Proximity Preserving Nonnegative Matrix Factorization

Yuya Ogawa[1], Koh Takeuchi[2], Yuya Sasaki[1], and Makoto Onizuka[1]

[1] Graduate School of Information Science of Technology, Osaka University, 1-5 Yamadaoka, Suita, Osaka, Japan
{ogawa.yuya,sasaki,onizuka}@ist.osaka-u.ac.jp

[2] NTT Communication Science Laboratories, 2-4 Hikaridai, Seika, Soraku, Kyoto, Japan
koh.t@acm.org

Abstract. We consider the problem of community detection. Although network embedding and representation learning methods are recently getting popular, we claim that they fall into suboptimal solutions for community detection, because they are based on indirect approach, which requires to apply clustering methods such as k-means to the embedding/representation vectors. We present PPNMF, proximity preserving nonnegative matrix factorization for community detection. The idea of PPNMF is three-hold. 1) PPNMF is based on direct approach: it directly minimizes its loss function for community detection. 2) Users can control the importance of observed edges over unobserved edges. 3) PPNMF can precisely capture the effects of the first-order and second-order proximities of vertexes to communities. Also, PPNMF employs the Adamic Adar index as the second-order proximity. The experiments validate that PPNMF performs better or comparable to existing methods in various real datasets for the tasks of community detection.

Keywords: community detection · NMF · Network analysis

1 Introduction

Graph is a fundamental data structure consisting of vertexes and their relationships. Graphs and network are ubiquitous in many application domains, such as web graphs [8], social networks [9], and paper citation networks [17]. Of particular interest, community detection is one of the most widely used techniques for graph processing. A recent paper on the statistics of graph processing usage [25] reports "Clustering is the most popular computation performed, while community detection is the most popular problem solved using machine learning".

There are many recent work for community detection methods [13, 33, 27, 32] and network embedding and representation learning methods [22, 10, 28–30]. The community detection methods directly obtain communities by minimizing their own loss functions for the purpose of community detection. So, they effectively capture mesoscopic structure of networks. In contrast, the network embedding and representation learning methods are also used for community detection, but in indirect way. That is, they first obtain embedding/representation vectors of vertexes and, then they apply traditional clustering methods (such as k-means) to the vectors. Since the network embedding and representation learning methods are designed to capture microscopic structure of networks, they are not suitable for community detection. In addition, network embedding methods obtain distributed representations of vertexes by minimizing their loss functions regardless of the community structure either linear or non-linear, so the distributed representations may not be suitable for k-means. Therefore, we claim that they fall into suboptimal solutions for community detection (we validate this conjecture in our experiments). These investigations suggest that we should take a direct approach for effective community detection.

As for the existing direct methods, considerable improvements have been made. Most of them are based on NMF, however there are still two fundamental problems. First, they treat observed edges (non-zero elements in adjacency matrix) and unobserved edges (zero elements) with equal importance. However, due to the sparsity of networks, the number of the observed edges is much smaller than the number of unobserved edges. So, the result of NMF is largely affected more by unobserved edges than observed edges. This observation implies that we should put larger importance on observed edges over unobserved edges, in particular for sparse networks. Second, there are many community detection methods that take either the first-order proximity of vertexes (such as modularity-based methods [18, 6]) or the second-order proximity (such as SCAN [31]) into account, however, they do not use both proximities at the same time. This observation implies that we should capture the both of their effects independently.

We present PPNMF, proximity preserving nonnegative matrix factorization for community detection. The idea of PPNMF is as follows. 1) PPNMF directly minimizes its loss function for community detection. It does not require additional step such as applying k-means clustering. 2) Users can control the importance of observed edges over unobserved edges, so that we can detect community structure with higher quality. 3) PPNMF uses the first-order proximity and the second-order proximity independently in its loss function, so that it preserves both of their effects in community structure. Also, PPNMF employs the Adamic Adar index [1] as for the second-order proximity, because it usually provides higher performance compared to other proximities [16, 21]. Although the second and the third ideas above is gradually getting popular in indirect approaches, there is no work on direct approaches that combine those ideas, as well as employing the Adamic Adar index at the same time. We extensively made experiments for various community detection and network embedding methods on various real datasets. We confirm that PPNMF performs better than or comparable to the existing methods for the tasks of community detection. We also confirm that our method is stable against the parameter selection.

The rest of this paper is organized as follows. We present our method in Section 2. Section 3 gives the purpose and results of the evaluations. Section 4 gives the details of the related work. Section 5 concludes this paper.

2 Proximity Preserving Nonnegative Matrix Factorization for Community Detection

In this section, we present our method, PPNMF (Proximity Preserving Nonnegative Matrix Factorization). PPNMF is designed as follows.

- PPNMF is based on direct approach for effective community detection. It directly obtains a community assignment representation of vertexes by minimizing the loss function for community detection extended from SymNMF.
- PPNMF permits users to control the loss weight between observed edges and unobserved edges in the loss function.
- PPNMF preserves both of the first-order and second-order proximities in the community assignment representation of vertexes. That is, PPNMF takes those proximities into its loss function independently.

First, we introduce our notations and preliminaries. Then, we present the details of PPNMF.

2.1 Notations and preliminaries

We define terms and notations in Table 1. We denote matrices by bold uppercase letters. $\mathbb{R}_+$ is the set of nonnegative real numbers.

