Structural Property-aware Multilayer Network Embedding for Latent Factor Analysis

Jie Lu[a,*], Junyu Xuan[a,b], Guangquan Zhang[a], Xiangfeng Luo[b]

[a] *Centre for Artificial Intelligence, Faculty of Engineering and Information Technology, University of Technology Sydney, PO Box 123, Broadway, NSW 2007, Sydney, Australia*
[b] *School of Computer Engineering and Science, Shanghai University, 99 Shangda Road, Shanghai, China*

Abstract

Multilayer network is a structure commonly used to describe and model the complex interaction between sets of entities/nodes. A three-layer example is the author-paper-word structure in which authors are linked by co-author relation, papers are linked by citation relation, and words are linked by semantic relation. Network embedding, which aims to project the nodes in the network into a relatively low-dimensional space for latent factor analysis, has recently emerged as an effective method for a variety of network-based tasks, such as collaborative filtering and link prediction. However, existing studies of network embedding both focus on the single-layer network and overlook the structural properties of the network, e.g., the degree distribution and communities, which are significant for node characterization, such as the preferences of users in a social network. In this paper, we propose four multilayer network embedding algorithms based on Nonnegative Matrix Factorization (NMF) with consideration given to four structural properties: whole network (NNMF), community (CNMF), degree distribution (DNMF), and max spanning tree (TNMF). Experiments on synthetic data show that the proposed algorithms are able to preserve the desired structural properties as designed. Experiments on real-world data show that multilayer network embedding improves the accuracy of document clustering and rec-

[*] Corresponding author

Email addresses: `Jie.Lu@uts.edu.au` (Jie Lu), `Junyu.Xuan@uts.edu.au` (Junyu Xuan), `Guangquan.Zhang@uts.edu.au` (Guangquan Zhang), `luoxf@shu.edu.cn` (Xiangfeng Luo)

ommendation, and the four embedding algorithms corresponding to the four structural properties demonstrate the differences in performance on these two tasks. These results can be directly used in document clustering and recommendation systems.

Keywords: Multilayer Network, Network Embedding, Nonnegative Matrix Factorization

1. Introduction

Multilayer network [1] is a structure commonly used to describe and model the complex interaction between sets of entities/nodes. The structure has attracted the attention of researchers from many areas, such as computer scientists, sociologists, physicists, and biologists, due to its pervasiveness. As a result, research into multilayer network has become a multidisciplinary area of study. To date, many types of multilayer network with a variety of structures and names have been developed in the literature [1, 2, 3]. Multilayer network, as defined in this study, is composed of several homogeneous networks in multiple layers and the nodes in different layers may have external links across the layers, as illustrated in Fig. 1. One example, in the text mining area, is the author-paper-keyword structure shown in Fig. 2. This structure is a multilayer network because it is composed of three layered networks (i.e., author social network, paper citation network and keyword co-occurrence network) and the nodes across networks are also linked (i.e., an author and a paper are linked if this author writes this paper, and a paper and a word are linked if this paper contains this word). In recommender systems, a multilayer network is composed of tag-user-movie mapping relations with tag similarity network, user social network and movie similarity network, as shown in Fig. 3. Multilayer network would also be a good choice for big data modelling because there are complex interactions between multiple sources or attributes due to the Variety property of big data [4]. Hence, it is crucial and urgent to develop more effective analytic tools for multilayer network to obtain better understanding and improving behaviour prediction of its underlying complex systems.

Network embedding has recentely emerged as an effective method for a variety of network-based tasks, such as collaborative filtering and link prediction. Its basic idea is to project the nodes in the network into a relatively low-dimensional space and provide each node with a new vector-based rep-

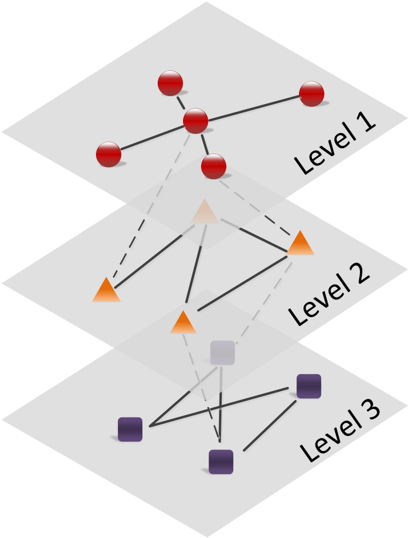

Figure 1: Multilayer network

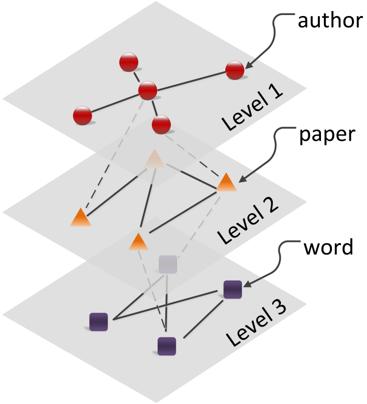

Figure 2: An instance in Text mining

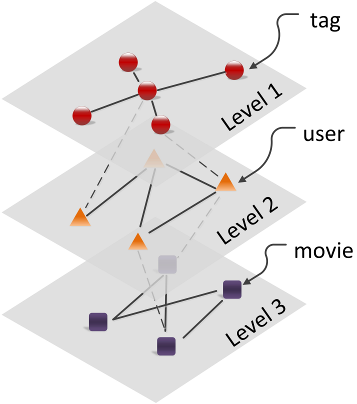

Figure 3: An instance in Recommender Systems

resentation. It is commonly believed that this new representation is not only more concise but also more innate, because it is expected to preserve the important characteristics of the network. One simple example is that each person is given a vector-based representation after a social network has been embedded, and two persons will be recommended as friends based on the similarity between their new vector-based representations. A similar example is that documents could be more accurately clustered using the new vector-based representation instead of the original word-based representation. A number of methods have been proposed in the literature for network embedding, including matrix facorization [5, 6], random walk [7, 8, 9], deep learning[10, 11], and so on. However, all these state-of-the-art methods are designed for a single-layer network and, importantly, do not consider the structural properties, i.e., community, degree distribution, and max spanning tree, during network embedding. Structural properties are the significant statistical properties of complex networks, and sometimes they have a greater ability to characterize the nature of a node than the whole network. Unfortunately, these structural properties are overlooked by existing network embedding methods.

In this paper, we propose four multilayer network embedding algorithms based on Nonnegative Matrix Factorization (NMF)[12] with considerations given to four different structural properties: whole network (NNMF), community (CNMF), degree distribution (DNMF) and max spanning tree (TNMF). Four objective functions are carefully designed to preserve the desired structural properties along with the multilayer network embedding. To optimize each objective function, the corresponding update rules are introduced. Experiments on synthetic data show that the designed algorithms have the ability to preserve the desired structural properties. To show the usefulness of these algorithms, two real-world tasks, document clustering and recommendation, are carried out. The results show that the proposed algorithms perform better than traditional NMF and other algorithms in achieving clustering and recommendation accuracy. A natural problem is to determine the difference between these structural properties in their impact on the real-world tasks, e.g., recommendation or clustering performance. To evaluate these differences, we conduct extensive experiments to compare the performance of each on two real-world tasks. As the experimental results show, we can achieve better performance by preserving the structural properties. We also compare the performance when different structural properties are retained in these tasks.

This paper makes the following contributions:

1. Four structural properties-aware multilayer network embedding algorithms based on nonnegative matrix factorization are proposed with consideration given to four significant structural properties.
2. Extensive experiments are conducted to show the ability of the proposed multilayer network embedding algorithms on the preservation of structural properties, and to compare the performances of the four

designed algorithms on two real-world tasks.

The rest of this paper is organized as follows. Section 2 reviews related work. The problem is formally defined in Section 3. Our algorithms for multilayer network embedding are proposed in Section 4. Experiments on synthetic data and real-world data are conducted in Section 5. Lastly, Section 6 concludes the study and discusses future work.

2. Related work

Since our motivation is to use complex network structural properties as constraints for nonnegative matrix factorization, this section is composed of two parts: 1) we will discus elementary introductions to, and research on, the structural properties of a complex network, and 2) we will discus recent works on network-related factorization models or algorithms.

2.1. Multilayer network

Complex network is an interdisciplinary research, attracting researchers from computer science, physics, sociology, biology, and so on. Due to the pervasiveness of the network phenomenon, complex network has been adopted to model many things, such as the users' friend network and the cell network in the brain. Comparing to the graph, the complex network area focuses more on the non-trivial structural properties. The two outstanding properties are small-world network published in Nature [13] and power-law degree distribution [14] published in Science. Besides, many other different network structural properties have also been discovered and defined in this area [15, 16]. However, it is commonly accepted that the following network structural properties are the most fundamental and significant for describing the structure of a complex network: community [17, 18], degree distribution [19, 20, 21], and max spanning tree [22, 23]. Recently, the multilayer network has attracted the attention of researchers. Its mathematical formulation is given in [3]. Similar to the one-layer complex network, its structures are defined and discussed in [2]. Apart from formalization and structure definition, the multilayer network has been used for modeling the influence propagation over microblogs [24] and the analysis and management of change propagation [25]. However, most state-of-the-art research on multilayer networks focuses on basic structure analysis. Since the traditional network structural properties (i.e., community, degree distribution and max spanning tree) do not consider the directions of the edges in the network and our aim is to preserve these network properties after the embedding, we assume in this paper that the network is undirected.

2.2. Network-related factorization models/algorithms

Nonnegative Matrix Factorization is a dominant tool for the recommender systems and document clustering in the literature. First, we give a brief introduction to nonnegative matrix factorization (NMF). Given a nonnegative matrix $Y_{a\times x}$ (extended to semi-nonnegative by [26]), the NMF aims to find two matrices $A_{a\times k}$ and $X_{k\times x}$ to minimize the following cost function

$$J\left(A,X\right)=\frac{1}{2}\|Y-AX\|_F^2, \tag{1}$$

where $\|\cdot\|_F$ is the Frobenius norm and the elements of A and X are also nonnegative. In the literature, constraints are added to the A or X in the cost function in Eq. (1) for purposes such as sparseness constraint[27], smooth constraint[28], orthogonal constraint[29], and label information [30]. All these constraints aim to make the discovered (k-dimensional) latent space preserve more properties.

