Overlapping Community Detection via Bounded Nonnegative Matrix Tri-Factorization

Yu Zhang and Dit-Yan Yeung
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
{zhangyu, dyyeung}@cse.ust.hk

ABSTRACT

Complex networks are ubiquitous in our daily life, with the World Wide Web, social networks, and academic citation networks being some of the common examples. It is well understood that modeling and understanding the network structure is of crucial importance to revealing the network functions. One important problem, known as community detection, is to detect and extract the community structure of networks. More recently, the focus in this research topic has been switched to the detection of overlapping communities. In this paper, based on the matrix factorization approach, we propose a method called bounded nonnegative matrix tri-factorization (BNMTF). Using three factors in the factorization, we can explicitly model and learn the community membership of each node as well as the interaction among communities. Based on a unified formulation for both directed and undirected networks, the optimization problem underlying BNMTF can use either the squared loss or the generalized KL-divergence as its loss function. In addition, to address the sparsity problem as a result of missing edges, we also propose another setting in which the loss function is defined only on the observed edges. We report some experiments on real-world datasets to demonstrate the superiority of BNMTF over other related matrix factorization methods.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; H.2.8 [**Database Management**]: Database Applications—*Data mining*

General Terms

Algorithms

Keywords

Network Analysis, Community Detection, NMF, BNMTF

KDD'12, August 12–16, 2012, Beijing, China.

1. INTRODUCTION

Complex networks are ubiquitous in our daily life. For example, the World Wide Web, social networks, and academic citation networks are some of the commonly encountered ones. In these networks, links or edges connecting entities represent relations between them, such as hyperlinks between webpages, friend relations between people, and citations in academic publications. It is well understood that modeling and understanding the network structure is of crucial importance to revealing the network functions. One important problem in understanding the structure of a network is to detect and extract its modular structure consisting of communities, which is often known as community detection [7]. Although the problem is intuitively easy to understand, what constitutes a community in a network does not yet have a well-accepted definition. In this paper, similar to the notion used in [7], we regard communities in a network as densely connected subsets of vertices with a relatively high ratio of the number of intra-community edges to the number of inter-community edges.

Over the past decade, research on community detection has mostly adopted the assumption that each node or vertex in a network belongs to one and only one community. We refer to this as the non-overlapping community detection problem [7, 13], which is similar in many ways to clustering in a lot of data mining applications. Some of the popular community detection methods in this category make use of a quality measure called modularity [4, 2] to evaluate the quality of the community structure found, possibly by directly maximizing an objective function which is based on the modularity measure. This assumption is apparently too restrictive. In many real-world networks such as social networks, it is not uncommon to find entities belonging to multiple groups or communities. In other words, the communities are not disjoint but do overlap. By relaxing the assumption, the community detection problem becomes more general and we refer to it as the overlapping community detection problem. The first method for this problem, called the clique percolation method (CPM), was proposed in [16]. CPM is based on the assumption that each community is a union of adjacent k-cliques. It is an influential method with many extensions proposed subsequently, e.g., [6, 17]. More recently, several methods have been developed based on matrix factorization. For example, nonnegative matrix factorization (NMF) [10] has been applied to develop some methods which deliver promising performance [18, 19].

Even though the NMF-based methods in [18, 19] exhibit good performance on some overlapping community detection applications, they do have drawbacks. For example, the

method in [18] uses the conventional NMF model and hence the physical meaning of the parameters is not very clear for the overlapping community detection problem, and the method in [19] has to use different formulations for directed and undirected networks. Moreover, methods in [18, 19] only consider one loss function, i.e., generalized KL-divergence in [18] and squared loss in [19], but in applications we still have no idea which loss function is a more suitable choice for the task at hand. In this paper, our goal is to overcome these drawbacks within the matrix factorization approach. We propose a method called bounded nonnegative matrix tri-factorization (BNMTF). Specifically, from the matrix factorization perspective, we use tri-factorization, i.e., three-factor factorization, in the form $\mathbf{UBU}^T$, where $\mathbf{U}$ represents the membership of each node in each community and $\mathbf{B}$ represents the interaction among all communities. Each entry in $\mathbf{U}$ corresponds to the probability that a node belongs to a community and hence its value is restricted to the range $[0, 1]$, i.e., $\mathbf{U}$ is bounded. Obviously, the physical meaning of $\mathbf{U}$ and $\mathbf{B}$ is very clear. We also note that the same tri-factorization form can be used for both directed and undirected networks. The only difference lies in $\mathbf{B}$, which is required to be symmetric for undirected networks. Unlike the previous methods [18, 19], we consider both the squared loss and generalized KL-divergence as loss functions. Moreover, we note that there is intrinsic ambiguity when an edge is missing between two vertices. While the two vertices involved may indeed be unrelated, it is also possible that the relation between them has not been observed. In the same spirit as many methods for collaborative filtering, we also consider another setting in which the loss function is defined on the edges with nonzero weights only. We will report some experiments on real-world datasets to demonstrate the superiority of BNMTF over existing NMF-based methods.