Table 1: Terms and notations

Symbol	definition
N	number of vertexes
E	number of edges
K	number of communities
$C = \{c_0, c_1, ..., c_{K-1}\}$	communities
$\boldsymbol{A} \in \mathbb{R}_+^{N\times N}$	adjacency matrix
$\boldsymbol{W} \in \mathbb{R}_+^{N\times N}$	structural similarity matrix
$\boldsymbol{V} \in \mathbb{R}_+^{N\times K}$	community assignment matrix
$\boldsymbol{B} \in \mathbb{R}_+^{N\times N}$	loss weight matrix
$\boldsymbol{J} \in \mathbb{R}_+^{N\times N}$	matrix with all elements 1
$\boldsymbol{X}_{ij}$	(i,j)-th entry of $\boldsymbol{X}$
$\mathrm{Tr}(\boldsymbol{X})$	trace of $\boldsymbol{X}$
$\|\|\boldsymbol{X}\|\|_F$	Frobenius norm of $\boldsymbol{X}$
$\boldsymbol{x}_i$	i-th row vector of matrix $\boldsymbol{X}$
$\odot$	Hadamard product

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a given undirected and unweighted network $\mathcal{G}$[3], where $\mathcal{V}$ and $\mathcal{E}$ represent the vertex set and the edge set, respectively. We use two

[3] In this paper, we use only unweighted networks. However, our model can be extended to weighted networks.

matrices, adjacency matrix $\boldsymbol{A}$ and structural similarity matrix $\boldsymbol{W}$ to describe the network. An entry of the adjacency matrix $\boldsymbol{A}_{ij}$ represents the relationship between vertex i and j. We have $\boldsymbol{A}_{ij} = 1$ if there is an edge between vertex i and j, and $\boldsymbol{A}_{ij} = 0$ otherwise. An entry of the structural similarity matrix $\boldsymbol{W}_{ij}$ represents the neighborhood structural similarity between vertex i and j.

Generally, networks consist of multiple communities. Let $C = \{c_0, c_1, ..., c_{K-1}\}$ denote communities, where c_i represents i-th community. We assume that each vertex belongs to a single community. We introduce two proximities between vertexes before describing PPNMF model.

first-order proximity The first-order proximity represents the local pairwise proximity between two vertices. We use the adjacency matrix to express the proximity. The first-order proximity is the most fundamental information of vertex relationship. Two vertexes connected with an edge are likely to belong to the same community. So, the community structure of a network should preserve this proximity. Actually, many community detection methods [33, 32, 27] and network embedding methods preserve this proximity. However, due to the sparsity of the network, the first-order proximity is insufficient to detect communities. To resolve this issue, we employ second-order proximity as shown next.

second-order proximity The second-order proximity represents the similarity of the neighborhood structure between two vertices. Two vertexes that share many common vertexes are likely to belong to the same community even if the vertexes do not connect directly. Many existing methods, such as LINE [28] and MNMF [30], also preserve the second-order proximity. By combining both the first-order and the second-order proximities, we can detect communities more precisely and, in addition, the community detection result becomes more robust to the sparsity of networks.

We employ the Adamic Adar index [1] for the second-order proximity, because it achieves high performance in link prediction task [16]. We define the structural similarity matrix $\boldsymbol{W}$ by using the idea of Adamic Adar index as follows.

$$\boldsymbol{W}_{ij} = \sum_{u \in N(i) \cap N(j)} \frac{1}{\log_{10}(|N(u)|)} \tag{1}$$

$N(x)$ and $|N(x)|$ denote the neighbor vertexes of vertex x and its size, respectively.

2.2 PPNMF Model

Based on the above discussions, we present our PPNMF model. We can obtain a community assignment matrix $\boldsymbol{V}$ from a given observed adjacency matrix $\boldsymbol{A}$ by leveraging the idea of SymNMF [13]. SymNMF minimizes the following loss function.

$$L = ||\boldsymbol{A} - \boldsymbol{V}\boldsymbol{V}^T||_F^2 \ , \quad s.t. \boldsymbol{V} \geq 0 \tag{2}$$

$\boldsymbol{V}$ expresses which vertex belongs to which community. Each row and column of $\boldsymbol{V}$ corresponds to a vertex and community, respectively. The community assignment for each vertex i is decided by choosing the column with the largest value in row vector $\boldsymbol{v}_i$.

Due to the sparsity of networks, the number of observed edges (non-zero elements in $\boldsymbol{A}$) is much smaller than that of unobserved edges (zero elements), such as more than two or three orders of magnitude. Therefore, in the loss function, we should mitigate the error penalty for zero elements more than non-zero elements in $\boldsymbol{A}$ as follows.

$$L_{1st} = ||(\boldsymbol{A} - \boldsymbol{V}\boldsymbol{V}^T) \odot \boldsymbol{B}||_F^2 \ , \quad s.t.\ \boldsymbol{V} \geq 0 \tag{3}$$

We define the loss weight matrix $\boldsymbol{B} = \beta\boldsymbol{A} + (1-\beta)(\boldsymbol{J} - \boldsymbol{A})$, s.t $0.5 \leq \beta \leq 1.0$. β is a balancing parameter of the error penalty between zero elements and non-zero elements in $\boldsymbol{A}$. By introducing $\boldsymbol{B}$, we can obtain a more precise community assignment representation of vertexes for real world networks. $\beta = 0.5$ indicates imposing penalty equally to zero elements and non-zero elements: This is equivalent to a setting without using $\boldsymbol{B}$.