The networks/relations between instances within the given dataset have been considered in a number of ways in NMF. One way is to let users define the must-link and cannot-link relations between instances [31], and then use these relations as constraints for the NMF. Another way is to construct a graph (the nodes are instances and links are relations between instances) as the constraint of NMF (also called graph-regularization [32, 33]). As we prove later, this constraint only preserves the community property of network. Some works have jointly considered two-side information during factorization. For example, the constraint in [34] considers user similarity network and post content; the constraint in [35] considers graphs from multiple domains. There are also works on using NMF to find the community structures of a network [36] or two networks [37] but not as constraints, as in this paper. These works are similar to this study, but they do not explore the structural properties of graphs. As aforementioned, there are many important properties for a given network. However, these network structure properties are disregarded during factorization, so our contribution is that the different structural properties are considered during factorization.

To summarize, many researchers have noticed the importance of network structure, and it has been considered in a variety of models. From their work, we can see that the network structure indeed can help to unveil the nature of data. However, to the best of our knowledge, little work has been done to consider the influences of complex network structural properties except the community, and other significant complex network structural properties (i.e., degree distribution and max spanning tree) are overlooked.

3. Problem definition

In this section, we will formally define and explain multilayer network and its embedding, and the designed algorithms to resolve this problem will be given in the subsequent section.

The multilayer network in this study is defined as,

Definition 1 (Multilayer Network, Ω)**.** *Multilayer network is composed of nodes with different characters. It includes horizontal networks H between nodes with the same character and vertical networks V between nodes with a different character, as shown in Fig. 1.*

This is an abstract and very common structure that can be used to model data from different areas. For example,

- in recommender systems, there is a multilayer network structure: tag-user-movie, in which users may have trust relations with each other, tags may have correlation relations with each other and movies may have similarity relations due to their genre information;
- in the text mining area, there is a multilayer network structure: author-paper-keyword, in which authors may have cooperation relations with each other, papers may have citation relations with each other and keywords may have semantic relations with each other.

The friend relations between users will apparently impact on users' ratings on movies, and the citation relations between papers will impact on the keyword usage of each paper. Thus, horizontal and vertical networks should be jointly considered when embedding a multilayer network which in this study is defined as,

Definition 2 (Multilayer Network Embedding)**.** *Multilayer network embedding is to project all the nodes into a latent k-dimensional space. Based on* *the NMF, it can be expressed as*

$$J\left(A_{n\times k}, X_{k\times p}\right) = \frac{1}{2}\|V_{n\times p} - A_{n\times k}X_{k\times p}\|_F^2 + \alpha \cdot \frac{1}{2}\|H_{n\times n} - A_{n\times k}A_{n\times k}^T\|_F^2, \tag{2}$$

where $A_{n\times k}$ is a nonnegative matrix with rows corresponding to one kind of node in the network $V_{n\times p}$ and $X_{k\times p}$ corresponding to the other kind of node in $V_{n\times p}$. We can see that nodes with different characters in $V_{n\times p}$ and $H_{n\times n}$ are all projected to a k-dimensional space, and A and X are the embedding results. α is the parameter used to adjust the weights of two parts in the cost function.

Here, H only represents one horizontal network, but multiple layers can easily be achieved by adding more items on the right side of Eq. (2). We only discuss one layered horizontal network for the sake of simplicity, and the comprehensive analysis of the situations with a different number of layers will be tested in the experiment section.

Note that the nonnegativity condition of A and X is useful. As discussed in [38, 39], NMF does not allow negative entries in the matrix factors A and X. These non-negativity constraints permit the combination of multiple basis 'factors' to represent the original matrix (i.e., the rating matrix of users by movies or a face image). But only additive combinations are allowed, because the non-zero elements of A and X are all positive. In contrast to real-valued embedding (without nonnegative constraints), no subtractions can occur. For these reasons, the non-negativity constraints are compatible with the intuitive notion of combining parts to form a whole, which is how NMF learns a parts-based representation. Each 'part' is in the range of normal rating space. Therefore, each 'part' can be regarded as a representative rating profile from a user community or interest group, and each user's ratings can be modeled as an additive mixture of rating profiles from user communities or interest groups. A user community can be thought of as an expression of a particular statistical pattern in the opinions of users, and typically has some kind of real world meaning. For example, a user community might be characterized by giving high ratings to programming books and low ratings to other books, and thus constitute a 'computer' group.

One problem when conducting multilayer network embedding is how to preserve the network structural properties, such as community, degree distribution and max spanning tree, after embedding to the new and relatively small k-dimensional latent space. Note that the minimization of Eq. (2)

cannot ensure the different network structural properties will definitely be preserved, as will be demonstrated in the experiments section.

4. Structural property-aware multilayer network embedding

In this section, we will introduce four algorithms for multilayer network embedding to preserve four different structural properties: whole network, community, degree distribution, and max spanning tree, respectively. There are two layers of nodes with vertical network $V_{n\times p}$, but only one layer horizontal network $H_{n\times n}$ will be considered for brevity.

4.1. NNMF: Preserve the whole network

In this situation, all edges in the network have the same status. The cost function is the same as Eq. (2). Since Eq. (2) contains second-order of matrix A, an approximation with lower computation complexity can be made by

$$P = \arg\min_{P\geq 0}\left\{\frac{1}{2}\|H - PP^T\|_F^2\right\} \tag{3}$$

and

$$J\left(A, X\right) = \frac{1}{2}\|V - AX\|_F^2 + \alpha\frac{1}{2}\|P - A\|_F^2. \tag{4}$$

The first equation is a symmetric NMF for network H. We can then obtain an optimized latent space which preserves the whole network and the new vector-based representation P of nodes (embedding results) by this latent space.

The derivatives of Eq. (4) with respect to A is

$$\begin{aligned}\frac{\partial J}{\partial A} &= \big(-VX^T + AXX^T\big) + \alpha\big(A - P\big)\\ &= A(XX^T + \alpha\cdot I) - (VX^T + \alpha P)\end{aligned} \tag{5}$$

and according to KKT condition

$$\big[\big(-VX^T + AXX^T\big) + \alpha\big(A - P\big)\big]_{ij}A_{ij} = 0, \tag{6}$$

the update rule is set as

$$A_{ij}^{t+1} \leftarrow A_{ij}^{t}\sqrt{\frac{\big[VX^T + \alpha P^{+}\big]_{ij}}{\big[AXX^T + \alpha A + \alpha P^{-}\big]_{ij}}} \tag{7}$$

where $P^+ = (|P_{ij}| + P_{ij})/2$, $P^- = (|P_{ij}| - P_{ij})/2$ and $P = P^+ - P^-$. For X, the derivative is

$$\frac{\partial J}{\partial X} = -A^T V + A^T A X. \tag{8}$$

According to KKT condition

$$\big[- A^T V + A^T A X \big]_{ij} X_{ij} = 0, \tag{9}$$

the update rule for X is

$$X_{ij} \leftarrow X_{ij} \left[\frac{\big[A^T V\big]_{ij}}{\big[A^T A X\big]_{ij}} \right]^{1/2}. \tag{10}$$

Since the update rule of X is the same as traditional NMF, we only discus the property of update rule of A.

Theorem 1. The cost function in Eq. (4) is nonincreasing under the update rules (7).

The proof of this theorem is given in the Appendix. We can see from the update rules for A and X that the A is influenced by the horizontal network structure in (7) since we only consider one-layer network. The update rule of X is the same as (10), since the new term does not impact on the derivative of cost function with respect to X.

However, the update rule (7) can only guarantee the nonincreasing property but not the convergence[40, 41]. The Eq. (38) is equal to update A through the gradient descent method with a special step size [42, 43]. The corresponding step size of Eq. (38) is

$$\eta_A = A_{ij} \left[\Big((AXX^T + \alpha A + \alpha P^-)_{ij}^{1/2} + (VX^T + \alpha P^+)_{ij}^{1/2} \Big) \big(AXX^T + \alpha A + \alpha P^- \big)_{ij}^{1/2} \right]^{-1} \tag{11}$$

As pointed out in [42, 41], Eq. (38) might not reach a stationary point due to improper step size. To ensure this update definitely reach a stationary point, we revise this step size [42] as,

$$\overline{\eta}_A = \overline{A_{ij}} \left[\Big((\overline{A}XX^T + \alpha \overline{A} + \alpha P^-)_{ij}^{1/2} + (VX^T + \alpha P^+)_{ij}^{1/2} \Big) \big(\overline{A}XX^T + \alpha \overline{A} + \alpha P^- \big)_{ij}^{1/2} + \delta \right]^{-1} \tag{12}$$

Algorithm 1: NNMF

```
Input: $H$ and $V$, Maximum Iteration number: $I_{\max}$
Output: $A$ and $X$
$P = \arg\min_{P \geq 0} \left\{ \frac{1}{2} \|H - PP^T\|_F^2 \right\}$;
while $i < I_{\max}$ do
    if $\frac{\partial J}{\partial A_{ij}} < 0$ in Eq. (5) then
        $\overline{A_{ij}} = max(A_{ij}, \sigma)$;
    compute $\overline{\eta}_A$ by Eq. (12);
    $A_{ij} \leftarrow A_{ij} - \overline{\eta}_A \frac{\partial J}{\partial A_{ij}}$;
    if $\frac{\partial J}{\partial X_{ij}} < 0$ in Eq. (8) then
        $\overline{X_{ij}} = max(X_{ij}, \sigma)$;
    compute $\overline{\eta}_X$ by Eq. (14);
    $X_{ij} \leftarrow X_{ij} - \overline{\eta}_X \frac{\partial J}{\partial X_{ij}}$;
    $i = i + 1$;
```

where

$$\overline{A_{ij}} = \begin{cases} A_{ij}, & if \frac{\partial J}{\partial A_{ij}} \geq 0 \\ max(A_{ij}, \sigma), & if \frac{\partial J}{\partial A_{ij}} < 0 \end{cases} \tag{13}$$

and δ and σ are two small positive numbers.