Notations. Throughout the paper, we use lowercase letters for scalars, bold lowercase letters for vectors, and bold uppercase letters for matrices. We use $\mathrm{tr}(\mathbf{M})$ to denote the trace of a square matrix $\mathbf{M}$. For vector and matrix norms, we use $\|\mathbf{M}\|_1$ to denote the l_1 norm of a matrix $\mathbf{M}$, which is equal to the sum of the absolute values of all the elements of $\mathbf{M}$, $\|\mathbf{M}\|_F$ to denote the Frobenius norm of $\mathbf{M}$, and $\|\mathbf{a}\|_1$ and $\|\mathbf{a}\|_2$ to denote the l_1 and l_2 norms, respectively, of a vector $\mathbf{a}$. To characterize a matrix, we write $\mathbf{M} \geq 0$ to mean that all elements in $\mathbf{M}$ are nonnegative, and $\mathbf{M}_1 \geq (\leq) \mathbf{M}_2$ to denote elementwise inequality between $\mathbf{M}_1$ and $\mathbf{M}_2$. We use $\mathbf{0}$ and $\mathbf{1}$ to denote a vector or matrix consisting of all zeros and ones, respectively, of the appropriate size. The identity matrix of an appropriate size is denoted by $\mathbf{I}$ and the Hadamard or elementwise product is denoted by $\odot$.

2. COMMUNITY DETECTION VIA BNMTF

In this section, we present the BNMTF model and put it in the context of related methods for overlapping community detection. Details on how to solve the optimization problem for parameter learning will be presented in the next section.

2.1 Model Formulation

Let us denote a network by $\mathcal{N} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of n vertices and $\mathcal{E}$ is a set of m edges with each of them connecting a pair of vertices in $\mathcal{V}$. The adjacency matrix is a nonnegative matrix $\mathbf{G} \in \mathbb{R}_+^{n\times n}$ whose (i, j)th entry g_{ij} represents the edge weight between the ith and jth vertices.

Depending on whether the network is directed or undirected, $\mathbf{G}$ is asymmetric or symmetric accordingly. Moreover, the network is usually far from being a fully connected network, implying that $\mathbf{G}$ is sparse and hence $m \ll n(n-1)/2$.

We assume that the maximum number of possible communities, denoted by k, is given. In our opinion, a maximum number is much easier to set than the exact number. We use a matrix $\mathbf{U} \in \mathbb{R}_+^{n\times k}$ to denote the community membership of the n vertices in $\mathcal{V}$ and $\mathbf{B} \in \mathbb{R}_+^{k\times k}$ to denote the community interaction matrix. For each entry u_{ij} in $\mathbf{U}$, we interpret it as the probability that the ith vertex belongs to the jth community. The higher the value of u_{ij}, the more active is the ith entity in the jth community. We thus have the following bounded constraint on each element of $\mathbf{U}$:

$$0 \leq u_{ij} \leq 1 \text{ or equivalently } \mathbf{0} \leq \mathbf{U} \leq \mathbf{1}. \tag{1}$$