Next, a pair of vertexes that shares many common vertexes should have a similar community assignment representation. Accordingly, we exploit the second-order proximity of a given network by minimize the following loss function.

$$L_{2nd} = \sum_{ij} \boldsymbol{W}_{ij} ||\boldsymbol{v}_i - \boldsymbol{v}_j||^2 \ , \quad s.t.\ \boldsymbol{V} \geq 0 \tag{4}$$

Now, we combine Equation 3 and 4 to preserve both of the first-order and second-order proximities in our loss function. Since the first-order proximity is more fundamental information than the second-order proximity, we design the loss function to treat the first-order proximity as the primary target, while using the second-order proximity as the secondary. The joint loss function is defined as follows.

$$\begin{aligned} L_{PPNMF} &= L_{1st} + \lambda L_{2nd} \\ &= ||(\boldsymbol{A} - \boldsymbol{V}\boldsymbol{V}^T) \odot \boldsymbol{B}||_F^2 + \lambda \sum_{ij} \boldsymbol{W}_{ij} ||\boldsymbol{v}_i - \boldsymbol{v}_j||^2 \\ & s.t.\ \boldsymbol{V} \geq 0 \end{aligned} \tag{5}$$

λ is a balancing parameter between the first-order proximity and the second-order proximity.

Finally, let $\boldsymbol{V}$ be a community assignment matrix obtained by minimizing L_{PPNMF}, an reconstructed adjacency matrix $\hat{\boldsymbol{A}}$ can be generated as follows:

$$\hat{\boldsymbol{A}} = \boldsymbol{V}\boldsymbol{V}^T \tag{6}$$

2.3 Optimization

The loss function of Equation 5 is not convex in $\boldsymbol{V}$. Therefore, it is not realistic to find global minimum of the function. We present an iterative algorithm to achieve a local minimum.

Updating Rule Equation 4 can be rewritten as:

$$L_{2nd} = 2\mathrm{Tr}(\boldsymbol{V}^T \boldsymbol{L} \boldsymbol{V}) \tag{7}$$

$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$, which is called graph Laplacian. $\boldsymbol{D}$ is a diagonal matrix whose entries are column sum of $\boldsymbol{W}$, $\boldsymbol{D}_{jj} = \sum_l \boldsymbol{W}_{jl}$. Therefore, the loss function is rewritten to:

$$L_{PPNMF}(\boldsymbol{V}) = ||(\boldsymbol{A} - \boldsymbol{V}\boldsymbol{V}^T) \odot \boldsymbol{B}||_F^2 + 2\lambda \mathrm{Tr}(\boldsymbol{V}^T \boldsymbol{L} \boldsymbol{V}) \tag{8}$$

Let $\boldsymbol{\alpha}$ be the Lagrange multiplier for constraint $\boldsymbol{V}_{ij} \geq 0$, the partial derivatives of the Lagrange $\mathcal{L}(\boldsymbol{V}; \boldsymbol{\alpha})$ with $\boldsymbol{V}$ is

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{V}} = -4(\boldsymbol{A} \odot \boldsymbol{B}^2)\boldsymbol{V} + 4((\boldsymbol{V}\boldsymbol{V}^T) \odot \boldsymbol{B}^2)\boldsymbol{V} + 4\boldsymbol{L}\boldsymbol{V} + \boldsymbol{\alpha} \tag{9}$$

$$= -4(\boldsymbol{A} \odot \boldsymbol{B}^2)\boldsymbol{V} + 4((\boldsymbol{V}\boldsymbol{V}^T) \odot \boldsymbol{B}^2)\boldsymbol{V} + 4\boldsymbol{D}\boldsymbol{V} - 4\boldsymbol{W}\boldsymbol{V} + \boldsymbol{\alpha} \tag{10}$$

Using the KKT conditions $\boldsymbol{\alpha}_{ij}\boldsymbol{V}_{ij} = 0$ and $\frac{\partial \mathcal{L}}{\partial \boldsymbol{V}} = 0$, we obtain the following equation:

$$-((\boldsymbol{A} \odot \boldsymbol{B}^2)\boldsymbol{V} + \boldsymbol{W}\boldsymbol{V})_{ij}\boldsymbol{V}_{ij} + (((\boldsymbol{V}\boldsymbol{V}^T) \odot \boldsymbol{B}^2)\boldsymbol{V} + \boldsymbol{D}\boldsymbol{V})_{ij}\boldsymbol{V}_{ij} = 0 \tag{11}$$

Therefore, we get the following update rule:

$$\boldsymbol{V}_{ij} \leftarrow \boldsymbol{V}_{ij} \frac{((\boldsymbol{A} \odot \boldsymbol{B}^2)\boldsymbol{V} + \boldsymbol{W}\boldsymbol{V})_{ij}}{(((\boldsymbol{V}\boldsymbol{V}^T) \odot \boldsymbol{B}^2)\boldsymbol{V} + \boldsymbol{D}\boldsymbol{V})_{ij}} \tag{12}$$