For X, the corresponding step size of update rule (10) is

$$\overline{\eta}_X = \overline{X}_{ij} \left[\left((A^T A \overline{X})_{ij}^{1/2} + (A^T V)_{ij}^{1/2} \right) \cdot \left(A^T A \overline{X} \right)_{ij}^{1/2} + \delta \right]^{-1} \tag{14}$$

The whole procedure is summarized in Algorithm 1.

4.2. CNMF: Preserve community

Community [16, 44, 37] is an important structural property of a complex network. Like the whole network, our idea is to project the nodes into a latent space which can preserve community rather than the whole network. Then, the problem becomes how to find a space to preserve the community. The Laplacian matrix, which is broadly used in spectral analysis[45, 46], of the original network matrix, H, is considered here. One of its definition is,

$$\begin{aligned} L &= D - H \\ (P, \lambda) &= svd(L) \end{aligned} \tag{15}$$

where D is the degree matrix defined as $d_{i,i} = \sum_j h_{i,j}$ and $d_{i,j} = 0 (i \neq j)$, $svd(\cdot)$ is the singular value decomposition operation, P is the eigenvectors and λ is eigenvalues.

With this latent space P, the cost function can be easily designed as

$$J(A, X) = \frac{1}{2}\|V - AX\|_F^2 + \alpha \cdot \frac{1}{2}\|P_k - A\|_F^2, \tag{16}$$

where P_k is the first k eigenvectors in P. Then, the update rules are the same as (7) and (10). Next, we try to prove the ability to preserve the community property of the designed cost function.

Theorem 2. The update rules (7) and (10) for cost function in Eq. (16) have the ability to preserve the community structure.

Proof. P's ability to preserve the community property originates from graph cut theory. In this theory, separating a group of nodes into k subgroups is equal to optimizing the following cost function

$$\begin{aligned} \min_G RatioCut(G_1, G_2, ..., G_k) &= \min_G \left\{ \frac{1}{2} \sum_{i=1}^{k} \frac{W(G_i, \overline{G_i})}{|G_i|} \right\} \\ &= \min_Q Tr(Q^T L Q), \ s.t. \ Q^T Q = I \end{aligned} \tag{17}$$

where $G = G_1, G_2, ..., G_k$ is the partition of nodes, $|G_i|$ is the number of nodes in G_i, and W is a designed indicator matrix (more details can be found in [45]). With a small relaxation, the solution Q in Eq. (17) is just the matrix that contains the first k eigenvectors of L. That means our P_k can optimize the cost function in Eq. (17), and then can give the best partition of the network. Therefore, the cost function in Eq. (16) is able to preserve the community property of the original network H. □

It should be noted that the cost function in Eq. (16) has the same trend with

$$J(A, X) = \frac{1}{2}\|V - AX\|_F^2 + \alpha \cdot tr(A^T L A). \tag{18}$$

```
Algorithm 2: CNMF
Input: $H$ and $V$, Maximum Iteration number: $I_{\max}$
Output: $A$ and $X$
$L = D - H$;
$P = svd(L)$;
while $i < I_{\max}$ do
    if $\frac{\partial J}{\partial A_{ij}} < 0$ in Eq. (5) then
        $A_{ij} = max(A_{ij}, \sigma)$;
    compute $\overline{\eta}_A$ by Eq. (12);
    $A_{ij} \leftarrow A_{ij} - \overline{\eta}_A \frac{\partial J}{\partial A_{ij}}$;
    if $\frac{\partial J}{\partial X_{ij}} < 0$ in Eq. (8) then
        $X_{ij} = max(X_{ij}, \sigma)$;
    compute $\overline{\eta}_X$ by Eq. (14);
    $X_{ij} \leftarrow X_{ij} - \overline{\eta}_X \frac{\partial J}{\partial X_{ij}}$;
    $i = i + 1$;
```

This equation directly combines the NMF and Graph-Cut, which are commonly adopted by other researches [32, 47] for graph-regularization. The difference between Eq. (16) and Eq. (18) is the same as the difference between Eq. (4) and Eq. (2).

The whole procedure is summarized in Algorithm 2.

4.3. DNMF: Preserve degree distribution

The degree distribution [16, 14, 21] is another very important structural property. A node i, in a network will have a number of neighbours as d_i. The degree distribution can be described by a function of $n_d \sim \pi(d)$, where n_d denotes the number of nodes with the same degree d and $\pi(\cdot)$ is the distribution of the number of degrees. Our idea to preserve the degree distribution is to maximize the correlation between the node degree sequences of the original matrix and the new generated matrix (formed by k-dimensional latent space).

Suppose the network matrix H is a binary matrix,

$$H_{n \times n} \mathbf{1}^{n \times 1} \tag{19}$$

will be a vector containing degrees of all nodes. Similarly,

$$AA^T \mathbf{1}^{n \times 1} \tag{20}$$

is also a vector containing degrees of all nodes in the k-dimensional latent space. The distance between two degree vectors can be evaluated by,

$$\|H_{n\times n}\mathbf{1}^{n\times 1} - AA^T\mathbf{1}^{n\times 1}\|_F^2 \tag{21}$$

If we can minimize this distance, the degree distributions in original space and new latent space will be similar. If the H is not a binary matrix but a real-valued matrix, the degree sequence an be seen as a *weighted degree distribution*. Then, a new cost function with this term is

$$J(A,X) = \frac{1}{2}\|V - AX\|_F^2 + \frac{1}{2}\alpha \cdot \|H_{n\times n}\mathbf{1}^{n\times 1} - AA^T\mathbf{1}^{n\times 1}\|_F^2. \tag{22}$$

The derivative of A is

$$\begin{aligned}\frac{\partial J(A,X)}{\partial A} &= -VX^T + AXX^T - \alpha \cdot H\mathbf{1}^{n\times 1}\mathbf{1}^{1\times n}A \\ &+ 2\alpha \cdot AA^T\mathbf{1}^{n\times 1}\mathbf{1}^{1\times n}A\end{aligned} \tag{23}$$

The update rule of A is set

$$A_{ij} \leftarrow A_{ij}\frac{\left[\left((AXX^T)_{ij}^2 + 8\alpha(AA^T\mathbf{1}A)_{ij}(VX^T + \alpha H\mathbf{1}A)_{ij}\right)^{1/2} - (AXX^T)_{ij}\right]^{\frac{1}{2}}}{\left[(AA^T\mathbf{1}^{n\times n}A)_{ij}\right]^{\frac{1}{2}}} \tag{24}$$

Theorem 3. The cost function in Eq. (22) is nonincreasing under the update rules (24).

The proof of this theorem is given in the Appendix. In order to ensure reaching to a stationary point, the revised step size for the update in (24) is,

$$\begin{aligned}\overline{\eta}_A = \overline{A}_{ij} \cdot &\left[(4\alpha \cdot (\overline{AA}^T\mathbf{1}\overline{A})_{ij})^{1/2} - \left((\overline{A}XX^T)_{ij}^2 + 8\alpha(\overline{AA}^T\mathbf{1}\overline{A})_{ij} \cdot (VX^T + \alpha \cdot H\mathbf{1}\overline{A})_{ij}\right)^{1/2}\right. \\ &\left.- (\overline{A}XX^T)_{ij}\right] \Big/ \left[(4\alpha \cdot (\overline{AA}^T\mathbf{1}\overline{A})_{ij})^{1/2} \cdot (-VX^T + \overline{A}XX^T - \alpha \cdot H\mathbf{1}\overline{A} + 2\alpha \cdot \overline{AA}^T\mathbf{1}\overline{A}) + \delta\right]\end{aligned} \tag{25}$$

where $\overline{A}$ satisfies Eq. (13) with $\frac{\partial J}{\partial A}$ replaced by Eq. (23).

The whole procedure is summarized in Algorithm 3.

```
Algorithm 3: DNMF
  Input: $H$ and $V$, Maximum Iteration number: $I_{\max}$
  Output: $A$ and $X$
  while $i < I_{\max}$ do
      if $\frac{\partial J}{\partial A_{ij}} < 0$ in Eq. (23) then
          $A_{ij} = max(A_{ij}, \sigma)$;
      compute $\overline{\eta}_A$ by Eq. (25);
      $A_{ij} \leftarrow A_{ij} - \overline{\eta}_A \frac{\partial J}{\partial A_{ij}}$;
      if $\frac{\partial J}{\partial X_{ij}} < 0$ in Eq. (8) then
          $X_{ij} = max(X_{ij}, \sigma)$;
      compute $\overline{\eta}_X$ by Eq. (14);
      $X_{ij} \leftarrow X_{ij} - \overline{\eta}_X \frac{\partial J}{\partial X_{ij}}$;
      $i = i + 1$;
```

4.4. TNMF: Preserve max spanning tree

The max spanning tree [16, 44] of network H is first mined,

$$T \circledast H = T^H \tag{26}$$

where $\circledast$ denotes the Hadamard (element-wise) product and T^H is the mined max spanning tree of network H and T is called the *tree-mask matrix* which is a binary matrix with $t_{i,j} = 1$ if it is in T^H otherwise $t_{i,j} = 0$, and $\overline{T}$ is the complement of T with $t_{i,j} = 1$ if it is not in T^H otherwise $t_{i,j} = 0$.

The cost function is

$$J(A, X) = \frac{1}{2}\|V - AX\|_F^2 - \alpha \cdot \frac{1}{4}\|T \circledast (AA^T) - \overline{T} \circledast (AA^T))\|_F^2. \tag{27}$$

The derivative with respect to A is

$$\frac{\partial J(A, X)}{\partial A} = \big(- VX^T + AXX^T\big) + \alpha \cdot (\overline{T} - T) \circledast (AA^T)A \tag{28}$$

The update rule for A is

$$A_{ij} \leftarrow A_{ij} \frac{\left[\left((AXX^T)^2_{ij} + 4\alpha(\overline{T} \circledast (AA^T)A)_{ij}(VX^T + \alpha T \circledast (AA^T)A)_{ij}\right)^{1/2} - (AXX^T)_{ij}\right]^{\frac{1}{2}}}{\left[2\alpha(\overline{T} \circledast AA^TA)\right]^{\frac{1}{2}}} \tag{29}$$

Theorem 4. The cost function in Eq. (27) is nonincreasing under the update rules (29).

