 0.0012	0.0414 ± 0.0009	0.0456 ± 0.0008	0.600 ± 0.061	0.752 ± 0.049
SDNE	0.0416 ± 0.0012	0.0394 ± 0.0011	0.0433 ± 0.0013	0.537 ± 0.057	0.694 ± 0.046
Node2vec	0.0392 ± 0.0008	0.0429 ± 0.0011	0.0454 ± 0.0010	0.593 ± 0.057	0.753 ± 0.039
Line	0.0441 ± 0.0010	0.0430 ± 0.0011	0.0543 ± 0.0015	0.522 ± 0.066	0.689 ± 0.063
GraRep	0.0349 ± 0.0010	0.0384 ± 0.0009	0.0419 ± 0.0010	0.593 ± 0.064	0.766 ± 0.048
AROPE	0.0409 ± 0.0011	0.0366 ± 0.0011	0.0402 ± 0.0013	0.593 ± 0.059	0.753 ± 0.051
M-NMF	0.0360 ± 0.0007	0.0343 ± 0.0011	–	–	–
NodeSketch	0.0417 ± 0.0020	0.0477 ± 0.0029	–	–	–
Our method[a]	0.0344 ± 0.0010	0.0377 ± 0.0011	0.0399 ± 0.0012	**0.609** ± 0.081	0.774 ± 0.048
Our method-M[b]	**0.0333** ± 0.0007	**0.0334** ± 0.0009	**0.0368** ± 0.0012	0.603 ± 0.034	**0.775** ± 0.031

[a] This is the method with balanced weights of each objective.
[b] This is the method with dynamic weights of each objective.

Graph Reconstruction. In this experiment, we used inner products to measure the similarity of vectors. The performance results in Table 3 illustrate that our method is significantly superior to the other baselines on four benchmark datasets. In addition, compared to the second-ranked method, our method improved by 140%, 5.3%, 72.6% and 66.1%, respectively. In fact, the contribution mainly comes from the distance-based negative sampling method, and we will discuss in the section of ablation study. And it shows that our method with MGDA-UB performs not well. This is because it tries to reduce losses but does not effectively maintain the balance of tasks. In general, it also shows that our method can effectively learn low-order graph structure information.

Table 3. Experimental results of graph reconstruction task.

	BlogCatalog	Wikipedia	PPI	email-Eu-core
DeepWalk	0.138	0.416	0.224	0.340
SDNE	0.038	0.077	0.085	0.329
Node2vec	0.203	0.451	0.285	0.233
Line	0.013	0.085	0.126	0.276
GraRep	0.060	0.185	0.137	0.384
AROPE	0.199	0.348	0.151	0.218
M-NMF	0.004	0.336	–	–
NodeSketch	0.010	0.044	–	–
Our method	**0.488**	**0.475**	**0.492**	**0.638**
Our method-M	0.186	0.348	0.181	0.615

Node Clustering. K-means algorithm [1] was adopted to cluster nodes in this experiment. The performance results for node clustering task are shown in Table 4. PPI is a multi-graph data set and is not suitable for node clustering task. It can be seen that our method is superior to other methods in overlapping graphs. And it shows that our method with MGDA-UB performs not very well. It also shows that the experimental results of SDNE or Line are not very good because they mainly learn low-order structural information, and they are not suitable for the task. The performance results indicate that our method can effectively learn the high-order graph structure information.

Table 4. Experimental results of node clustering task.

	BlogCatalog	Wikipedia	email-Eu-core
	MGH-NMI	MGH-NMI	NMI
DeepWalk	0.0213	0.0017	0.700
SDNE	0.0	0.0040	0.575
Node2vec	0.0226	0.0024	0.702
Line	0.0	0.0022	0.665
GraRep	0.0264	0.0019	0.677
AROPE	0.0	0.0028	0.460
M-NMF	0.0049	0.0019	–
NodeSketch	0.0	0.0014	–
Our method	**0.0387**	0.0032	**0.708**
Our method-M	0.0097	**0.0041**	0.676

4.3 Ablation Study of Negative Sampling

Since we use distance-based negative sampling method to improve learning representations. In this section, we investigate the contributions of distance-based negative sampling to the performance. Some variants of our method are as follows:

- Variant-1: It learns low- or high-order graph information without negative sampling.
- Variant-2: Every negative sample has the same weight.
- Variant-3: It performs negative sampling based on the degree of nodes.