Algorithm 1 shows the overall optimization process in PPNMF, which consists of two steps. In pre-training step, we first initialize the community assignment matrix $\boldsymbol{V}$ randomly (line 2 in Algorithm 1). To achieve a better local minimum, we pre-train $\boldsymbol{V}$ by SymNMF, which is a simple and effective method for community detection (line 3-6 in Algorithm 1). Then, in main training step, we update $\boldsymbol{V}$ according to Eq.(12) (line 8-11 in Algorithm 1). Finally, we obtain the matrix $\boldsymbol{V}$ (line 12 in Algorithm 1). After the optimization, $\boldsymbol{V}$ stores community assignment information. In the matrix, each column corresponds to a community. For each vertex, we choose the column index with the largest value in its representation as its assigned community. Note: max-iter indicates the number of the iterations we apply the update rule.

2.4 Computation complexity

The overall computation of PPNMF depends on the matrix multiplication in the updating rules. Since the computation of the matrix product $\boldsymbol{X}\boldsymbol{Y}$ ($\boldsymbol{X} \in \mathbb{R}_+^{x \times z}, \boldsymbol{Y} \in \mathbb{R}_+^{z \times y}$) is $O(xyz)$, the pre-training complexity and the main training complexity are $O((N^2K + NK^2)t)$ and $O((N^2K)t)$, respectively. t is the number of iterations. Since usually $K \leq N$, the overall computation complexity of our method is $O((N^2K)t)$. Observe that this complexity is the same as that of SymNMF, which is the base algorithm of our method.

Algorithm 1 Optimization algorithm of PPNMF

```
Require: Adjacency matrix $\boldsymbol{A}$, Structural similarity matrix $\boldsymbol{W}$, Loss weight matrix $\boldsymbol{B}$
Ensure: Community assignment matrix $\boldsymbol{V}$
 1: pre-training step
 2:     initialize randomly $\boldsymbol{V}$
 3:     training by SymNMF
 4:     for $i = 1$ to max-iter do
 5:         $\boldsymbol{V}_{ij} \leftarrow \boldsymbol{V}_{ij} \frac{(\boldsymbol{AV})_{ij}}{(\boldsymbol{VV}^T\boldsymbol{V})_{ij}}$
 6:     end for
 7: main training step
 8:     training by Eq. (12)
 9:     for $i = 1$ to max-iter do
10:         $\boldsymbol{V}_{ij} \leftarrow \boldsymbol{V}_{ij} \frac{((\boldsymbol{A} \odot \boldsymbol{B}^2)\boldsymbol{V} + \boldsymbol{WV})_{ij}}{(((\boldsymbol{VV}^T) \odot \boldsymbol{B}^2)\boldsymbol{V} + \boldsymbol{DV})_{ij}}$
11:     end for
12: return $\boldsymbol{V}$
```

3 Experiments

We perform experiments to compare the performance of our method PPNMF to those of state-of-the-art methods. We use six real network datasets with ground-truth communities. We design the experiments to answer the following questions:

Q1 Can PPNMF obtain better community assignment representation of vertexes? (Section 3.3)

Q2 Is the second-order proximity essential for community assignment representation of vertexes? Should we control the importance of observed edges over unobserved edges? (Section 3.4)

Q3 How largely the parameters affect the performance? (Section 3.5)

3.1 Datasets

We use six real networks, four social networks and two citation networks. We show the statistics of the datasets in Figure 2.

Table 2: Statistics of datasets: N is the number of vertexes. E is the number of edges. K is the number of ground-truth communities.

Dataset	N	E	K
parliament	451	5823	7
polblog	1490	16627	2
cora	2708	5278	7
citeseer	3312	4732	6
blogcatalog	5196	171743	7
flickr	7575	239738	9

- parliament : Parliament network is a social network in french parliament. The vertexes represent parliament members. They have an edge if they cosigned a bill together. The dataset is used in [3].
- polblog[4] : Polblog network is a social network. The vertexes and the edges represent blogs about US politics and web links, respectively.
- cora[5] : Cora network is a citation network on the cora. The vertexes and the edges represent the papers and their citations, respectively.
- citeseer[6] : Citeseer network is a citation network on the citeseer. The vertexes and the edges represent the papers and their citations, respectively.
- blogcatalog : Blogcatalog nework is a social network on blogctalog website. The vertexes and the edges represent the bloggers and their friendship, respectively. The dataset is used in [11].
- flickr : Flickr network is a social network on flickr website. The vertexes and the edges represent the photographers and their relationship, respectively. The dataset is used in [11].

3.2 Baseline Methods

To confirm that PPNMF can obtain better community assignment representation of vertexes, we choose four recent NMF-based methods and five state-of-the-art network embedding methods as baseline methods.