The proof of this theorem is given in the Appendix. In order to ensure reaching to a stationary point, the revised step size for the update in Eq. (29) is,

$$\begin{aligned}\overline{\eta}_A =& \overline{A}_{ij} \cdot \Bigg[\left(2\alpha \cdot (\overline{T} \circledast (\overline{AA}^T)\overline{A}\right)^{1/2}_{ij} - \Bigg((\overline{A}XX^T)^2_{ij} \\ & + 4\alpha(\overline{T} \circledast (\overline{AA}^T)\overline{A})_{ij}(VX^T + \alpha T \circledast (\overline{AA}^T)\overline{A})_{ij} \Bigg)^{1/2} \\ & - (\overline{A}XX^T)_{ij} \Bigg] \Big/ \Bigg[(2\alpha \cdot \left(\overline{T} \circledast (\overline{AA}^T)\overline{A}\right)^{1/2}_{ij} \\ & \cdot \left(-VX^T + \overline{A}XX^T + \alpha \cdot (\overline{T} - T) \circledast (\overline{AA}^T)\overline{A}\right)_{ij} + \delta \Bigg]\end{aligned} \tag{30}$$

where $\overline{A}$ satisfies Eq. (13) with $\frac{\partial J}{\partial A}$ replaced by Eq. (28).

The whole procedure is summarized in Algorithm 4.

5. Experimental results and analysis

This section is composed of two parts: experiments on synthetic data and experiments on real-world data. The first part is used to verify the correctness of our proposed algorithms, and the second part is used to show the usefulness of our work. For the sake of brevity, abbreviations are used as follows: NNMF for the algorithm proposed in Section 4.1 to preserve the whole network; CNMF for the algorithm proposed in Section 4.2 to preserve

Algorithm 4: TNMF

```
Input: $H$ and $V$, Maximum Iteration number: $I_{\max}$
Output: $A$ and $X$
Obtain Max spanning tree $T$ of $H$;
while $i < I_{\max}$ do
    if $\frac{\partial J}{\partial A_{ij}} < 0$ in Eq. (28) then
        $\overline{A_{ij}} = max(A_{ij}, \sigma)$;
    compute $\overline{\eta}_A$ by Eq. (30);
    $A_{ij} \leftarrow A_{ij} - \overline{\eta}_A \frac{\partial J}{\partial A_{ij}}$;
    if $\frac{\partial J}{\partial X_{ij}} < 0$ in Eq. (8) then
        $\overline{X_{ij}} = max(X_{ij}, \sigma)$;
    compute $\overline{\eta}_X$ by Eq. (14);
    $X_{ij} \leftarrow X_{ij} - \overline{\eta}_X \frac{\partial J}{\partial X_{ij}}$;
    $i = i + 1$;
```

Table 1: Evaluation metrics of document clustering/communities

Jaccard Coefficient	$JC = \frac{a}{a+b+c}$
Folkes & Mallows	$FM = \left(\frac{a}{a+b} \cdot \frac{a}{a+c}\right)^{1/2}$
F1 measure	$F1 = \frac{2a^2}{2a^2+ac+ab}$

the community; DNMF for the algorithm proposed in Section 4.3 to preserve the degree distribution; TNMF for the algorithm proposed in Section 4.4 to preserve the max spanning tree.

5.1. Experiments on synthetic data

In this section, we randomly generate a multilayer network represented by two matrices, i.e., V and H, to verify the abilities of the proposed algorithms to preserve the desired network structural properties.

5.1.1. Test for community preservation

For a given multilayer network $< V, H >$, we want to embed $< V, H >$ and preserve the community of network H at the same time. To show the ability of CNMF to keep the community property of the original network H, we compare the communities of the re-constructed networks (H_{NNMF} and H_{CNMF}) with the communities of the original network H. Here, NNMF is a

Table 2: Comparison between NNMF and CNMF on preserving community

Algorithm	n=10 (K=3)		
	JC	FM	F1
NNMF	0.1914 ± 0.0642	0.3240 ± 0.0930	0.3165 ± 0.0879
CNMF	0.2542 ± 0.0667	0.5172 ± 0.0885	0.4615 ± 0.0870
p	3.1114*e*-68	1.6704*e*-173	2.7164*e*-128
Algorithm	n=100 (K=10)		
	JC	FM	F1
NNMF	0.1894 ± 0.0200	0.3701 ± 0.0487	0.3180 ± 0.0282
CNMF	0.3437 ± 0.0952	0.5209 ± 0.0983	0.5044 ± 0.1022
p	5.80668*e*-172	2.0965*e*-158	1.7230*e*-188

reasonable baseline because 1) CNMF and NNMF are both based on NMF and they both consider the network, and 2) the only difference between them is that degree distribution is not considered in NNMF. So we can tell the ability of CNMF on preserving the degree distribution through the comparison between NNMF and CNMF. The same applies to DNMF and TNMF. For quantification purposes, we use clustering evaluation metrics. The evaluation metrics of document clustering are Jaccard Coefficient (JC), Folkes&Mallows (FM) and F1 measure (F1). Given a clustering result,

- a is the number of two points that are in the same cluster of both benchmark and clustering results;

- b is the number of two points that are in the same cluster of benchmark but in different clusters of clustering results;

- c is the number of two points that are not in the same cluster of both benchmark but in the same cluster of clustering results.

and three metrics are computed by equations in Table 1 (larger means better).

We randomly generate 1000 pairs of matrices $< V, H >$ for each size (10 and 100). The comparison between the communities of re-constructed network with the communities of the original network are shown in Table 2. The statistical significance of the results are also proved using the Friedman Test[1]. The p in Table 2 denotes the p-value of the test, returned as a scalar

[1] A nonparametric test and its implementation is from Matlab.

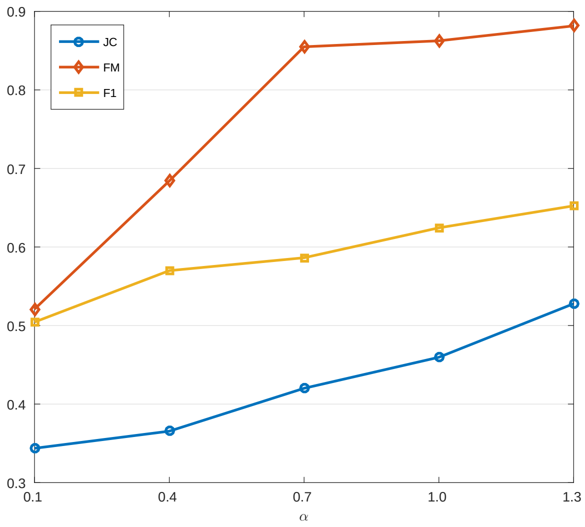

Figure 4: The change of community preserving performance from CNMF with different α.

Table 3: Comparison between NNMF and DNMF on preserving of degree distribution

Algorithm	n = 10 (K=3)	n = 100 (K=10)
NNMF	0.0658 ± 0.3314	0.0045 ± 0.0980
DNMF	0.5548 ± 0.2531	0.9079 ± 0.0190
p	9.4907e-122	1.7958e-219

value in the range $[0, 1]$, which is the probability of observing a test statistic as extreme as, or more extreme than, the observed value under the null hypothesis. The null hypothesis here is that NNMF and CNMF have the same effect on the community preserving. Since p is smaller than 0.01, it believes that the null hypothesis could be rejected. From these results, we can draw the conclusion that CNMF preserves the community structure better than NNMF. The above comparison is with $\alpha = 0.1$. Next, we have adjusted the value of α in the cost function for the community property preserving. The results are shown in Fig. 4. From both the cost function form and the results in the figure, it is known that the performance of CNMF on preserving the community property increases with the value of α.

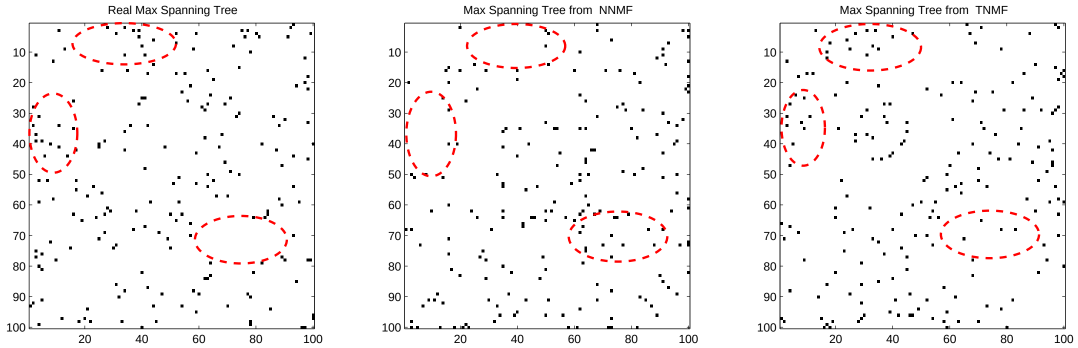

Figure 5: Comparison between NNMF and TNMF on the ability to preserve the max spanning tree. Each point in the figure denotes a link of max spanning tree of the network. The left figure denotes the real max spanning tree of the original network; the centre figure denotes the max spanning tree from NNMF; the right figure denotes the max spanning tree from TNMF. Parts of them are highlighted as an ellipse(red). We can make an approximate comparison through these three ellipses.

5.1.2. Test for degree distribution preservation

A pair of matrices are randomly generated $< V, H >$. H is the original network whose degree distribution we want to keep. After running both NNMF and DNMF, we obtain A_{NNMF} and A_{DNMF} which can re-construct the network by $H_{NNMF} = A_{NNMF}A_{NNMF}^T$ and $H_{DNMF} = A_{DNMF}A_{DNMF}^T$. To show the ability of DNMF to retain the degree distribution of the original network H, we compute the correlation between the degree distributions of the re-constructed networks (H_{NNMF} and H_{DNMF}) and the original network H. If the degree distribution of H_{DNMF} has larger correlation coefficient with the degree distribution of H than the degree distribution of H_{NNMF}, we can draw the conclusion that DNMF preserves degree distribution better than H_{NNMF}. Here, we consider two matrix sizes: 10 and 100. For each size, we randomly generate 1000 pairs of matrices. The number of hidden factors is set as 3 and 10 and $\alpha = 1$. The average and standard deviation of results are given in Table 3. The p in Table 3 denotes the p-value of the Friedman test, and the null hypothesis here is that NNMF and DNMF have the same effect on the degree distribution preserving. These numbers show that the DNMF preserves more of the network degree distribution than NNMF.