First, we conduct an experiment by removing negative sampling, where our method learns low and high order graph structure information with only positive sample. Then, we conduct an experiment where all negative samples have equal weight. The performance results in Table 5 illustrate that distance-based negative sampling can effectively improve the ability to learn representations. The method without negative sampling does not perform well, especially in node clustering and graph reconstruction tasks. This shows that we need not only positive samples in the training process, but also negative samples as noise to improve the model. And the experimental results of the method with equal weight negative samples have improved, but there is still a gap between the distance-based negative sampling method. If we do negative sampling based on the degree of nodes, it shows that does not perform well in node classification and node clustering. This is because negative sampling relies too much on nodes with large degrees and neglects most nodes with small degrees. These results demonstrate the contributions of distance-based negative sampling method.

Table 5. Experimental results about ablation study of negative sampling on BlogCatalog.

	Hamming loss	MGH-NMI	MAP
Our method	0.0344 ± 0.0010	0.0387	0.488
Variant-1	0.0371 ± 0.0009	0.0131	0.058
Variant-2	0.0348 ± 0.0011	0.0159	0.418
Variant-3	0.0363 ± 0.0009	0.0150	0.547

4.4 Study of Hyper Parameters

We investigate the how different hyper parameters affect the performance in this section. Specifically, we evaluate the results in terms of number of context, negative samples and embedding dimensions.

Performance on Different Number of Context. We first show how the number of context affects the performance in Table 6. We can see that it has little effect on node classification. In node clustering, results first increase with the growth of number of context and then remain stable. In graph reconstruction, results first increase and then drop, and it achieves the best performance of reconstruction at 24. This shows that learning too many context nodes at the same time is not good for identifying neighbors.

Performance on Different Number of Negative Samples. From Table 6, we can see that it has a little effect on node classification. However, it has a greater impact on graph reconstruction. The more negative samples, the better the results of graph reconstruction task. Too many negative samples will negatively affect node classification and clustering tasks, since this reduces the learning effect of context nodes.

Performance on Different Embedding Dimensions. At last, we show how different embedding dimensions affect performance. From Table 6, we can see that the results of our method become worse as the dimensions increase in node classification. In node clustering, when the dimensions increase, the performance initially gets better and then becomes worse. The reason is that too large dimensions will introduce noise, which has negative impact on performance. The influence on the task of graph reconstruction is volatile.

Table 6. Experimental results of different hyper parameters on BlogCatalog.

	Hamming loss	MGH-NMI	MAP
context = 8	0.0347 ± 0.0012	0.0252	0.473
context = 16	0.0344 ± 0.0010	0.0387	0.472
context = 24	0.0344 ± 0.0010	0.0388	0.488
context = 32	0.0345 ± 0.0006	0.0391	0.482
context = 48	0.0343 ± 0.0008	0.0392	0.444
neg = 8	0.0341 ± 0.0007	0.0393	0.427
neg = 16	0.0342 ± 0.0008	0.0384	0.459
neg = 24	0.0344 ± 0.0010	0.0388	0.488
neg = 32	0.0343 ± 0.0008	0.0388	0.486
neg = 48	0.0349 ± 0.0011	0.0378	0.497
dim = 32	0.0322 ± 0.0006	0.0334	0.403
dim = 64	0.0332 ± 0.0007	0.0331	0.472
dim = 96	0.0340 ± 0.0009	0.0390	0.453
dim = 128	0.0344 ± 0.0010	0.0388	0.488
dim = 256	0.0372 ± 0.0009	0.0278	0.494

5 Conclusion

In this paper, we present a new graph embedding method that focuses on learning low and high order graph structure information. Our method fuses structure-preserving model with random walk sampling, which can learn multi-order graph structure information simultaneously. Our method also introduces distance-based negative sampling method for improving the learning representations by collecting noise samples. It measures the importance of negative samples through the distance between nodes. We evaluate our embedding method in node classification, graph reconstruction and node clustering tasks. The experimental results demonstrate that our proposed method provides very competitive results compared with seven state-of-the-art baselines on four benchmark datasets.

Acknowledgement. This paper is Supported by National Key Research and Development Program of China (Grant No. 2017YFB0803003) and National Science Foundation for Young Scientists of China (Grant No. 61702507).