- SymNMF [13] : SymNMF is a fundamental method for graph clustering. SymNMF preserves only first-order proximity [7]. SymNMF can be seen as a special case of our method by setting $\beta = 0.5$ and $\lambda = 0$.
- NSED [27] : NSED is a nonnegative symmetric encoder-decoder model proposed for community detection. NSED preserves only first-order proximity.
- DANMF [32] : DANMF is a deep autoencoder-like NMF model proposed for community detection. The model adopts the deep learning to NMF. DANMF preserves only first-order proximity.
- DeepWalk [22] : DeepWalk adopts random walk and skip-gram to learn network embedding. DeepWalk preserves first-order, second-order, and higher-order proximities.
- node2vec [10] : node2vec is an extension of DeepWalk and it adopts biased random walk to generate vertex representation vectors. node2vec preserves first-order, second-order, and higher-order proximities.
- LINE [28] : LINE uses an objective function to preserve first-order proximity and second-order proximity separately.
- SDNE [29]: SDNE employs deep neural networks to compute vertex embeddings. SDNE preserves first-order and second-order proximities and also balances the error penalty between observed edges and unobserved edges.

[4] http://www-personal.umich.edu/mejn/netdata/
[5] https://linqs.soe.ucsc.edu/data
[6] https://linqs.soe.ucsc.edu/data
[7] According to SymNMF paper [13], SymNMF can use either the first-order proximity or the second-order proximity. We use the former one in this paper.

– MNMF [30] : MNMF designs its objective function to detect communities and generate vertex embeddings simultaneously. MNMF balances the effects of first-order and second-order proximities and modularity measure.

As for microbenchmark of PPNMF, we use PPNMF($\beta = 0.5$), SymNMF, and PPNMF(W). PPNMF($\beta = 0.5$) indicates a special case of PPNMF, which is equivalent to a case without using the loss weight matrix $\boldsymbol{B}$. We also use SymNMF, which is more special case of PPNMF($\beta = 0.5$) without using the second-order proximity. PPNMF(W) is a variation of PPNMF by exchanging $\boldsymbol{A}$ and $\boldsymbol{W}$ in the loss function, that is, PPNMF(W) treats the second-order proximity as the primary target, while using the first-order proximity as the secondary. PPNMF(W) is used to verify the appropriateness of the design of PPNMF.

Table 3: Performance evaluation based on ARI (bold numbers represent the best results. SymNMF, node2vec, and DeepWalk are abbreviated as SNMF, n2v, and DW, respectively.)

methods	PPNMF	DANMF	NSED	SNMF	DW	n2v	LINE	SDNE	MNMF
parliament	**0.942**	0.612	0.572	0.584	0.629	0.769	0.374	0.378	0.002
polblog	**0.621**	0.602	0.274	0.584	0.470	0.476	0.016	0.012	0.001
cora	0.366	0.280	0.176	0.243	0.270	**0.394**	0.056	0.044	0.008
citeseer	**0.197**	0.100	0.029	0.084	0.121	0.183	0.011	0.006	0.003
blogcatalog	**0.160**	0.142	0.071	0.132	0.102	0.130	0.149	0.035	0.002
flickr	**0.152**	0.054	0.031	0.101	0.101	0.111	0.103	0.012	0.000

Table 4: Performance evaluation based on NMI (bold numbers represent the best results. SymNMF, node2vec, and DeepWalk are abbreviated as SNMF, n2v, and DW, respectively.)

methods	PPNMF	DANMF	NSED	SNMF	DW	n2v	LINE	SDNE	MNMF
parliament	**0.900**	0.694	0.594	0.685	0.755	0.845	0.519	0.570	0.027
polblog	**0.522**	0.510	0.233	0.483	0.449	0.453	0.018	0.067	0.005
cora	0.446	0.371	0.325	0.345	0.376	**0.463**	0.090	0.093	0.015
citeseer	**0.236**	0.170	0.132	0.187	0.119	**0.236**	0.018	0.022	0.007
blogcatalog	**0.237**	0.232	0.111	0.220	0.193	0.222	0.230	0.120	0.005
flickr	**0.222**	0.111	0.048	0.171	0.163	0.172	0.168	0.028	0.003

3.3 Community detection

We evaluate the performance for community detection based on typical metrics, ARI, NMI, and Purity. On these metrics, larger value indicates better performance taking from 0 to 1. A detailed information is described in [5]. For the

Table 5: Performance evaluation based on Purity (bold numbers represent the best results. SymNMF, node2vec, and DeepWalk are abbreviated as SNMF, n2v, and DW, respectively.)

methods	PPNMF	DANMF	NSED	SNMF	DW	n2v	LINE	SDNE	MNMF
parliament	**0.967**	0.849	0.749	0.886	0.923	0.966	0.779	0.786	0.406
polblog	**0.894**	0.884	0.762	0.863	0.843	0.845	0.564	0.557	0.508
cora	0.633	0.578	0.483	0.562	0.559	**0.643**	0.379	0.375	0.302
citeseer	**0.501**	0.374	0.335	0.405	0.400	0.498	0.261	0.256	0.213
blogcatalog	**0.427**	0.394	0.326	0.380	0.380	0.392	0.399	0.297	0.199
flickr	**0.400**	0.252	0.208	0.343	0.323	0.342	0.348	0.161	0.130

NMF-based methods, we set the number of community K as the number of ground-truth communities. For the network embedding methods, we set the size of distributed representation of vertexes to be 64, and then apply the standard k-means algorithm to identify communities as described in [32]. We set the number of iteration in PPNMF to be 500. All experiments are conducted on a server with Tesla T4 GPU and 13GB RAM running Ubuntu 18.04. The parameters of all methods are searched to achieve the highest score of the metrics. The search space of each method is decided based on the description in its original paper. The parameters of PPNMF, β and λ, are tuned in the range of $\{0.6, 0.7, 0.8, 0.9, 0.99\}$ and $\{0.0001, 0.001, 0.01, 0.1, 0.5\}$, respectively.