Table 4: Comparison between NNMF and TNMF on preserving max spanning tree

Algorithm	n = 10 (max = 9)	n = 100 (max = 99)
NNMF	2.5 ± 1.3	21.4 ± 6.3
TNMF	6.2 ± 0.7	56.1 ± 7.2
p	7.2787e-211	1.7958e-219

5.1.3. Test for max spanning tree preservation

For the max spanning tree, we first randomly generate a pair of $V_{100\times 100}$ and $H_{100\times 100}$. Using NNMF and TNMF to factorize V and H obtain two A_{NNMF} and A_{TNMF} which can re-construct the network H by $H_{NNMF} = A_{NNMF}A_{NNMF}^T$ and $H_{TNMF} = A_{TNMF}A_{TNMF}^T$. After extracting max spanning trees of three networks H, H_{NNMF} and H_{TNMF}, we can compare the similarity between the re-constructed trees (T_{NNMF} and T_{TNMF}) and the benchmark T_H. An illustrative example is shown in Fig. 5, in which each point in the matrix denotes an edge between two nodes. To quantify the ability of TNMF to retain the tree structure, we randomly generated 1000 pairs of V and H for each size: 10 and 100. For each pair $< V, H >$, we run both NNMF and TNMF, and compute the similarity between the re-constructed trees with the benchmark tree by the number of overlap edges as

$$s(T_H, T_{NNMF}) = sum(T_H \circledast T_{NNMF}) \tag{31}$$

where $sum(M)$ is to count the number of 1 in matrix M. The number of hidden factors are set as 3 and 10 and $\alpha = 1$. If the re-constructed max spanning tree from TNMF has more overlap edges with the original max spanning tree than the tree from NNMF, we draw the conclusion that TNMF is able to preserve the max spanning tree better than NNMF. The results are shown in Table 4. The *max* in the table denotes the maximum number of edges in the max spanning tree. For a network with n nodes, the maximum number of edges in the max spanning tree is $n-1$. The p in Table 4 denotes the p-value of the Friedman test, and the null hypothesis here is that NNMF and TNMF have the same effect on the max spanning tree preserving.

5.2. Experiments on real-world data

The above section shows the abilities of the proposed algorithms to preserve the desired network structural properties. In this section, we will compare the efficiency of the proposed algorithms for real-world applications, including document clustering and recommendation.

Table 5: Statistics of *Cora* dataset

Number of documents	2,708
Number of keywords	1,433
Number of links	5,429
Number of classes	7

5.2.1. Document clustering with one-layer network

Our document data is *Cora*[2], which is a public dataset and consists of 2708 scientific publications classified into one of seven classes. A multilayer network is composed by a horizontal network and a vertical network. The horizontal network $H_{2708\times2708}$ is the citation relations between publications and it consists of 5429 links. This vertical network $V_{2708\times1433}$ is constructed by the mapping relation between documents and words. The detailed statistics are shown in Table 5.

After the embedding of $V_{2708\times1433}$ and $H_{2708\times2708}$ through our proposed algorithms, NNMF, CNMF, DNMF and TNMF, the documents are projected into a latent space and then given new representations. It is believed that the different classes are formed as a result of the intrinsics of the documents, so if the discovered latent space is good enough, it will cause the documents to cluster into these seven classes. According to this idea, we conduct the document clustering through the learned matrix A (the new representations of documents) by the k-means clustering algorithm[3]. To compare the efficiency, we also implement standard NMF which does not consider the horizontal network $H_{2708\times2708}$ and Relational Topic Model (RTM)[48] which is a successful Bayesian model for the relational data and considers both horizontal network (document-word relationships) and vertical network (document network). The final results are shown in Fig. 6. Three subfigures denote three clustering result comparisons by the metric in Table 1. We have tested four numbers of factors: $K = 100$, $K = 300$, $K = 500$ and $K = 1000$. In each subfigure in Fig. 6, we have compared the results from NNMF, CNMF, DNMF, TNMF, NMF and RTM on the clustering evaluation metric. Except for NMF, the algorithms all consider the effects from the horizontal net-

[2] http://linqs.cs.umd.edu/projects/projects/lbc/

[3] Since k-means is not very stable, the reported results in this work are the average of ten independent runs.

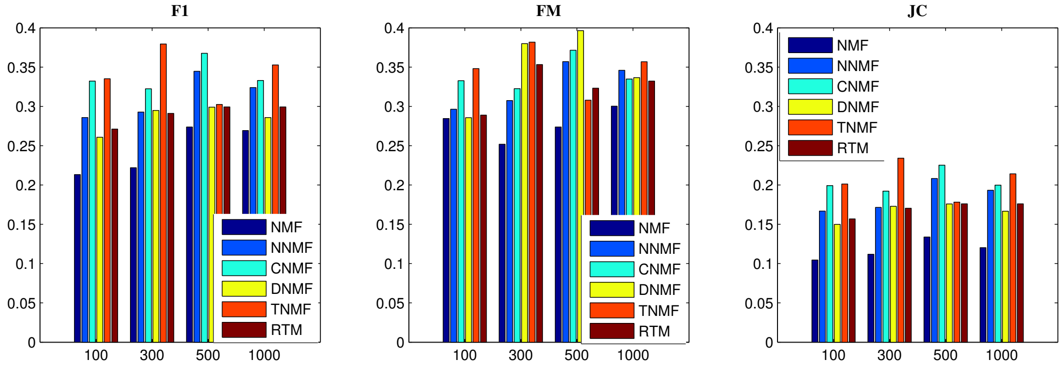

Figure 6: Comparison of document clustering on different values of K. The influence of K values with $\alpha = 0.1$.($K = 100$, $K = 300$, $K = 500$, and $K = 1000$)

work $H_{2708\times2708}$. From this result, we can see that NMF achieves the worst performance compared to others. Thus, we can draw the conclusion that incorporating the citation network is helpful for the clustering of the publications. Except for DNMF, which has similar performance to RTM, NNMF, CNMF and TNMF are better than RTM on this document clustering task. Notably, CNMF and TNMF achieve the best performance of all the algorithms. The reason is that the community structure of CNMF is beneficial for the clustering because it encourages 'similar' nodes to cluster together. For example, two documents d_i and d_j are in the same community in network H due to their 'similarity'. Retaining the community structure will make it more possible for d_i and d_j to remain within the same cluster under the new representations. In the TNMF, preserving max spanning tree encourages the most important relations of all the nodes/documents. These relations in the tree can be seen as the 'bones' of a network, which determine the weighted distances between the nodes (documents). Therefore, the TNMF can benefit for the document clustering. Each document will exhibit two natures from two networks: H and V. If the two natures are consistent, the constraint from V will help to enhance the learned network structure from H; If the two natures are not consistent, there will be a contradiction between H and V, which will prevent the network structure learning from H.

Table 6: Statistics of *Lastfm* dataset

number of users	$1,892$
number of artists	$17,632$
number of friend relations	$12,717$
test user-artist pairs	$5,000$

5.2.2. Recommendation with one-layer network

A public dataset *Lastfm*[4] is adopted which is a commonly used dataset for evaluating algorithms for recommender systems. This dataset contains music artist listening information from a set of users from Last.fm online music system and a social network between these users. Each entry of this dataset denotes the number of a user listening the songs of an artist. For this dataset, a vertical network $V_{1892\times17632}$ is formed by users and artists, and V_{ij} represents the count of user i listening artist j. The horizontal network $H_{1892\times1892}$ is the user friend network. The statistics are shown in Table 6. The recommendation task for this online music system is to recommend the artists to the users. After a certain number of user-artist pairs are retained as the test data, the algorithms are evaluated by predicting the counts of users listening to these artists which are expressions of the users' interests on the artists.

The evaluation metric is Mean Absolute Error (MAE) which is the simplest and the most intuitive. The definition is

$$MAE = \frac{1}{N}\sum_{u,i}|\widehat{r}_{u,i} - r_{u,i}|, \tag{32}$$

where N is the number of test user-artist parts, $r_{u,i}$ is the real count, and $\widehat{r}_{u,i}$ is the predicted count. The correlation coefficient is computed by

$$\rho = \frac{\sum_i (r_i - \bar{r})(\widehat{r}_i - \bar{\widehat{r}})}{\sqrt{\sum_i (r_i - \bar{r})^2 \sum_i (\widehat{r}_i - \bar{\widehat{r}})^2}}. \tag{33}$$

To show the performance of the proposed algorithms on the recommendation, we compared them with two state-of-the-art methods: Graph regularized NMF (GNMF) [32, 47] and Trust Semantic Fusion (TSF) [49]. Note that

[4] http://labrosa.ee.columbia.edu/millionsong/lastfm

TSF is a method based on Collaborative Filtering rather than embedding. The results are shown in Fig. 7. The standard NMF (without the horizontal network $H_{1892\times1892}$) has the worst performance of all the algorithms, which is similar to the document clustering in Section 5.2.1. We can therefore draw the conclusion that incorporating user friend network $H_{1892\times1892}$ improves the performance of the recommendation. GNMF has the same property as CNMF for preserving the community structure of $H_{1892\times1892}$, as discussed in Section 4.2. The results in Fig. 7 also show that GNMF has similar performance to CNMF. The performance of TSF is a little better than the standard NMF, so the strategy of TSF to combine the user social network with the collaborative filtering is successful. However, the performance of TSF is generally worse than other algorithms. Although DNMF has good performance on MAE, the correlation with DNMF is the worst of all the proposed algorithms and the state-of-the-art GNMF. This reflects that DNMF predicts accurate values for some test data but not all data. The reason is that DNMF preserves the network degrees of all the nodes/users. The nodes with relatively large degrees will tend to have large weights in the weighted degree distribution, and DNMF will have a bias toward these nodes during embedding. Therefore, DNMF tends to achieve good results for the nodes/users with large degrees. In all the algorithms, TNMF achieves the best performance both on MAE and Correlation. As discussed in Section 5.2.1, preserving max spanning tree encourages the most important relations (the 'bones' of a network) of all the nodes/users. These relations reflect not only the 'distances' between nodes but also the degrees of nodes.

