Knowledge-Based Systems 208 (2020) 106448

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Manifold graph embedding with structure information propagation for community discovery

Shuliang Xu [a], Shenglan Liu [b], Lin Feng [b,*]

[a] Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian, China
[b] School of Innovation and Entrepreneurship, Dalian University of Technology, Dalian, China

A R T I C L E   I N F O

Article history:
Received 4 March 2020
Received in revised form 6 September 2020
Accepted 8 September 2020
Available online 16 September 2020

Keywords:
Graph embedding
Community discovery
Matrix factorization
Low rank learning
Clustering analysis

A B S T R A C T

Community discovery is an important topic of network representation learning. Manifold learning has been widely applied to network representation learning. However, most manifold learning algorithms do not consider the asymmetry of edges which is not accord with the structure of social networks because the influence of nodes is not symmetrical. In this paper, a community discovery algorithm based on manifold graph embedding with structure information propagation mechanism is proposed. The proposed algorithm uses high order approximation matrix to obtain the local and global structure information of a graph, then low rank decomposition is introduced to obtain the node vectors and the context vectors. Finally, the node vectors can be adjusted by structure information. The proposed algorithm and comparison algorithms are conducted on the experimental data sets. The experimental results show that the proposed algorithm outperforms the comparison algorithms on the most experimental data sets. The experimental results prove that the proposed algorithm is an effective algorithm for community discovery.

1. Introduction

With the development of social media, graph embedding has attracted much attention in recent years [1–3]. Graph embedding is to obtain the low dimension representation of network nodes and the embedding result has a direct influence on the results of community discovery. Because social network is unstructured data, how to obtain better low dimension representation of network nodes is still an open problem. So far, researchers have proposed many approaches to obtain the embedding vectors of nodes. For those approaches, they can be divided into three classes: random walk, matrix factorization and deep learning.

The graph embedding approach based on random walk is to generate a node sequence as the context, then node vectors are obtained by word2vec [4]. The representative methods include DeepWalk [5], node2vec [6], LINE [7], subgraph2vec [8], struc2vec [9] and metapath2vec [10], etc. The above approaches have been proved that they can effectively deal with sparse and large-scale social network data sets. However, those approaches lack a clear optimization objective and the local and global information of the network are separately learned. The final result is simply connecting the two expression vectors [11].

Matrix factorization is also common in community discovery. The graph embedding algorithm with matrix factorization mechanism is to decompose a matrix into two or multiple sub-matrices and the dimension of final sub-matrix is lower than the matrix. The matrix factorization algorithms have a good explanation. It has been developed many algorithms, such as NMTF [12], AROPE [13], M-NMF [14], NetSMF [15], NECS [16] and ProNE [17], etc. However, many the graph embedding algorithms with matrix factorization mechanism do not consider the asymmetry of edges because the influence of nodes is not symmetrical. In addition, the time complexity of the most algorithms is also high.

Deep learning is an effective algorithm for graph embedding. There are many deep graph embedding algorithms having been proposed. Deep graph embedding algorithm uses nonlinear activation function (such as relu, sigmoid, tanh, etc.) to fit nonlinear data and it can obtain more abstract and high-level representation features by combining low-level features. Therefore deep graph embedding algorithms have better a performance. Up to now, many deep graph embedding algorithms have been proposed such as DGI [18], SPEM [19], DAEGC [20] and GREEN [21], etc. However, the explanation of deep embedding algorithms is weak, there are many parameters for the deep embedding algorithms and it is not easy to select appropriate parameters.

For social network, the inter influence of the two modes in the same edge is not symmetric. It is means that a node may be important for another node and the importance in the reverse direction is not as large as the positive direction. The above

* Corresponding author.
E-mail addresses: xushulianghao@126.com (S. Xu), fenglin@dlut.edu.cn (L. Feng).

https://doi.org/10.1016/j.knosys.2020.106448
0950-7051/© 2020 Elsevier B.V. All rights reserved.

phenomenon is easy to understand. For example, the behaviors of stars can easily affect general public, but the behaviors of general public cannot easily affect stars. In the current, many graph embedding algorithms ignore the asymmetry. In order to address the asymmetry of edges, a manifold graph embedding with structure information propagation for community discovery (MGE) is proposed in this paper. MGE algorithm considers the degree of a node when it computes the weight of an edge and the weight is determined by the degree and the similarity of nodes. Therefore the embedding result of MGE algorithm considers the asymmetry of edges. In addition, MGE algorithm uses high order approximation matrix to obtain the local and global structure information of a graph. The low dimensional node vectors and context vectors can be obtained from matrix factorization and low rank learning. The main contributions of this paper are as follows:

- A manifold graph embedding algorithm is proposed in this paper. The proposed algorithm considers the asymmetry of edges which is a significant characteristic of social network.
- A structure information propagation mechanism is introduced in the proposed algorithm. The final embedding result can be adjusted by structure information.
- The proposed algorithm uses high order approximation matrix to obtain the local and global structure information of a graph. It is different from traditional algorithms that the proposed algorithm considers the influence of node itself and the weight of an edge is determined by the degree of node and the cosine similarity of the two nodes.

The rest of this paper are organized as follow: Section 2 reviews some related works; Section 3 introduces the details of MGE algorithm; the experiments and results are in Sections 4 and 5 concludes the paper.

2. Related works

Graph embedding has recently become a paradigm to represent network nodes by low dimensional vectors. Unstructured graph data is more and more common in practical applications and graph embedding has attracted much attention from researchers. There are many related works having been published in recent years.

Yang et al. propose a fast network embedding algorithm called NEU [22]. NEU algorithm is based on random walk and uses high order proximity approximation to obtain the global information of network; then introduces spectral decomposition, singular value decomposition and word2vec to compute the low dimensional embedding vectors according to the high order proximity approximation matrix which forms three different kinds of algorithms.

Chen et al. propose a fast and accurate network embedding algorithm called FastRP [23]. FastRP algorithm uses random projection and projects the adjacency matrix into a sparse and low dimensional matrix. Random projection matrix is a sparse matrix and generated from normal distribution. The final embedding matrix is obtained by weighted combination. The result of FastRP algorithm completely dependents on random projection and there is no matrix factorization. Therefore FastRP algorithm has an obvious advantage of speed.

Long et al. propose a network embedding algorithm which can preserve the hierarchical community structure called SpaceNE [24]. SpaceNE algorithm is a subspace learning algorithm and preserves three constraints: approximate constraint of point pair, similarity constraint of hierarchical structure and low rank constraint which preserves the low rank property of subspace. The hierarchical community structure is related to the depth, therefore SpaceNE algorithm can be seen as a neural network algorithm.