Table 3, 4, and 5 show the community detection performances based on ARI, NMI, and Purity, respectively. The results are the average of 10 runs. On these tables, bold numbers represent the best results. We describe analysis on the tables as follows.

As you can see, our method PPNMF outperforms all the baseline methods except for cora dataset. For example, on ARI, NMI, and Purity results on flickr dataset, our method PPNMF achieves 4.1%, 5.0%, and 5.2% improvement compared to the second best method, respectively. Observe that SDNE results show relatively low in performance. SDNE is similar to PPNMF in that it balances the error penalty between observed edges and unobserved edges and it also preserves the first-order and second-order proximities. However, since SDNE is based on indirect approach, the relatively low results indicate that our assumption is correct: direct approach is superior to indirect approach in general. In addition, the relatively low performance of SDNE and LINE is explained by the fact that they are designed to capture the microscopic structure of networks, which is not suitable for community detection. DeepWalk and node2vec preserve not only the first-order and second-order proximities but also higher-order proximity. Accordingly, they can capture mesoscopic structure to an extent. However, they cannot capture large community structure in flickr. MNNF is an interesting method that adopts modularity measure to its loss function in addition to the first-order and second-order proximities. However, the matrix for representing modularity clustering is very sparse, each vertex belongs to a single community, so the quality of communities is very low.

3.4 Microbenchmark

Table 6, 7 and 8 show the results of microbenchmark. PPNMF outperforms PPNMF($\beta = 0.5$) on all datasets. This indicates that PPNMF generates better community assignment representation of vertexes by controlling the importance of observed edges over unobserved edges. In addition, PPNMF($\beta = 0.5$) outperforms SymNMF on all datasets. This demonstrates that the second-order proximity is essential for community detection. Finally, PPNMF outperforms PPNMF(W) on all datasets. This result supports our assumption: since the first-order proximity is more fundamental information than the second-order proximity, we should use the first-order proximity as the primary target of the loss function.

Table 6: Performance evaluation based on ARI with microbenchmark (bold numbers represent the best results)

method	PPNMF	PPNMF($\beta = 0.5$)	PPNMF(W)	SymNMF
parliament	**0.942**	0.993	0.632	0.584
polblog	**0.621**	0.600	0.604	0.584
cora	**0.366**	0.267	0.216	0.243
citeseer	**0.197**	0.147	0.087	0.084
blogcatalog	**0.160**	0.148	0.133	0.132
flickr	**0.152**	0.146	0.098	0.101

Table 7: Performance evaluation based on NMI with microbenchmark (bold numbers represent the best results)

method	PPNMF	PPNMF($\beta = 0.5$)	PPNMF(W)	SymNMF
parliament	**0.900**	0.890	0.715	0.685
polblog	**0.522**	0.494	0.502	0.483
cora	**0.446**	0.378	0.288	0.345
citeseer	**0.236**	0.223	0.134	0.187
blogcatalog	**0.237**	0.223	0.211	0.220
flickr	**0.222**	0.21 8	0.138	0.171

Table 8: Performance evaluation based on Purity with microbenchmark (bold numbers represent the best results)

method	PPNMF	PPNMF($\beta = 0.5$)	PPNMF(W)	SymNMF
parliament	**0.967**	0.962	0.887	0.886
polblog	**0.894**	0.888	0.884	0.863
cora	**0.633**	0.578	0.535	0.562
citeseer	**0.501**	0.449	0.360	0.405
blogcatalog	**0.427**	0.401	0.417	0.380
flickr	**0.400**	0.389	0.319	0.343

Table 9: Overall training time

Dataset	training time (sec)
parliament	0.36
polblog	4.44
cora	13.93
citeseer	19.67
blogcatalog	29.60
flickr	68.79

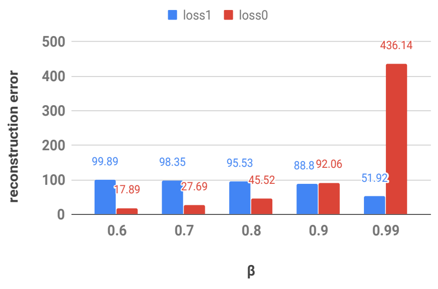
Table 10: Best β and Density (in descending order of Density)

Dataset	best β	Density (%)
parliament	0.60	2.86
polblog	0.70	0.80
blogcatalog	0.90	0.64
flickr	0.80	0.41
cora	0.99	0.07
citeseer	0.99	0.04

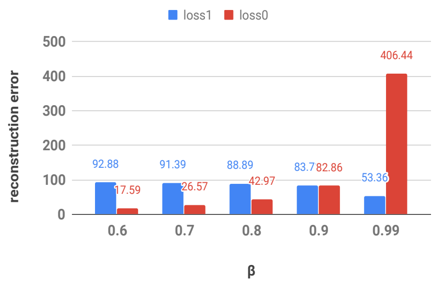
We show the runtime for overall training in Table 9. Despite of the large number of 500 iterations, on the largest dataset flickr, PPNMF completes the training in about 70 second.