5.2.3. Document clustering with two-layer network

First, we introduce the dataset we use. Documents are a collection of papers from *CiteSeer*[50]. There are 3312 papers in the whole corpus. Each paper is represented by a binary vector using words. The labels of these papers are set as their research areas, such as AI (Artificial Intelligence), ML (Machine Learning), Agents, DB (Database), IR (Information Retrieval) and HCI (Human-Computer Interaction). The statistics are shown in Table 7.

A two-layer network is composed by two horizontal networks $H_{3312\times3312}$ and $H_{3703\times3703}$ and one vertical network $V_{3312\times3703}$, and they are constructed as following. The first layer horizontal network $H_{3312\times3312}$ is a paper citation network, which is formed by the citation relations between papers. The second layer horizontal network $H_{3703\times3703}$ is the keyword concurrence network, which is formed by the concurrence relations between keywords. The vertical

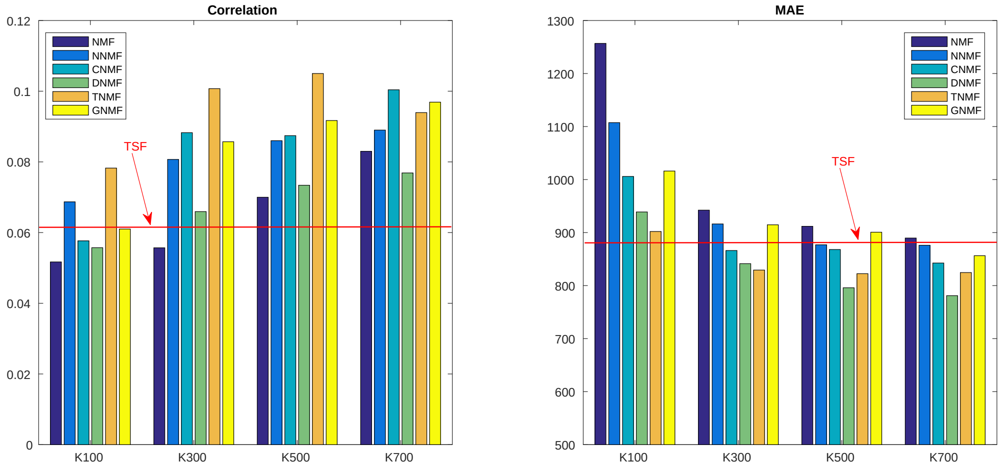

Figure 7: Comparison of recommendation performances of different algorithms with different values of K ($K = 100$, $K = 300$, $K = 500$, and $K = 700$) and $\alpha = 10$. Note that there is a line in each figure denoting the value from TSF because TSF does not need to predefine the value of K.

Table 7: Statistics of *CiteSeer* dataset

Number of documents	3,312
Number of keywords	3,703
Number of classes	6

network $V_{3312\times3703}$ is the mapping relation between documents and keywords (there will be a link between a keyword and a document if this keyword shows in this document).

It should be noted that we keep only one type of structure for two horizontal networks. For example, CNMF only keeps the community property of both networks, hence Eq. (16) only adds $\alpha \cdot \frac{1}{2}\|P_X - X\|_F^2$. For brevity, the same coefficient is used for both networks.

Here, we compare the performances between the one-layer network $H_{3312\times3312}$ and the two-layer network $H_{3312\times3312}$ and $H_{3703\times3703}$. The comparisons on three evaluation metrics are shown in Fig. 8. Except CNMF with $K = 300$, two-layer network outweighs the one-layer network. It means that incorporating the keyword co-occurrent network can improve the document clustering task with only the document citation network.

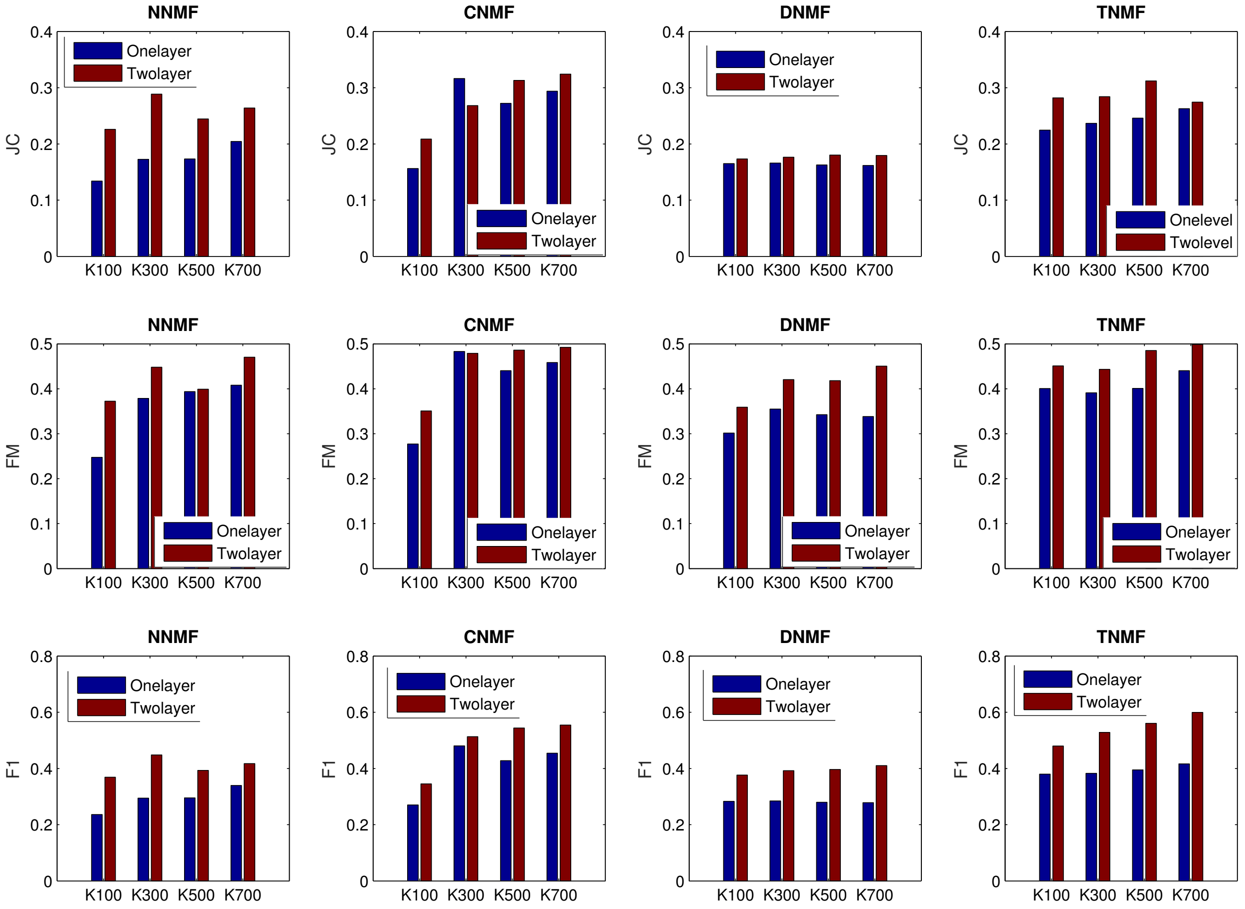

Figure 8: Comparison of the influence from one-layer network (with one horizontal network and one vertical network) and two-layer network (with two horizontal networks and one vertical network) on document clustering with $\alpha = 0.1$ and $K = 100$, $K = 300$, $K = 500$ and $K = 700$.

Table 8: Statistics of *Delicious* dataset

number of users	$1,867$
number of URLs	$5,633$
test user-URL pairs	$2,000$

5.2.4. Recommendation with two-layer network

The dataset used for recommendation with two-layer network is *Delicious*. This dataset is collected from the Delicious website[5], which records the bookmark options of users on webpages/URLs in this website according to users' interests [51]. We filter this dataset by keeping the URLs that are marked by at least three users, and the number of URL is 5633. The links between URLs are generated by their tags. In the Delicious website, each URL will get tags from users, and these tags will give an hint for the content/semantics of this URL. We use these tags to compute the content similarity between URLs with tag-vector representations through cosine similarity metric. The statistics of the dataset are shown in Table 8. In this dataset, vertical network $V_{1867\times5633}$ is formed by users and URLs. The horizontal networks are: the user friend network $H_{1867\times1867}$ and the content similarity network $H_{5633\times5633}$. A certain number of user-URL pairs are kept to be the test data. The evaluation metrics are in Eq. (32) and Eq. (33). As shown in Fig. 9, the two-layer network still outweighs the one-layer network.