Qiu et al. propose a sparse matrix factorization embedding algorithm for large-scale network called NetSMF [15]. NetSMF algorithm uses random walk to generate a subgraph. The number of vertexes is equal to the number of vertexes of the original graph and the number of edges is far less than the number of edges of the original graph. Therefore it is equal to transform original adjacency matrix into a sparse matrix. The co-occurrence matrix of subgraph is also computed. Then the low dimensional embedding vectors and context vectors can be obtained by eigenvalue decomposition. It is obvious that the performance of NetSMF algorithm is sensitive to the generated subgraph.

Wu et al. propose a manifold embedding algorithm for attributed network called MARINE [25]. MARINE algorithm introduces manifold regularization and the constraint of maximum mean discrepancy in a reproducing kernel Hilbert space and uses manifold learning and kernel method to obtain embedding vectors. The final embedding vectors can be computed by iterative solution.

For the related works, adjacency matrix is symmetric and the asymmetry of edges is not considered. It means the influence of the nodes in the network is also symmetric which is not in line with the reality of social network. In the real world, the mutual influence of nodes in social network is often asymmetric. Therefore the proposed algorithm considers the asymmetry of edges and uses the asymmetric high order approximation matrix to obtain the local and global structure information of a graph which does not ignore the asymmetric mutual influence of nodes in social network.

3. Proposed method

For a graph $G = \langle V, E \rangle$, V is the set of vertexes and E is the set of edges. Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be the adjacency matrix of graph G. For $\forall v_i, v_j \in V$ and $e_{ij} \in E$, then $A_{ij} = 1$; if $v_i, v_j \in V$ and $e_{ij} \notin E$, then $A_{ij} = 0$. Graph embedding is to learn a mapping function f: $V \mapsto \mathbb{R}^{d\times n}$ which projects each node into a d-dimensional space ($d \ll n$) and preserves the structure similarity of node pairs.

3.1. Manifold graph embedding with low rank decomposition

Let $\boldsymbol{A} \leftarrow \boldsymbol{D}^{-1}\boldsymbol{A}$, $\boldsymbol{D}$ be a diagonal matrix and $D_{ii} = \sum_{j=1}^{n} A_{ji}$. It is known that the high order approximation includes the global information of graph. $\boldsymbol{M}$ is defined as follow:

$$\boldsymbol{M} = \frac{1}{2}\boldsymbol{A} + \frac{1}{2^2}\boldsymbol{A}^2 + \cdots + \frac{1}{2^r}\boldsymbol{A}^r = \sum_{i=1}^{r} \frac{1}{2^i}\boldsymbol{A}^i \quad (1)$$

where r is the order of the adjacency matrix. A large r means $\boldsymbol{A}^r$ includes the more global information. A small r means $\boldsymbol{A}^r$ includes the more local information. Therefore $\boldsymbol{M}$ includes the global and local information. We can use $\boldsymbol{M}$ to obtain the low dimensional embedding of nodes.

In manifold learning, data can be reconstructed from the neighborhood data points [26]. Let $\boldsymbol{y}_i$ be the embedding vector of v_i and $\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n] \in \mathbb{R}^{d\times n}$. Therefore the manifold regularization is as

$$\min_{\boldsymbol{Y}} \sum_{i,j} W_{ij} \left\| \boldsymbol{y}_i - \boldsymbol{y}_j \right\|_F^2 \quad (2)$$

where W_{ij} is the weight between v_i and v_j. For social network, W_{ij} is not symmetrical and $W_{ij} \neq W_{ji}$.

In social network, the degree of vertex is an important measure and a large degree means the vertex has a large influence.

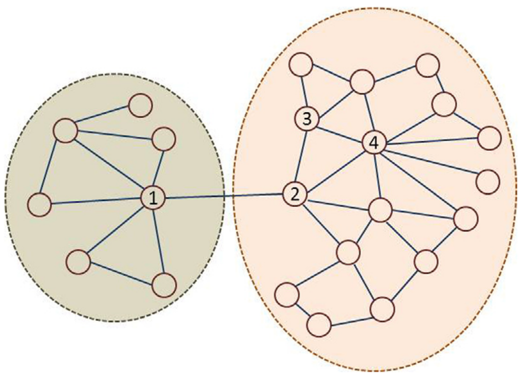

Fig. 1. The weights of edges in social network are not symmetrical.

As is showed in Fig. 1, it is known that the weights of edges in social network are not symmetrical; in other words, the influence of two nodes is not symmetrical. For node 1, there are 6 neighbor nodes. The degree of node 2 is 5 which is the largest degree in the neighbor nodes of node 1. Therefore node 2 is the most important nodes for node 1 in its neighbor nodes. For node 2, there are 5 neighbor nodes. The node with the largest degree is node 4 in the neighbor nodes. It means the most important node for node 2 is node 4 and node 1 is not the most important node for node 2 although node 2 has the largest influence on node 1. Therefore the weight between v_i and v_j is determined as

$$W_{ij} = \begin{cases} \dfrac{\theta}{1+\exp\left(-\left(M_{ij}+\lambda\cdot Dgree\left(v_j\right)\right)\right)} & \theta \geq \tau \\ 0 & \theta < \tau \quad \text{or} \quad v_i = v_j \end{cases} \tag{3}$$

where $\theta = \frac{\boldsymbol{M}_{i,:}\cdot\boldsymbol{M}_{j,:}}{\|\boldsymbol{M}_{i,:}\|_2\cdot\|\boldsymbol{M}_{j,:}\|_2}$, $\boldsymbol{M}_{i,:}\cdot\boldsymbol{M}_{j,:}$ is the inner product, $\|\cdot\|_2$ is the 2-norm and $Dgree\left(v_j\right)$ is the degree of v_j. λ is a predefined parameter to balance the influence of the degree. τ is the threshold of the similarity. If the cosine similarity is larger than τ, it is said that the two nodes are similar, otherwise the two nodes are not similar.