3.5 Parameter analysis

We investigate the parameter sensitivity of PPNMF. PPNMF has two parameters β and λ. β controls the loss weight between observed edges and unobserved edges. λ balances the loss weight between the first-order proximity and the second-order proximity. We describe the analysis of each parameters as follows.

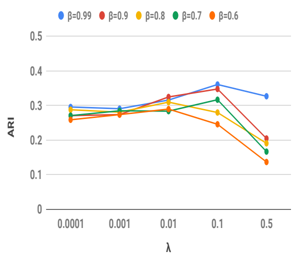


(a) cora

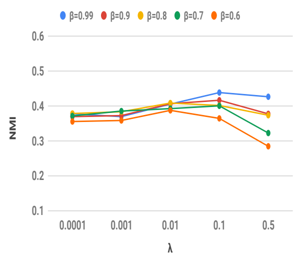


(b) citeseer

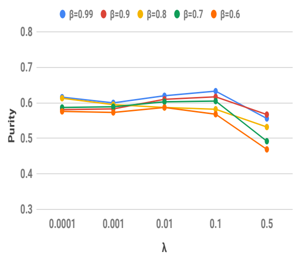
Fig. 1: Reconstruction error



(a) ARI



(b) NMI



(c) Purity

Fig. 2: Parameter sensitivity on cora

Loss weight controlling parameter β We show that best β and the density on each dataset in Table 10. The best β denotes β value that achieves the highest score in the metrics on each dataset. The density denotes $\frac{100E}{N^2}$ for each dataset. The table shows that the value of the best β is inversely correlated with the density of network. Actually, the best β of the densest network parliament is 0.6. On the other hand, the best β of the sparsest network citeseer is 0.99. It is our future work we automatically choose the best β depending on dataset by taking its sparsity into account.

Next, we introduce reconstruction error analysis. We show how largely the parameter β affects the reconstruction error on cora and citeseer in Figure 1. The loss1 represents the sum of reconstruction error between non-zero elements in adjacency matrix $\boldsymbol{A}$ and the elements of the expected adjacency matrix for non-zero elements in $\boldsymbol{A}$. The loss0 is defined similarly. The sum of loss1 and loss0 equals the overall reconstruction error. As β becomes larger, loss0 increases.

Balancing parameter between first-order and second-order proximity $\boldsymbol{\lambda}$ We show how largely the parameter λ affects the performance on cora network in Figure 2. As λ becomes larger, PPNMF concentrates more on the second-order proximity. When $\beta = 0.99$ (the best β for cora), we can see that the performance of $\lambda = 0.1$ is better than that of $\lambda = 0.0001$. It demonstrates that both the first-order and second-order proximities are essential for community detection.

4 Related Work

As we explained in Section 1, there are many recent work for community detection and network embedding methods.

4.1 Community detection

Real networks consist of multiple groups with dense connectivity. Group is often called community. Any two vertexes in the same community tend to have high similarity mutually. The purpose of community detection task is to reveal a community structure underlying in a given network. The task has been very popular and many researchers have put a lot of effort into it [18, 6, 9, 27, 32]. Traditional community detection methods, such as modularity-based methods [18, 6] and min-cut-based methods [7, 26], are characterized by the property of networks. The modularity-based methods are the most widely used. The methods detect communities through modularity optimization such as, by spectral optimization [19]. Traditional methods, including modularity-based methods, usually assign each vertex only to a single community.

4.2 Network embedding

Network embedding and representation learning methods generate low dimensional representation of vertexes. Principal component analysis (PCA) [12], Locally Linear Embedding [24] and Laplacian Eigenmaps [2] are classical network embedding methods. Recently, various network embedding or representation learning methods have been proposed [22, 28–30]. Although network embedding methods can obtain distributed representations of vertexes, they can not effectively capture community structure in two reasons. First, since they are based on indirect approaches, they tend to produce suboptimal results: they need to apply traditional clustering methods (such as k-means) to the distributed representations. k-means does not work well in such usage, because community structure can be nonlinear and k-means is not suitable for high dimension of distributed representations, which is known as "the curse of dimensionality" phenomena. Second, the representation is generated to capture the microscopic structure of network. This is not suitable for obtaining community structure, which requires to capture the mesoscopic structure of network.

SDNE [29] is similar to our method in that 1) it controls the error penalty between observed edges and unobserved edges, and 2) it preserves first-order and second-order proximities by using regularized autoencoder. However, since it is based on indirect approach, the quality of the community detection results is suboptimal (See Section 3).

DeepWalk [22] and node2vec [10] preserve higher order proximity by using random walks. node2vec employs biased random walks that can explore neighborhoods in breadth-first search (BFS) as well as in depth-first search (DFS). LINE [28] uses two functions that preserve the first-order proximity and second-order proximity independently. MNMF [30] incorporates modularity into network embedding to preserve community structures. Qiu et al. [23] prove that network embedding methods such as DeepWalk and LINE closely relate to matrix factorization.