Despite the good performances on both document clustering and recommendation tasks, we still want to give an advice that it should be very careful to combine the different networks: vertical network and different horizontal networks. For example, suppose there are a user-movie vertical network and a movie network for the recommendation task, and the movie network is formed by the distance between the releasing dates of the movies. We know that the releasing date of a movie will not highly impact on the rating from users. In this situation, the combination of two networks may not improve the rating prediction due to the 'noise' from the movie network. Since the information in the movie network will not help the prediction of the ratings, the constraint from movie network will decrease the learned information from the user-movie vertical network. However, if the formation of the movie network is replaced by another strategy: the movies with similar director and

[5] http://www.delicious.com

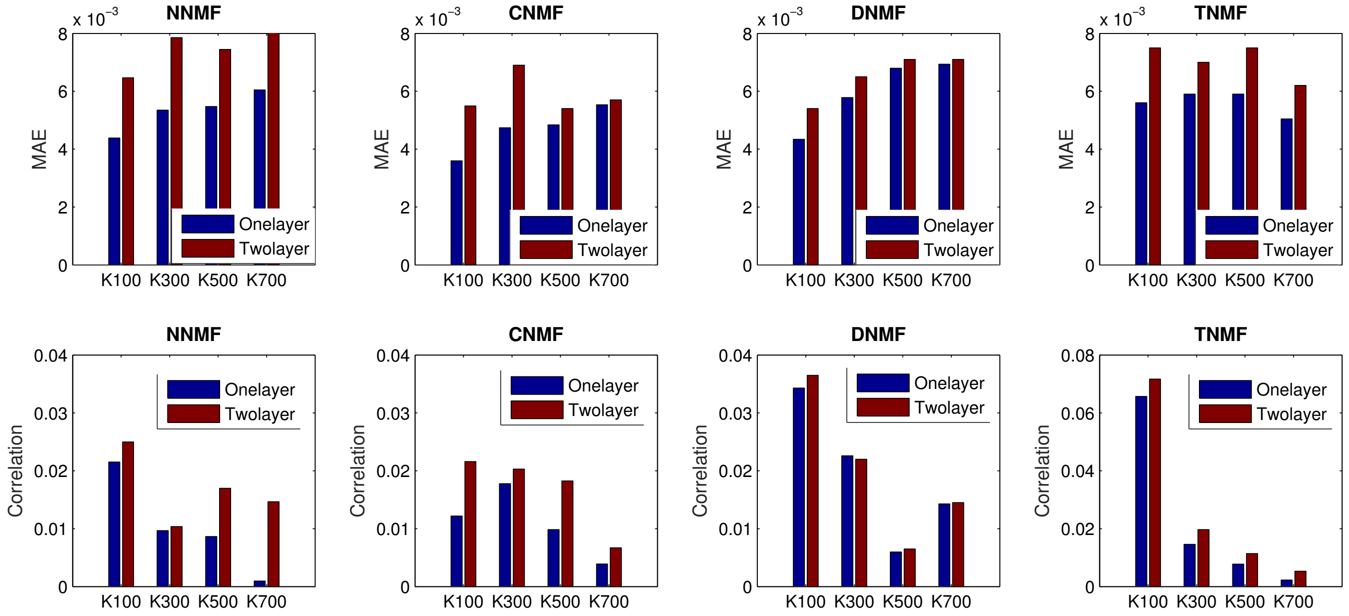

Figure 9: Comparison of the influence from one-layer network (with one horizontal network and one vertical network) and two-layer network (with two horizontal networks and one vertical network) on recommendation with $\alpha = 10$ and $K = 100$, $K = 300$, $K = 500$ and $K = 700$.

actors tend to have links, this new movie network will help to predict the user ratings on movies different from the old movie network. In this new situation, the combination of the movie network with user-movie network will help to improve the rating prediction. Therefore, the consistence between the different networks is important and should be considered before using the multilayer network embedding.

6. Conclusion and future study

In this paper, we have proposed four multilayer network embedding algorithms with four different structural property constraints based on nonnegative matrix factorization. The network structural constraints have been incorporated into the cost function of a traditional nonnegative matrix factorization-based embedding algorithm. The optimization of the new cost function constructs a new latent space and projects all nodes in the various layers into this new latent space. At the same time, the projected nodes will retain the original network structure as far as possible. Four algorithms have been carefully designed to find the optimal latent spaces. Lastly, experiments on synthetic and real-world data show that our algorithms are able to preserve the desired structural properties and can be used for recommender systems and document clustering.

A number of interesting study points remain; for example, the dimension number of the discovered latent space needs to be predefined in the present algorithms. The ability to automatically find an optimized number for the shared latent space will make multilayer network embedding more practical for real-world tasks. We will consider using Bayesian nonparametric learning [52] to resolve this problem in the future. Meanwhile, conflict may exist between networks at different layers, so directly embedding this kind of multilayer network would obtain worse results. How to detect, measure and avoid conflicts between different networks is therefore significant for secure multilayer network embedding, and we will also consider this point in our future study.

Acknowledgements

Research work reported in this paper was partly supported by the Australian Research Council (ARC) under discovery grant DP150101645 and the China Scholarship Council.

Appendix

6.1. Proof of Theorem 1

Proof. According to Eq. (7), give the objective function with respect to A as

$$F(A) = tr\Big(- VX^TA^T + \frac{1}{2}AXX^TA^T + \frac{1}{2}\alpha \cdot AA^T - \alpha \cdot P^+A^T + \alpha \cdot P^-A^T\Big) \tag{34}$$

and define

$$\begin{aligned} G(A, A^t) = & \sum_{ij} -(VX^T)_{ij}A^t_{ij}(1 + \log \frac{A_{ij}}{A^t_{ij}}) + \sum_{ij} \frac{(A^tXX^T)_{ij}A^2_{ij}}{2A^t_{ij}} + \sum_{ij} \alpha \frac{(A^t_{ij})A^2_{ij}}{2A^t_{ij}} \\ & - \sum_{ij} \alpha(P^+)_{ij}A^t_{ij}(1 + \log \frac{A_{ij}}{A^t_{ij}}) + \sum_{ij} \alpha(P^-)_{ij} \frac{A^2_{ij} + (A^t_{ij})^2}{A^t_{ij}}. \end{aligned} \tag{35}$$

Then, $G(A, A^t)$ is the auxiliary function of $F(A)$, because the conditions

$$\begin{aligned} F(A^t) &= G(A^t, A^t) \\ G(A^t, A^t) &\geq G(A^{t+1}, A^t) \\ G(A^{t+1}, A^t) &\geq F(A^{t+1}) \end{aligned} \tag{36}$$

are satisfied when A^{t+1} takes the minimum value of $G(A, A^t)$ with respect to A_{ij}. The derivative of $G(A, A^t)$ is

$$\begin{aligned} \frac{\partial G(A, A^t)}{\partial A_{ij}} =& -\frac{(VX^T)_{ij}A^t_{ij}}{A_{ij}} + \frac{(A^tXX^T)_{ij}A_{ij}}{A^t_{ij}} \\ &+ \alpha\frac{A^t_{ij}A_{ij}}{A^t_{ij}} + \alpha\frac{(P^-_{ij})A_{ij}}{A^t_{ij}} - \alpha\frac{P^+_{ij}A^t_{ij}}{A_{ij}} \end{aligned} \tag{37}$$

and set A^{t+1} as the minimal value of $G(A, A^t)$ through $\frac{\partial G(A,A^t)}{\partial A_{ij}} = 0$. We have

$$A^{t+1}_{ij} = A^t_{ij}\sqrt{\frac{\left[VX^T + \alpha P^+\right]_{ij}}{\left[AXX^T + \alpha A + \alpha P^-\right]_{ij}}} \tag{38}$$

Therefore, we know that the update of A according to update rule (7) will lead to the non-increasing of $J(A, X)$.

□

6.2. Proof of Theorem 3

Proof. According to Eq. (22), give the objective function with respect to A as

$$F(A) = tr\Big(- YX^TA^T + \frac{1}{2}AXX^TA^T - \alpha \cdot H\mathbf{1}^{n\times n}AA^T + \frac{1}{2}\alpha \cdot AA^T\mathbf{1}^{n\times n}AA^T\Big) \tag{39}$$

and define

$$\begin{aligned} G(A, A^t) =& -\sum_{ij}\left((VX^T)_{ij}A^t_{ij}(1 + \log\frac{A_{ij}}{A^t_{ij}})\right) + \frac{1}{2}\sum_{ij}\left(\frac{(A^tXX^T)_{ij}A^2_{ij}}{A^t_{ij}}\right) \\ &- \sum_{ij}\left(\alpha \cdot (H\mathbf{1}^{n\times n}A^t)_{ij}A^t_{ij}(1 + \log\frac{A_{ij}}{A^t_{ij}})\right) + \frac{1}{2}\sum_{ij}\left(\alpha \cdot \frac{(A^t(A^t)^T\mathbf{1}^{n\times n}A^t)_{ij}A^4_{ij}}{(A^t_{ij})^3}\right) \end{aligned} \tag{40}$$

Then, $G(A, A^t)$ is the auxiliary function of $F(A)$, because Eq. (36) is satisfied. The derivative of $G(A, A^t)$ with respect to A is

$$\begin{aligned}\frac{\partial G(A,A^t)}{\partial A_{ij}} = &-\frac{(VX^T)_{ij}A^t_{ij}}{A_{ij}} + \frac{(A^tXX^T)_{ij}A_{ij}}{A^t_{ij}} \\ &-\alpha\cdot\frac{((H\mathbf{1}^{n\times n})A^t)_{ij}A^t_{ij}}{A_{ij}} + 2\alpha\cdot\frac{(AA^T\mathbf{1}^{n\times 1}\mathbf{1}^{1\times n}A)_{ij}A^3_{ij}}{(A^t_{ij})^3}\end{aligned} \tag{41}$$

and set A^{t+1} through $\frac{\partial G(A,A^t)}{\partial A_{ij}} = 0$, and then the update rule (24) is derived. □

6.3. Proof of Theorem 4

Proof. According to Eq. (27), give the objective function with respect to A

$$F(A) = tr\Big(- VX^TA^T + \frac{1}{2}AXX^TA^T + \frac{1}{4}\alpha\cdot\big((\overline{T}-T)\circledast AA^TAA^T\Big) \tag{42}$$

and define

$$\begin{aligned}G(A,A^t) = &-\sum_{ij}\left((VX^T)_{ij}A^t_{ij}(1+\log\frac{A_{ij}}{A^t_{ij}})\right) \\ &+\frac{1}{2}\sum_{ij}\left(\frac{(A^tXX^T)_{ij}A^2_{ij}}{(A^t_{ij})}\right) \\ &-\frac{1}{4}\alpha\sum_{ijkl}\left(T_{il}A^t_{il}A^t_{kl}A^t_{ik}A^t_{ij}\left(1+\log\frac{A_{il}A_{kl}A_{ik}A_{ij}}{A^t_{il}A^t_{kl}A^t_{ik}A^t_{ij}}\right)\right) \\ &+\frac{1}{4}\alpha\sum_{ij}\left(\frac{(\overline{T}\circledast(A^t(A^t)^T)A^t)_{ij}(A_{ij})^4}{(A^t_{ij})^3}\right).\end{aligned} \tag{43}$$

Then, $G(A, A^t)$ is the auxiliary function of $F(A)$, because Eq. (36) is satisfied. The derivative of $G(A, A^t)$ with respect to A is

$$\begin{aligned}\frac{\partial G(A,A^t)}{\partial A_{ij}} = &-\frac{(VX^T)_{ij}A^t_{ij}}{A_{ij}} + \frac{(A^tXX^T)_{ij}A_{ij}}{(A^t_{ij})} \\ &-\alpha\frac{(T\circledast(A^t(A^t)^T)A^t)_{ij}A^t_{ij}}{A_{ij}} + \alpha\frac{(\overline{T}\circledast(A^t(A^t)^T)A^t)_{ij}A^3_{ij}}{(A^t_{ij})^3}\end{aligned} \tag{44}$$

and set A_{ij}^{t+1} as the minimal value of $G(A, A^t)$ through $\frac{\partial G(A,A^t)}{\partial A_{ij}} = 0$. Then, we obtain update rule (29). Therefore, we know that the update of A according to (29) will lead to the non-increasing of $J(A, X)$.