After $\boldsymbol{W}$ is computed, the reconstruction weight is also determined. For graph embedding, it is expected that the reconstruction error is as small as possible and the low dimensional embedding vectors are low rank. Therefore the optimization problem is expressed as follow:

$$\begin{aligned} &\min_{\boldsymbol{Y}} tr\left(\boldsymbol{Y}^T\boldsymbol{L}\boldsymbol{Y}\right) + \frac{\alpha}{2}\|\boldsymbol{X}\|_F^2 + \beta\|\boldsymbol{Y}\|_* + \gamma\|\boldsymbol{E}\|_{2,1} \\ &s.t.\ \boldsymbol{M} = \boldsymbol{X}\boldsymbol{Y} + \boldsymbol{E} \end{aligned} \tag{4}$$

where $tr(\cdot)$ is the trace of a matrix, $\boldsymbol{Y}^T$ is seen as the context embedding vectors and $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ is the node embedding vectors, $\boldsymbol{E} \in \mathbb{R}^{n\times n}$ can be seen as the noise, $\|\cdot\|_F$ is Frobenius norm and $\|\cdot\|_{2,1}$ is $\ell_{2,1}$-norm. $\boldsymbol{L} = \boldsymbol{D}' - \boldsymbol{W}$, $\boldsymbol{D}'$ is a diagonal matrix and $D'_{ii} = \sum_{j=1}^n W_{ji}$. α, β and γ are parameters. Eq. (4) can be solved by Augmented Lagrange Method [27,28] and is translated into Eq. (5).

$$\begin{aligned} &\min_{\boldsymbol{Y}} tr\left(\boldsymbol{Y}^T\boldsymbol{L}\boldsymbol{Y}\right) + \frac{\alpha}{2}\|\boldsymbol{X}\|_F^2 + \beta\|\boldsymbol{J}\|_* + \gamma\|\boldsymbol{E}\|_{2,1} \\ &s.t.\ \boldsymbol{M} = \boldsymbol{X}\boldsymbol{Y} + \boldsymbol{E}, \boldsymbol{Y} = \boldsymbol{J} \end{aligned} \tag{5}$$

The following Lagrangian function can be constructed from Eq. (5):

$$\begin{aligned} L =& tr\left(\boldsymbol{Y}^T\boldsymbol{L}\boldsymbol{Y}\right) + \frac{\alpha}{2}\|\boldsymbol{X}\|_F^2 + \beta\|\boldsymbol{J}\|_* + \gamma\|\boldsymbol{E}\|_{2,1} \\ &+ tr\left(\boldsymbol{\eta}\left(\boldsymbol{M} - \boldsymbol{X}\boldsymbol{Y} - \boldsymbol{E}\right)\right) + tr\left(\boldsymbol{\nu}\left(\boldsymbol{Y} - \boldsymbol{J}\right)\right) \\ &+ \frac{\mu}{2}\left(\|\boldsymbol{M} - \boldsymbol{X}\boldsymbol{Y} - \boldsymbol{E}\|_F^2 + \|\boldsymbol{Y} - \boldsymbol{J}\|_F^2\right) \end{aligned} \tag{6}$$

where $\boldsymbol{\eta}$, $\boldsymbol{\nu}$ and μ are the Lagrange multipliers.

When the other variables are fixed, $\boldsymbol{J}$ is updated as

$$\begin{aligned} \boldsymbol{J} &= \arg\min_{\boldsymbol{J}} = \beta\|\boldsymbol{J}\|_* - tr\left(\boldsymbol{\nu}\boldsymbol{J}\right) + \frac{\mu}{2}\|\boldsymbol{Y} - \boldsymbol{J}\|_F^2 \\ &= \arg\min_{\boldsymbol{J}} \frac{\beta}{\mu}\|\boldsymbol{J}\|_* + \frac{1}{2}\left\|\boldsymbol{J} - \boldsymbol{Y} - \frac{\boldsymbol{\nu}^T}{\mu}\right\|_F^2 \end{aligned} \tag{7}$$

The problem solution of Eq. (7) is obtained as the method in the Ref. [28,29].

When the other variables are fixed, $\frac{\partial L}{\partial \boldsymbol{X}}$ is computed as

$$\frac{\partial L}{\partial \boldsymbol{X}} = \alpha\boldsymbol{X} - \boldsymbol{\eta}^T\boldsymbol{Y}^T - \mu\left(\boldsymbol{M} - \boldsymbol{X}\boldsymbol{Y} - \boldsymbol{E}\right)\boldsymbol{Y}^T = 0 \tag{8}$$

Therefore $\boldsymbol{X}$ is solved as follow:

$$\boldsymbol{X} = \left(\boldsymbol{\eta}^T\boldsymbol{Y}^T + \mu\boldsymbol{M}\boldsymbol{Y}^T - \mu\boldsymbol{E}\boldsymbol{Y}^T\right)\left(\alpha\boldsymbol{I} - \mu\boldsymbol{Y}\boldsymbol{Y}^T\right)^{-1} \tag{9}$$

When the other variables are fixed, $\frac{\partial L}{\partial \boldsymbol{Y}}$ is updated as

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{Y}} &= \boldsymbol{Y}\boldsymbol{L}^T + \boldsymbol{Y}\boldsymbol{L} - \boldsymbol{X}^T\boldsymbol{\eta}^T + \boldsymbol{\nu}^T + \mu\boldsymbol{Y}^T - \mu\boldsymbol{J} \\ &+ \mu\boldsymbol{X}^T\left(\boldsymbol{M} - \boldsymbol{X}\boldsymbol{Y} - \boldsymbol{E}\right) = 0 \end{aligned} \tag{10}$$

Therefore $\boldsymbol{Y}$ is solved as follow:

$$\boldsymbol{Y} = \left(\mu\boldsymbol{I} + \mu\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\left(\boldsymbol{X}^T\boldsymbol{\eta}^T - \boldsymbol{\nu}^T + \mu\boldsymbol{J} + \boldsymbol{Q}\right) \tag{11}$$

where $\boldsymbol{Q} = \mu\boldsymbol{X}^T\boldsymbol{M} - \mu\boldsymbol{X}^T\boldsymbol{E} - \boldsymbol{Y}\boldsymbol{L} - \boldsymbol{Y}\boldsymbol{L}^T$.