4.3 NMF based method

NMF [15] factorizes a data matrix into two low-dimensional matrixes. Because of its high interpretability, NMF is widely used in many applications [4, 27, 32]. Especially, NMF is applied to community detection [13, 14, 27, 32]. SymNMF [13, 14] is an extension of NMF for graph clustering. SymNMF is related to spectral clustering [20], and takes a nonnegative similarity matrix as an input. NSED [27] is a nonnegative symmetric encoder-decorder approach. NSED uses NMF both in its encoder and decoder components. DANMF [32] is a deep autoencoder-like NMF model. DANMF learns hierarchical mappings between the original network and the final community assignment based on a deep autoencoder-like architecture. All the above NMF-base methods can directly obtain a community assignment representation of vertexes. However, these methods preserve either of first-order proximity or second-order proximity and do not balance the error penalty between observed edges and unobserved edges. In this paper, we aim to directly generate a community assignment representation that preserves first-order and second-order proximities and that is controlled by balancing the error penalty between observed edges and unobserved edge.

5 Conclusion

In this paper, we proposed a novel NMF-based model, namely PPNMF, to tackle the problem of community detection. Our model achieves a better, distributed, nonnegative and sparse representation of vertexes with the combination the first order and second order proximities. The extensive experimental results on community detection task demonstrate that PPNMF is superior to existing methods.

References

1. Adamic, L.A., Adar, E.: Friends and neighbors on the web. Social networks **25**(3), 211–230 (2003)
2. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Advances in neural information processing systems. pp. 585–591 (2002)
3. Bojchevski, A., Günnemann, S.: Bayesian robust attributed graph clustering: Joint learning of partial anomalies and group structure. In: Proceedings of AAAI (2018)
4. Cai, D., He, X., Wu, X., Han, J.: Non-negative matrix factorization on manifold. In: Proceedings of ICDM. pp. 63–72 (2008)
5. Chakraborty, T., Dalmia, A., Mukherjee, A., Ganguly, N.: Metrics for community analysis: A survey. ACM Computing Surveys (CSUR) **50**(4), 54 (2017)
6. Clauset, A., Newman, M.E., Moore, C.: Finding community structure in very large networks. Physical review E **70**(6), 066111 (2004)
7. Ding, C.H., He, X., Zha, H., Gu, M., Simon, H.D.: A min-max cut algorithm for graph partitioning and data clustering. In: Proceedings of ICDM. pp. 107–114 (2001)
8. Flake, G.W., Lawrence, S., Giles, C.L., Coetzee, F.M.: Self-organization and identification of web communities. Computer (2002)
9. Fortunato, S.: Community detection in graphs. Physics reports (2010)
10. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of SIGKDD. pp. 855–864 (2016)
11. Huang, X., Li, J., Hu, X.: Accelerated attributed network embedding. In: Proceedings of SDM. pp. 633–641 (2017)
12. Jolliffe, I.: Principal component analysis. Springer (2011)
13. Kuang, D., Ding, C., Park, H.: Symmetric nonnegative matrix factorization for graph clustering. In: Proceedings of the 2012 SIAM international conference on data mining. pp. 106–117. SIAM (2012)
14. Kuang, D., Yun, S., Park, H.: SymNMF: nonnegative low-rank approximation of a similarity matrix for graph clustering. Journal of Global Optimization **62**(3), 545–574 (2015)
15. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: Advances in neural information processing systems. pp. 556–562 (2001)
16. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. Journal of the American society for information science and technology **58**(7), 1019–1031 (2007)
17. McCallum, A.K., Nigam, K., Rennie, J., Seymore, K.: Automating the construction of internet portals with machine learning. Information Retrieval **3**(2), 127–163 (2000)

18. Newman, M.E.: Fast algorithm for detecting community structure in networks. Physical review E **69**(6), 066133 (2004)
19. Newman, M.E.: Spectral methods for community detection and graph partitioning. Physical Review E **88**(4), 042822 (2013)
20. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: Advances in neural information processing systems. pp. 849–856 (2002)
21. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: Proceedings of SIGKDD. pp. 1105–1114 (2016)
22. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of SIGKDD. pp. 701–710 (2014)
23. Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., Tang, J.: Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In: Proceedings of WSDM. pp. 459–467 (2018)
24. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. science **290**(5500), 2323–2326 (2000)
25. Sahu, S., Mhedhbi, A., Salihoglu, S., Lin, J., Özsu, M.T.: The Ubiquity of Large Graphs and Surprising Challenges of Graph Processing. PVLDB (2017)
26. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on pattern analysis and machine intelligence **22**(8), 888–905 (2000)
27. Sun, B.J., Shen, H., Gao, J., Ouyang, W., Cheng, X.: A non-negative symmetric encoder-decoder approach for community detection. In: Proceedings of CIKM. pp. 597–606 (2017)
28. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: Proceedings of WWW. pp. 1067–1077 (2015)
29. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: Proceedings of SIGKDD. pp. 1225–1234 (2016)
30. Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., Yang, S.: Community preserving network embedding. In: Proceedings of AAAI (2017)
31. Xu, X., Yuruk, N., Feng, Z., Schweiger, T.A.: SCAN: a structural clustering algorithm for networks. In: Proceedings of SIGKDD. pp. 824–833 (2007)
32. Ye, F., Chen, C., Zheng, Z.: Deep autoencoder-like nonnegative matrix factorization for community detection. In: Proceedings of CIKM. pp. 1393–1402 (2018)
33. Yuan, Z., Oja, E.: Projective nonnegative matrix factorization for image compression and feature extraction. In: Scandinavian Conference on Image Analysis. pp. 333–342. Springer (2005)