□

References

[1] M. Kivel, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, M. A. Porter, Multilayer networks, Journal of Complex Networks 2 (3) (2014) 203–271.

[2] S. Boccaletti, G. Bianconi, R. Criado, C. del Genio, J. Gmez-Gardees, M. Romance, I. Sendia-Nadal, Z. Wang, M. Zanin, The structure and dynamics of multilayer networks, Physics Reports 544 (1) (2014) 1 – 122.

[3] M. De Domenico, A. Solé-Ribalta, E. Cozzo, M. Kivelä, Y. Moreno, M. A. Porter, S. Gómez, A. Arenas, Mathematical formulation of multilayer networks, Physical Review X 3 (2013) 041022.

[4] P. Hitzler, K. Janowicz, Linked data, big data, and the 4th paradigm, Semantic Web 4 (3) (2013) 233–235.

[5] S. Zhu, K. Yu, Y. Chi, Y. Gong, Combining content and link for classification using matrix factorization, in: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '07, ACM, New York, NY, USA, 2007, pp. 487–494.

[6] B. Shaw, T. Jebara, Structure preserving embedding, in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, ACM, New York, NY, USA, 2009, pp. 937–944.

[7] 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, KDD '14, ACM, New York, NY, USA, 2014, pp. 701–710.

[8] A. Grover, J. Leskovec, Node2vec: Scalable feature learning for networks, in: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, ACM, New York, NY, USA, 2016, pp. 855–864.

[9] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in: Proceedings of the 24th International Conference on World Wide Web, WWW '15, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 2015, pp. 1067–1077.

[10] Z. Yuan, J. Sang, Y. Liu, C. Xu, Latent feature learning in social media network, in: Proceedings of the 21st ACM International Conference on Multimedia, MM '13, ACM, New York, NY, USA, 2013, pp. 253–262.

[11] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, T. S. Huang, Heterogeneous network embedding via deep architectures, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15, ACM, New York, NY, USA, 2015, pp. 119–128.

[12] Y.-X. Wang, Y.-J. Zhang, Nonnegative matrix factorization: a comprehensive review, IEEE Transactions on Knowledge and Data Engineering 25 (6) (2013) 1336–1353.

[13] D. J. Watts, S. H. Strogatz, Collective dynamics of small-world networks, Nature 393 (6684) (1998) 440–442.

[14] A.-L. Barabsi, R. Albert, Emergence of scaling in random networks, Science 286 (5439) (1999) 509–512.

[15] M. E. Newman, The structure and function of complex networks, SIAM Review 45 (2) (2003) 167–256.

[16] E. Estrada, The structure of complex networks: theory and applications, Oxford University Press, 2011.

[17] T. Narayanan, S. Subramaniam, A newtonian framework for community detection in undirected biological networks, IEEE Transactions on Biomedical Circuits and Systems 8 (1) (2014) 65–73.

[18] J. Chen, Y. Saad, Dense subgraph extraction with application to community detection, IEEE Transactions on Knowledge and Data Engineering 24 (7) (2012) 1216–1230.

[19] M. W. Hadley, M. F. McGranaghan, A. Willey, C. W. Liew, E. R. Reynolds, A new measure based on degree distribution that links information theory and network graph analysis, Neural Systems & Circuits 2 (1).

[20] D. Centola, The spread of behavior in an online social network experiment, Science 329 (5996) (2010) 1194–1197.

[21] L. A. Adamic, B. A. Huberman, Power-law distribution of the world wide web, Science 287 (5461) (2000) 2115.

[22] F. Meyer, L. Najman, Segmentation, minimum spanning tree and hierarchies, John Wiley & Sons, Inc., 2013, pp. 229–261.

[23] W. B. March, P. Ram, A. G. Gray, Fast euclidean minimum spanning tree: algorithm, analysis, and applications, in: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD10), ACM, Washington, DC, USA, 2010, pp. 603–612.

[24] C. Li, J. Luo, J. Huang, J. Fan, Multi-layer network for influence propagation over microblog, in: M. Chau, G. Wang, W. Yue, H. Chen (Eds.), Intelligence and Security Informatics, Vol. 7299 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 60–72.

[25] M. Pasqual, O. de Weck, Multilayer network model for analysis and management of change propagation, Research in Engineering Design 23 (4) (2012) 305–328.

[26] C. Ding, T. Li, M. Jordan, Convex and semi-nonnegative matrix factorizations, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (1) (2010) 45–55.

[27] P. O. Hoyer, Non-negative matrix factorization with sparseness constraints, The Journal of Machine Learning Research 5 (2004) 1457–1469.

[28] W. Ren, G. Li, D. Tu, L. Jia, Nonnegative matrix factorization with regularizations, IEEE Journal on Emerging and Selected Topics in Circuits and Systems 4 (1) (2014) 153–164.

[29] Z. Li, X. Wu, H. Peng, Nonnegative matrix factorization on orthogonal subspace, Pattern Recognition Letters 31 (9) (2010) 905 – 911.

[30] H. Liu, Z. Wu, X. Li, D. Cai, T. Huang, Constrained nonnegative matrix factorization for image representation, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (7) (2012) 1299–1311.

[31] Y. Chen, M. Rege, M. Dong, J. Hua, Non-negative matrix factorization for semi-supervised data clustering, Knowledge and Information Systems 17 (3) (2008) 355–379.

[32] D. Cai, X. He, J. Han, T. Huang, Graph regularized nonnegative matrix factorization for data representation, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (8) (2011) 1548–1560.

[33] R. Zhi, M. Flierl, Q. Ruan, W. Kleijn, Graph-preserving sparse nonnegative matrix factorization with application to facial expression recognition, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 41 (1) (2011) 38–52.

[34] P. Cui, F. Wang, S. Liu, M. Ou, S. Yang, L. Sun, Who should share what?: Item-level social influence prediction for users and posts ranking, in: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '11, ACM, New York, NY, USA, 2011, pp. 185–194.

[35] W. Cheng, X. Zhang, Z. Guo, Y. Wu, P. F. Sullivan, W. Wang, Flexible and robust co-regularized multi-domain graph clustering, in: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '13, ACM, New York, NY, USA, 2013, pp. 320–328.

[36] Y.-R. Lin, J. Sun, H. Sundaram, A. Kelliher, P. Castro, R. Konuru, Community discovery via metagraph factorization, ACM Transactions on Knowledge Discovery from Data 5 (3) (2011) 17:1–17:44.

[37] F. Wang, T. Li, X. Wang, S. Zhu, C. Ding, Community discovery using nonnegative matrix factorization, Data Mining and Knowledge Discovery 22 (3) (2011) 493–521.

[38] D. D. Lee, H. S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (6755) (1999) 788–791.

[39] S. Zhang, W. Wang, J. Ford, F. Makedon, Learning from incomplete ratings using non-negative matrix factorization, in: SDM, Vol. 6, SIAM, 2006, pp. 548–552.

[40] Y. Xu, W. Yin, A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion, SIAM J. Imaging Sciences 6 (3) (2013) 1758–1789.

[41] C.-J. Lin, Projected gradient methods for nonnegative matrix factorization, Neural Computation 19 (10) (2007) 2756–2779.

[42] C.-J. Lin, On the convergence of multiplicative update algorithms for nonnegative matrix factorization, IEEE Transactions on Neural Networks 18 (6) (2007) 1589–1596.

[43] D. D. Lee, H. S. Seung, Algorithms for non-negative matrix factorization, in: T. Leen, T. Dietterich, V. Tresp (Eds.), Advances in Neural Information Processing Systems (NIPS), MIT Press, 2001, pp. 556–562.

[44] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, D.-U. Hwang, Complex networks: Structure and dynamics, Physics Reports 424 (45) (2006) 175 – 308.

[45] U. Von Luxburg, A tutorial on spectral clustering, Statistics and Computing 17 (4) (2007) 395–416.

[46] T. Jin, J. Yu, J. You, K. Zeng, C. Li, Z. Yu, Low-rank matrix factorization with multiple hypergraph regularizer, Pattern Recognition 48 (3) (2015) 1011 – 1022.

[47] Q. Gu, J. Zhou, C. H. Ding, Collaborative filtering: weighted nonnegative matrix factorization incorporating user and item graphs., in: SIAM International Conference on Data Mining (SDM10), SIAM, Columbus, Ohio, USA, 2010, pp. 199–210.

[48] J. Chang, D. M. Blei, Relational topic models for document networks, in: International Conference on Artificial Intelligence and Statistics, 2009, pp. 81–88.

[49] Q. Shambour, J. Lu, A trust-semantic fusion-based recommendation approach for e-business applications, Decision Support Systems 54 (1) (2012) 768 – 780.

[50] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, T. Eliassi-Rad, Collective classification in network data, AI Magazine 29 (3) (2008) 93–106.

[51] I. Cantador, P. Brusilovsky, T. Kuflik, 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011), in: Proceedings of the 5th ACM conference on Recommender systems, RecSys 2011, ACM, New York, NY, USA, 2011.

[52] J. Xuan, J. Lu, G. Zhang, R. Y. D. Xu, X. Luo, Doubly nonparametric sparse nonnegative matrix factorization based on dependent indian buffet processes, IEEE Transactions on Neural Networks and Learning Systems PP (99) (2017) 1–15. doi:10.1109/TNNLS.2017.2676817.