When the other variables are fixed, $\boldsymbol{E}$ is updated as

$$\begin{aligned} \boldsymbol{E} &= \arg\min_{\boldsymbol{E}} \gamma\|\boldsymbol{E}\|_{2,1} - tr\left(\boldsymbol{\eta}\boldsymbol{E}\right) + \frac{\mu}{2}\|\boldsymbol{M} - \boldsymbol{X}\boldsymbol{Y} - \boldsymbol{E}\|_F^2 \\ &= \arg\min_{\boldsymbol{E}} \frac{\gamma}{\mu}\|\boldsymbol{E}\|_{2,1} + \frac{1}{2}\left\|\boldsymbol{E} + \boldsymbol{X}\boldsymbol{Y} - \boldsymbol{M} - \frac{\boldsymbol{\eta}^T}{\mu}\right\|_F^2 \end{aligned} \tag{12}$$

The problem solution of Eq. (12) is also obtained as the method in the Ref. [28]. The Lagrange multipliers $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$ are updated as

$$\begin{aligned} &\boldsymbol{\eta} \leftarrow \boldsymbol{\eta} + \mu\left(\boldsymbol{M} - \boldsymbol{X}\boldsymbol{Y} - \boldsymbol{E}\right)^T \\ &\boldsymbol{\nu} \leftarrow \boldsymbol{\nu} + \mu\left(\boldsymbol{Y} - \boldsymbol{J}\right)^T \end{aligned} \tag{13}$$

The variable μ is updated as the method in the Ref. [28].

The variables of Eq. (5) are iteratively updated as the above steps until the objective function is convergent. Therefore the detail steps of manifold graph embedding with low rank decomposition are summarized as Algorithm 1.

Algorithm 1

Input: A graph $G = \langle V, E\rangle$; α, β and γ.
Output: The low dimensional embedding $\boldsymbol{X}$ and $\boldsymbol{Y}$.

```
1:  Initialize $X, Y, J, E, \eta$ and $\nu$;
2:  Compute $M$ as Eq. (1);
3:  Compute $W$ as Eq. (3);
4:  while The objective function is not convergent do
5:      Update $J$ as Eq. (7);
6:      Update $X$ as Eq. (9);
7:      Update $Y$ as Eq. (11);
8:      Update $E$ as Eq. (12);
9:      Update $\eta$ and $\nu$ as Eq. (13);
10:     Update $\mu$;
11: end while
```

In Algorithm 1, $\boldsymbol{M}$ includes the global and local information of graph. Therefore the low dimensional embedding resulting from $\boldsymbol{M}$ is more appropriate than the result by directly using the adjacency matrix. In the objective optimization function (Eq. (4)), the manifold regularization and the low rank learning are introduced which result in the good generalization ability and the robustness.

3.2. Structure information propagation

From Eq. (3), it is known that $\boldsymbol{W}$ is not symmetrical. It means the neighborhood relation of nodes is also not symmetrical. In Algorithm 1, we can obtain the context embedding vectors $\boldsymbol{Y}$. When computing $\boldsymbol{W}$, only the neighborhood relation is considered and $\boldsymbol{Y}$ is ignored. After we obtain the embedding vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$, we can use $\boldsymbol{Y}$ to update $\boldsymbol{X}$. Once $\boldsymbol{X}_i$ is updated, the other embedding vectors are updated according to $\boldsymbol{X}_i$. Therefore the structure information is propagated in the whole network. For $\forall v_i, v_j \in V$, the new weight based on $\boldsymbol{X}$ and $\boldsymbol{Y}$ is as

$$W'_{ij} = \begin{cases} \dfrac{1}{1+\exp\left(-\boldsymbol{X}_{i,:}\cdot\boldsymbol{Y}_{:,j}\right)} & W_{ij} > 0 \\ 0 & W_{ij} = 0 \end{cases} \tag{14}$$

Therefore for each node $v_i \in V$ $(i = 1, 2, \ldots, n)$, the low dimensional embedding vector $\boldsymbol{X}_i$ is updated as

$$\boldsymbol{X}_i \leftarrow \sum_{j=1}^{n} W'_{ij}\boldsymbol{X}_j \tag{15}$$

We can iteratively update the low dimensional embedding vectors $\boldsymbol{X}$ until it is convergent. The structure information propagation steps are summarized as Algorithm 2.

Algorithm 2

```
Input: The low dimensional embedding vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$.
Output: The updated vectors $\boldsymbol{X}$.
1: while $\boldsymbol{X}$ is not convergent do
2:     Compute the weight $\boldsymbol{W}'$;
3:     for $\forall v_i \in V$ do
4:         Update $\boldsymbol{X}_i$ as Eq. (15);
5:     end for
6: end while
```

In Algorithm 2, both the node embedding vectors and the context embedding vectors are employed to update $\boldsymbol{X}$. After $\boldsymbol{X}$ is completely updated, we can also fuse $\boldsymbol{X}$ with the other embedding results such as random projection and then k-means algorithm is used to obtain the clusters of graph. The diagrammatic sketch of MGE algorithm is showed as Fig. 2.

3.3. Time complexity analysis

In MGE algorithm, the time complexity of computing $\boldsymbol{M}$ is $\mathcal{O}\left(n^3 r\right)$; it takes $\mathcal{O}\left(n^2\right)$ to compute $\boldsymbol{W}$. Let t be the iteration number of Algorithm 1, the time complexity of updating $\boldsymbol{J}$ is $\mathcal{O}\left(n^2 dt\right)$; it costs $\mathcal{O}\left(n^2 dt + 2nd^2 t\right)$ to update $\boldsymbol{X}$; the time complexity of updating $\boldsymbol{Y}$ is $\mathcal{O}\left(nd^2 t\right)$; it takes $\mathcal{O}\left(n^2 dt\right)$ to update $\boldsymbol{E}$; the time complexity of Eq. (13) is $\mathcal{O}\left(n^2 dt\right)$. Let t_1 be the iteration number of Algorithm 2. The time complexity of Algorithm 2 is $\mathcal{O}\left(n^2 t_1\right)$. Let t_2 be the iteration number of k-means algorithm. The time complexity of k-means algorithm is $\mathcal{O}\left(nkdt_2\right)$ where k is the number of clusters. Therefore the time complexity of MGE algorithm is $\mathcal{O}\left(n^3 r + n^2 dt + nd^2 t + n^2 t_1 + nkdt_2\right)$. It is known that $n^2 dt \geq nd^2 t$ because of $n \geq d$ and r, t, t_1, t_2, k, d are the predefined constants. Therefore the final time complexity of MGE algorithm is $\mathcal{O}\left(n^3\right)$.

4. Experimental results

In order to test the effectiveness of MGE algorithm, we choose DeepWalk [5], node2vec [6], M-NMF [14], AROPE [13], LLE [30] and LE [31] as the comparison algorithms. All algorithms are conducted on an Ubuntu serve with Intel Core i9-7900X CPU and 56G RAM. MGE algorithm is implemented by python 3.7. For MGE algorithm, $r = 4, \lambda \in [0.0001, 3], \tau \in [0.1, 0.7], \alpha = 1.8, \beta = 0.8, \gamma = 0.8$. The dimension of the embedding vectors d is set to 64.