References

1. Arthur, D., Vassilvitskii, S.: k-means++: the advantages of careful seeding. Technical report, Stanford (2006)
2. Bhagat, S., Cormode, G., Muthukrishnan, S.: Node classification in social networks. In: Social Network Data Analytics, pp. 115–148. Springer, Boston (2011). https://doi.org/10.1007/978-1-4419-8462-3_5
3. Cai, H., Zheng, V.W., Chang, K.C.C.: A comprehensive survey of graph embedding: problems, techniques, and applications. IEEE Trans. Knowl. Data Eng. **30**(9), 1616–1637 (2018)
4. Cao, S., Lu, W., Xu, Q.: GraRep: learning graph representations with global structural information. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, pp. 891–900. ACM (2015)
5. Cavallari, S., Zheng, V.W., Cai, H., Chang, K.C.C., Cambria, E.: Learning community embedding with community detection and node embedding on graphs. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 377–386. ACM (2017)
6. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
7. Gibaja, E., Ventura, S.: A tutorial on multilabel learning. ACM Comput. Surv. (CSUR) **47**(3), 52 (2015)
8. Girvan, M., Newman, M.E.: Community structure in social and biological networks. Proc. Natl. Acad. Sci. **99**(12), 7821–7826 (2002)
9. 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 (2016)
10. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, pp. 1024–1034 (2017)
11. Hinton, G.E.: Connectionist learning procedures. In: Machine Learning, pp. 555–610. Elsevier (1990)

12. Jaggi, M.: Revisiting Frank-Wolfe: projection-free sparse convex optimization. In: Proceedings of the 30th International Conference on Machine Learning, pp. 427–435 (2013)
13. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, ICLR 2017 (2017). https://openreview.net/forum?id=SJU4ayYgl
14. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. J. Am. Soc. Inf. Sci. Technol. **58**(7), 1019–1031 (2007)
15. McDaid, A.F., Greene, D., Hurley, N.: Normalized mutual information to evaluate overlapping community finding algorithms. arXiv preprint arXiv:1110.2515 (2011)
16. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
17. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1105–1114. ACM (2016)
18. Pan, S., Wu, J., Zhu, X., Zhang, C., Wang, Y.: Tri-party deep network representation. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, pp. 1895–1901. AAAI Press (2016)
19. 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 (2014)
20. Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. Proc. Natl. Acad. Sci. **105**(4), 1118–1123 (2008)
21. Rozemberczki, B., Kiss, O., Sarkar, R.: An API oriented open-source python framework for unsupervised learning on graphs. arXiv preprint arXiv:2003.04819 (2020)
22. Sener, O., Koltun, V.: Multi-task learning as multi-objective optimization. In: Advances in Neural Information Processing Systems, pp. 527–538 (2018)
23. 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 (2015)
24. Tsitsulin, A., Mottin, D., Karras, P., Müller, E.: VERSE: versatile graph embeddings from similarity measures. In: Proceedings of the 2018 World Wide Web Conference, pp. 539–548. International World Wide Web Conferences Steering Committee (2018)
25. Uysal, A.K., Gunal, S.: A novel probabilistic feature selection method for text classification. Knowl.-Based Syst. **36**, 226–235 (2012)
26. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: 6th International Conference on Learning Representations, ICLR 2018 (2018). https://openreview.net/forum?id=rJXMpikCZ
27. 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, pp. 1225–1234. ACM (2016)
28. Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., Yang, S.: Community preserving network embedding. In: Thirty-first AAAI Conference on Artificial Intelligence (2017)
29. Yang, D., Rosso, P., Li, B., Cudre-Mauroux, P.: NodeSketch: highly-efficient graph embeddings via recursive sketching. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1162–1172 (2019)

30. Yang, P., Sun, X., Li, W., Ma, S., Wu, W., Wang, H.: SGM: sequence generation model for multi-label classification. In: Proceedings of the 27th International Conference on Computational Linguistics, pp. 3915–3926 (2018)
31. Yang, Y.Y., Lin, Y.A., Chu, H.M., Lin, H.T.: Deep learning with a rethinking structure for multi-label classification. arXiv preprint arXiv:1802.01697 (2018)
32. Zhang, Z., Cui, P., Wang, X., Pei, J., Yao, X., Zhu, W.: Arbitrary-order proximity preserved network embedding. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2778–2786. ACM (2018)