Table 1
The details of the experimental data sets.

Data set	# Nodes	# Edges	# Communities	Density
Wisconsin	265	530	5	2.0000
Football	115	613	12	5.3304
Email	1005	25 571	42	25.4438
Politics	105	441	3	4.2000
Adjnoun	112	425	2	3.7946
PPI	3890	38 739	50	9.9586
Blogcatalog	10 312	667 966	39	64.7756

Density = # Edges/# Nodes.

4.1. Experimental data sets

In the experiments, we choose the following data sets as the experimental data sets.[1] [2] The details of the data sets are showed as Table 1.

4.2. Evaluation criteria

In order to evaluate the effectiveness of the algorithms, we choose Adjusted Rand Index (ARI), Fowlkes and Mallows Index (FM), Normalized Mutual Information (NMI), Modularity, and Purity as the evaluation criteria. The descriptions of ARI, FM and NMI can be seen from the site.[3] Modularity and Purity are defined as follow:

1. Modularity [32]:

$$\text{Modularity} = \frac{1}{2m}\sum_{i,j,i\neq j}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\cdot\delta_{i,j} \tag{16}$$

where m is the number of edges, k_i is the degree of the vertex v_i and k_j is the degree of the vertex v_j. If v_i and v_j are in the same community, $\delta_{i,j} = 1$; otherwise, $\delta_{i,j} = 0$.

2. Purity:

$$\text{Purity} = \frac{1}{n}\sum_{i=1}^{k}\max\left(\left|\boldsymbol{C}_i \cap \boldsymbol{L}_j\right|\right) \quad (j = 1, 2, \ldots, N) \tag{17}$$

where the final clustering result is $\boldsymbol{C} = \{\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_k\}$, $\boldsymbol{L} = \{\boldsymbol{L}_1, \boldsymbol{L}_2, \ldots, \boldsymbol{L}_N\}$ is the class label of the data set and N is the number of real classes.

4.3. Results and analysis

In order to test the performance of MGE algorithm and the comparison algorithms, we execute the algorithms on the experimental data sets. The test results are showed as Tables 2–6 and Figs. 3–4.

Tables 2–6 show the results of MGE algorithm and the comparison algorithms on the experimental data sets. In MGE algorithm, $r = 4, \alpha = 1.8, \beta = 0.8, \gamma = 0.8, \lambda = 0.001, \tau = 0.5$. On wisconsin and adjnoun data sets, the evaluation criteria of MGE algorithm get the best results except for FM index. On politics and football data sets, the advantages of MGE algorithm are obvious and the most evaluation criteria of MGE algorithm

[1] http://snap.stanford.edu/data/index.html.
[2] http://www-personal.umich.edu/~mejn/netdata/.
[3] https://scikitlearn.org/stable/modules/clustering.html.

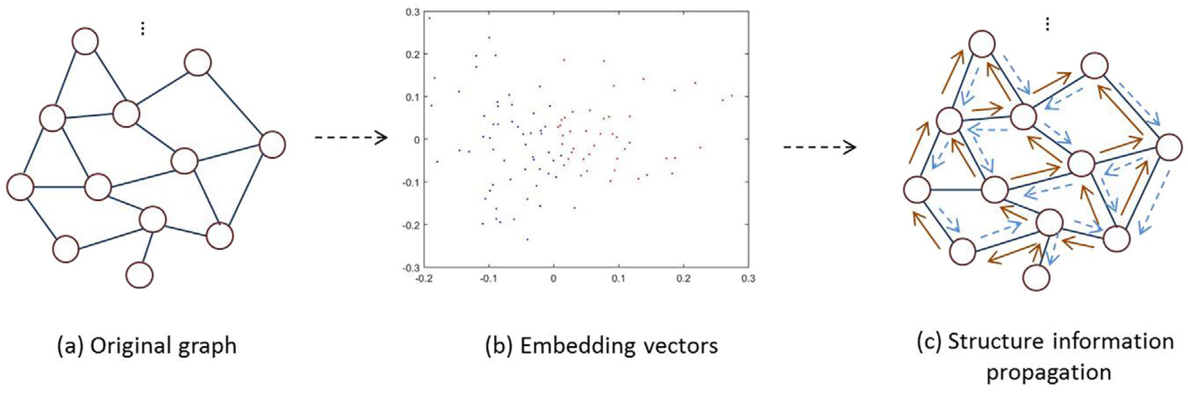

Fig. 2. The diagrammatic sketch of MGE algorithm.

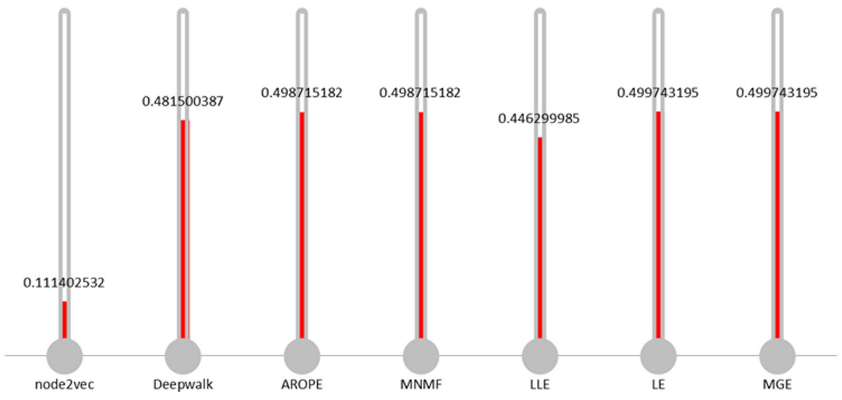

Fig. 3. The modularity of the algorithms on PPI data set.

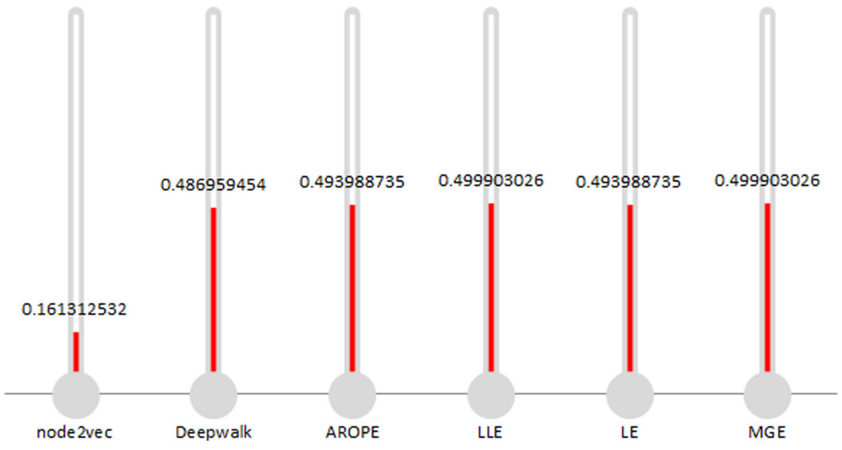

Fig. 4. The modularity of the algorithms on blogcatalog data set. *MNMF algorithm cannot get results in 100 h.

Table 2
The results of the algorithms on wisconsin data set.

	ARI	NMI	FM	Purity	Modularity
node2vec	0.0523	0.0568	0.3641	0.4717	0.1596
Deepwalk	0.0534	0.0695	0.3239	0.4943	**0.4962**
AROPE	0.0257	0.0401	0.3413	0.4679	0.4850
MNMF	0.0078	0.0388	0.2600	0.4679	0.4850
LLE	−0.0205	0.0606	0.5067	0.4679	**0.4962**
LE	0.0052	0.0356	**0.5474**	0.4642	**0.4962**
MGE	**0.0997**	**0.0773**	0.4264	**0.5094**	**0.4962**

are much better than the comparison algorithms. On email data set, Deepwalk algorithm gets the best result and MGE algorithm is the second best. For Deepwalk algorithm, it is a deep learning algorithm; email data set is a dense network and the amount of data can meet the training requirement of deep learning, therefore Deepwalk algorithm can get better result. Figs. 3–4 show the modularity results of the algorithms. Modularity can reflect the strength of network community structure. From the results, it is known that MGE algorithm and LE algorithm get the best results on PPI data set; MGE algorithm and LLE algorithm get the best results on blogcatalog data set. Therefore we can conclude that MGE algorithm is an effective algorithm for network embedding.

Table 3
The results of the algorithms on football data set.

	ARI	NMI	FM	Purity	Modularity
node2vec	0.0225	0.2877	0.1017	0.2870	0.1748
Deepwalk	0.5859	0.7607	0.6191	0.7652	**0.4913**
AROPE	–	–	–	–	–
MNMF	0.2262	0.4653	0.3649	0.4087	0.3973
LLE	0.2371	0.4774	0.3459	0.4523	0.3973
LE	0.1305	0.3938	0.2031	0.4523	**0.4913**
MGE	**0.7540**	**0.8754**	**0.7850**	**0.8261**	**0.4913**

— means AROPE algorithm cannot get results in 100 h.

Table 4
The results of the algorithms on email data set.

	ARI	NMI	FM	Purity	Modularity
node2vec	0.0006	0.1967	0.0388	0.1612	0.2328
Deepwalk	**0.4623**	**0.6815**	**0.4890**	**0.6557**	**0.4990**
AROPE	−0.0009	0.1585	0.0448	0.1522	0.4682
MNMF	0.0848	0.3457	0.1210	0.3085	0.4682
LLE	0.1213	0.5487	0.1845	0.5303	0.4682
LE	0.0005	0.2328	0.1786	0.1920	**0.4990**
MGE	0.3107	0.6023	0.4034	0.5045	**0.4990**

Table 5
The results of the algorithms on politics data set.

	ARI	NMI	FM	Purity	Modularity
node2vec	0.0035	0.0133	0.3986	0.4762	0.3160
Deepwalk	0.5095	0.4274	0.6957	0.7905	**0.4905**
AROPE	–	–	–	–	–
MNMF	0.4622	0.3858	0.6676	0.8095	**0.4905**
LLE	−0.0198	0.1098	0.5714	0.4857	**0.4905**
LE	0.0537	0.1312	0.6250	0.5048	**0.4905**
MGE	**0.6745**	**0.5745**	**0.8061**	**0.8476**	0.4905

Table 6
The results of the algorithms on adjnoun data set.

	ARI	NMI	FM	Purity	Modularity
node2vec	−0.0001	0.0051	0.5240	0.5446	0.4253
Deepwalk	−0.0039	0.0035	0.4951	0.5357	**0.4911**
AROPE	–	–	–	–	–
MNMF	−0.0078	0.0009	0.4922	0.5179	**0.4911**
LLE	−0.0012	0.0315	0.6976	0.5179	**0.4911**
LE	0.0013	0.0349	**0.6985**	0.5268	**0.4911**
MGE	**0.0176**	**0.1070**	0.6740	**0.5714**	**0.4911**

In order to test the influence of the parameters on the performance of MGE algorithm. We choose wisconsin data set as the experimental data set. For MGE algorithm, $r = 4, \alpha = 1.8, \beta = 0.8, \gamma = 0.8, \lambda = 0.001, \tau = 0.5$. When a parameter is investigated, we set the parameter with different values and the values of the other parameter are fixed. The test results of MGE algorithm are showed as Figs. 5–10.

Figs. 5–10 show the test results of MGE algorithm with different parameter values on wisconsin data set. Eq. (5) shows that the parameters have an influence on the performance of MGE algorithm. From Fig. 5, it is known that r has an influence on the performance of MGE algorithm. When r is 4, the most evaluation criteria get the best results. Fig. 6 shows the influence of λ on the performance of MGE algorithm. A large λ value means the degree of node has a large influence on the weight of edge which is seen as Eq. (3). From Fig. 6, it is known that the most evaluation criteria get the best results when λ is 0.001. Fig. 7 is the result of MGE algorithm with different τ values. τ determines the threshold when the weight of edge is computed. It can be seen that the most evaluation criteria get the best results when τ is 0.1. Fig. 8 shows the result of MGE algorithm with different α values. α can affect the generalization ability of the algorithm. The most

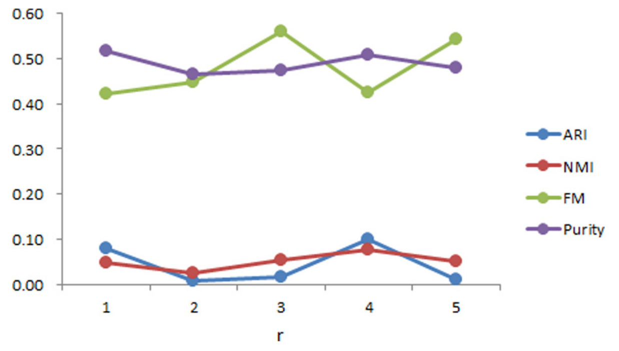

Fig. 5. The results of MGE algorithm with different r values on wisconsin data set.

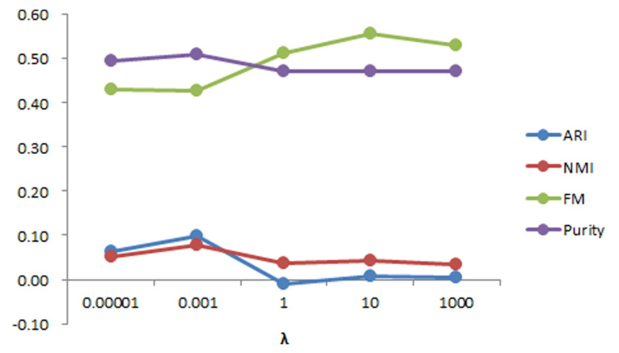

Fig. 6. The results of MGE algorithm with different λ values on wisconsin data set.

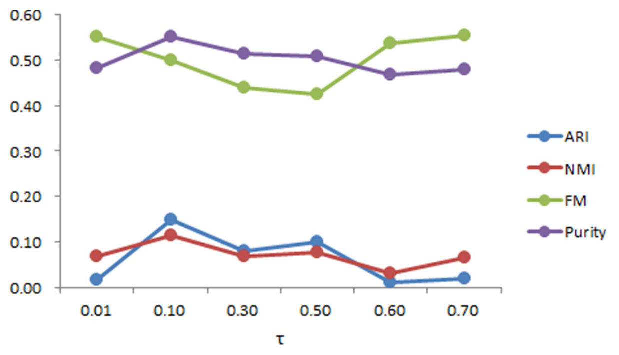

Fig. 7. The results of MGE algorithm with different τ values on wisconsin data set.

evaluation criteria get the best results when α is 1. Fig. 9 shows the result of MGE algorithm with different β values. β determines the robustness of the algorithm. It can be seen that the most evaluation criteria get the best results when β is 0.5. Fig. 10 is the result of MGE algorithm with different γ values. γ determines the sparsity of the solution. It can be seen that the most evaluation criteria get the best results when γ is 2.

In order to test the scalability of MGE algorithm, we conduct MGE algorithm on different data sets. For MGE algorithm, $r = 4, \alpha = 1.8, \beta = 0.8, \gamma = 0.8, \lambda = 0.001, \tau = 0.5$. The time cost is showed as Table 7 and Fig. 11.

Table 7 shows the time cost of MGE algorithm on different data sets. When the size of social network is not very large, the time cost is also not very large. The time cost is also large when the size of social network is large. From the results, it is known that the time cost of MGE algorithm increases with the increase of nodes' number and the increase trend of the time cost is not

Table 7
The results of the algorithms on adjnoun data set.

	Wisconsin	Football	Email	Politics	Adjnoun	PPI	Blogcatalog
Time cost	4.5409	4.2960	526.7925	2.6840	2.7322	11 942.0463	199 065.1719
# Nodes	265	115	1005	105	112	3890	10312

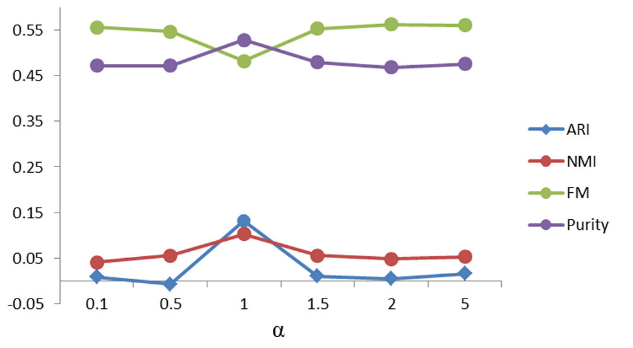

Fig. 8. The results of MGE algorithm with different α values on wisconsin data set.

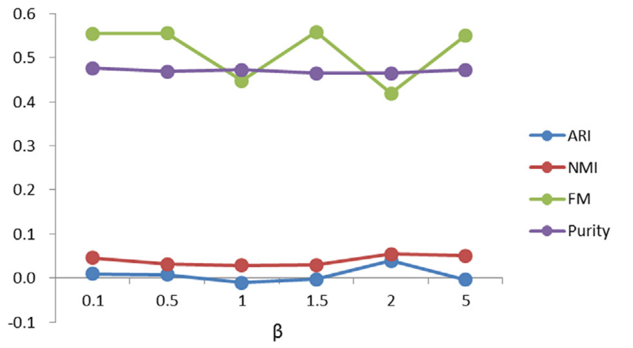

Fig. 9. The results of MGE algorithm with different β values on wisconsin data set.

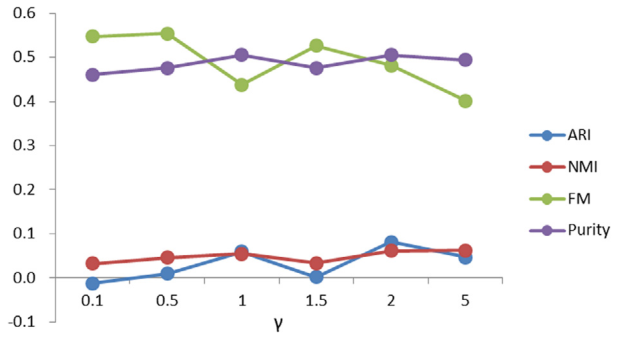

Fig. 10. The results of MGE algorithm with different γ values on wisconsin data set.

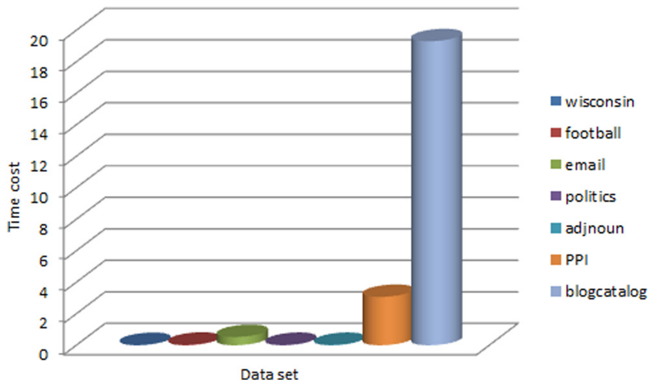

Fig. 11. The average time cost of MGE algorithm on different data sets.

linear. Fig. 11 shows the average time cost of each node. From Fig. 11, it can be seen that the average time cost of each node on blogcatalog data set is much more than the other data sets. It means MGE algorithm is sensitive to the size of social network which is consistent with the time complexity analysis and the result of Table 7.

5. Conclusions

In this paper, a manifold graph embedding algorithm with structure information propagation for community discovery is proposed. The proposed algorithm considers the asymmetry of edge in social network to construct high order proximity approximation matrix. Then manifold embedding and low rank decomposition are introduced to obtain the embedding vectors and the embedding vectors can be updated according to the structural information. The proposed algorithm and comparison algorithms are conducted on the experimental data sets. The experimental results show that the proposed algorithm is an effective algorithm. From the experimental results, it can be seen that the time cost of the proposed is high. Therefore we will introduce distributed computing or GPU acceleration to accelerate the proposed algorithm in the future.

CRediT authorship contribution statement

Shuliang Xu: Conceptualization, Methodology, Writing - original draft, Visualization, Investigation, Writing - review & editing. **Shenglan Liu:** Data curation, Investigation. **Lin Feng:** Supervision, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by National Natural Science Fund of China (Nos.61972064, 61672130), Liaoning Revitalization Talents Program, China (No.XLY-C1806006), Fundamental Research Funds for the Central Universities, China (No.DUT19RC(3)012). The authors are grateful to the editor and the anonymous reviewers for the constructive comments that helped to improve the quality and presentation of this paper.

References

[1] H. Cai, V.W. Zheng, K.C.-C. Chang, A comprehensive survey of graph embedding: Problems, techniques, and applications, IEEE Trans. Knowl. Data Eng. 30 (9) (2018) 1616–1637.
[2] P. Cui, X. Wang, J. Pei, W. Zhu, A survey on network embedding, IEEE Trans. Knowl. Data Eng. 31 (5) (2018) 833–852.
[3] P. Goyal, E. Ferrara, Graph embedding techniques, applications, and performance: A survey, Knowl.-Based Syst. 151 (2018) 78–94.

[4] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, 2013, arXiv preprint arXiv:1301.3781.

[5] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 701–710.

[6] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 855–864.

[7] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in: Proceedings of the 21th International Conference on World Wide Web, 2015, pp. 1067–1077.

[8] A. Narayanan, M. Chandramohan, L. Chen, Y. Liu, S. Saminathan, Subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs, 2016, arXiv preprint arXiv:1606.08928.

[9] L.F. Ribeiro, P.H. Saverese, D.R. Figueiredo, struc2vec: Learning node representations from structural identity, in: Proceedings of the 23th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 385–394.

[10] Y. Dong, N.V. Chawla, A. Swami, metapath2vec: Scalable representation learning for heterogeneous networks, in: Proceedings of the 23th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 135–144.

[11] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: Proceedings of the 22th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1225–1234.

[12] Y. Pei, N. Chakraborty, K. Sycara, Nonnegative matrix tri-factorization with graph regularization for community detection in social networks, in: The 24th International Joint Conference on Artificial Intelligence, 2015, pp. 2083–2089.

[13] Z. Zhang, P. Cui, X. Wang, J. Pei, X. Yao, W. Zhu, Arbitrary-order proximity preserved network embedding, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018, pp. 2778–2786.

[14] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, S. Yang, Community preserving network embedding, in: The 31th AAAI Conference on Artificial Intelligence, 2017, pp. 203–209.

[15] J. Qiu, Y. Dong, H. Ma, J. Li, C. Wang, K. Wang, J. Tang, Netsmf: Large-scale network embedding as sparse matrix factorization, in: Proceedings of the 25th International Conference on World Wide Web, 2019, pp. 1509–1520.

[16] Y. Li, Y. Wang, T. Zhang, J. Zhang, Y. Chang, Learning network embedding with community structural information, in: The 28th International Joint Conference on Artificial Intelligence, 2019, pp. 2937–2943.

[17] J. Zhang, Y. Dong, Y. Wang, J. Tang, M. Ding, Prone: fast and scalable network representation learning, in: The 28th International Joint Conference on Artificial Intelligence, 2019, pp. 4278–4284.

[18] P. Veličković, W. Fedus, W.L. Hamilton, P. Liò, Y. Bengio, R.D. Hjelm, Deep graph infomax, in: The 7th International Conference on Learning Representations, 2019.

[19] S. Zhang, H. Yin, Q. Wang, T. Chen, H. Chen, Q.V.H. Nguyen, Inferring substitutable products with deep network embedding, in: The 28th International Joint Conference on Artificial Intelligence, 2019, pp. 4306–4312.

[20] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, C. Zhang, Attributed graph clustering: A deep attentional embedding approach, in: The 28th International Joint Conference on Artificial Intelligence, 2019, pp. 3670–3676.

[21] Y. Guo, G. Ding, J. Han, H. Shao, X. Lou, Q. Dai, Zero-shot learning with many classes by high-rank deep embedding networks, in: The 28th International Joint Conference on Artificial Intelligence, 2019, pp. 2428–2434.

[22] C. Yang, M. Sun, Z. Liu, C. Tu, Fast network embedding enhancement via high order proximity approximation, in: The 26th International Joint Conference on Artificial Intelligence, 2017, pp. 3894–3900.

[23] H. Chen, S.F. Sultan, Y. Tian, M. Chen, S. Skiena, Fast and accurate network embeddings via very sparse random projection, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 399–408.

[24] Q. Long, Y. Wang, L. Du, G. Song, Y. Jin, W. Lin, Hierarchical community structure preserving network embedding: A subspace approach, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 409–418.

[25] J. Wu, J. He, Scalable manifold-regularized attributed network embedding via maximum mean discrepancy, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 2101–2104.

[26] S. Hasan, E. Curry, Word re-embedding via manifold dimensionality retention, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 321–326.

[27] X. Zhang, Matrix Analysis and Applications, Tsinghua University Press, 2014.

[28] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, Y. Ma, Robust recovery of subspace structures by low-rank representation, IEEE Trans. Pattern Anal. Mach. Intell. 35 (1) (2012) 171–184.

[29] G. Liu, Z. Lin, Y. Yu, Robust subspace segmentation by low-rank representation, in: Proceedings of the 27th International Conference on Machine Learning, 2010, pp. 663–670.

[30] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.

[31] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, in: Advances in Neural Information Processing Systems, 2002, pp. 585–591.

[32] M.E. Newman, Fast algorithm for detecting community structure in networks, Phys. Rev. E 69 (6) (2004) 066133.