The matrix $\mathbf{B}$ represents the relations between communities. For example, the communities (or interest groups) 'Economics' and 'Politics' are strongly related and hence we expect a large value for the corresponding entry in $\mathbf{B}$; on the contrary, the communities 'Movies' and 'Politics' are only weakly related and hence the corresponding entry in $\mathbf{B}$ is likely to be small. The product form $\mathbf{UBU}^T$ represents the relation between any two vertices in terms of the community structure. We want to use it to approximate the adjacency matrix $\mathbf{G}$. In other words, we approximate $\mathbf{G}$ by a nonnegative matrix tri-factorization $\mathbf{UBU}^T$ with bounded $\mathbf{U}$:

$$\mathbf{G} \approx \hat{\mathbf{G}} \equiv \mathbf{UBU}^T.$$

Two loss functions, namely, squared loss and generalized KL-divergence, are used to measure the approximation error. Specifically, they are defined as

$$L_{sq}(\mathbf{G}, \mathbf{U}, \mathbf{B}) = \|\mathbf{G} - \hat{\mathbf{G}}\|_F^2 \tag{2}$$

$$L_{kl}(\mathbf{G}, \mathbf{U}, \mathbf{B}) = \sum_{i,j} \left(g_{ij} \ln \frac{g_{ij}}{\hat{g}_{ij}} - g_{ij} + \hat{g}_{ij} \right), \tag{3}$$

where $\hat{g}_{ij}$ is the (i, j)th entry of $\hat{\mathbf{G}}$.

As discussed above, $\mathbf{G}$ is usually sparse. In previous community detection methods, the zero entries in $\mathbf{G}$ are often interpreted as having no edges between the corresponding vertices. However, it is also possible that some otherwise nonzero entries have not been observed during the data collection process. If a loss function is defined on the whole matrix $\mathbf{G}$, then even the second case will be treated as the first case, bringing additional noise to the learning process. Here we propose another loss function which is defined only on the edges with $g_{ij} > 0$:

$$l_{sq}(\mathbf{G}, \mathbf{U}, \mathbf{B}) = \sum_{g_{ij}>0} (g_{ij} - \hat{g}_{ij})^2 \tag{4}$$

$$l_{kl}(\mathbf{G}, \mathbf{U}, \mathbf{B}) = \sum_{g_{ij}>0} \left(g_{ij} \ln \frac{g_{ij}}{\hat{g}_{ij}} - g_{ij} + \hat{g}_{ij} \right). \tag{5}$$

In real networks, usually each vertex does not participate in too many communities and so $\mathbf{U}$ is sparse. To enhance the sparsity of $\mathbf{U}$, we use the l_1 norm to regularize it, i.e., $\|\mathbf{U}\|_1 = \mathbf{1}^T\mathbf{U}\mathbf{1}$ due to the nonnegativity of $\mathbf{U}$. Moreover, by utilizing the l_1 norm of $\mathbf{U}$, we can reduce the effective number of free parameters in $\mathbf{U}$ and hence control the complexity of the model.

Combining the several considerations above, the optimization problem underlying BNMTF can be formulated as

$$\min_{\mathbf{U},\mathbf{B}} \; L(\mathbf{G}, \mathbf{U}, \mathbf{B}) + \lambda \mathbf{1}^T\mathbf{U}\mathbf{1} \;\text{ s.t. } \mathbf{0} \leq \mathbf{U} \leq \mathbf{1}, \mathbf{B} \geq \mathbf{0}. \tag{6}$$

In the objective function, the loss function $L(\mathbf{G}, \mathbf{U}, \mathbf{B})$ can be any of the loss functions defined in Eqs. (2) to (5) and the regularization parameter $\lambda > 0$ balances the tradeoff between the approximation error and the complexity of $\mathbf{U}$.

2.2 Related Methods

As discussed above, two NMF-based community detection methods have been developed. In this subsection, we give a more in-depth review of these two methods to put the BNMTF model in perspective.

Psorakis et al. [18] used the conventional NMF model which has two factors. Specifically, using the generalized KL-divergence as in Eq. (3), the model approximates $\mathbf{G}$ using $\mathbf{X}\mathbf{Y}^T$ for directed networks and $\mathbf{X}\mathbf{X}^T$ for undirected networks. One disadvantage of this formulation is that the physical meaning of $\mathbf{X}$ and $\mathbf{Y}$ is not very clear, because $\mathbf{X}$ and $\mathbf{Y}$ cannot be interpreted directly as representing community membership since different entries are of different scales. From the perspective of BNMTF, $\mathbf{X}$ (or $\mathbf{Y}$) may take the form $\mathbf{U}\mathbf{V}$ where $\mathbf{V}$ is related to $\mathbf{B}$. Thus, $\mathbf{X}$ (or $\mathbf{Y}$) captures both $\mathbf{U}$ and $\mathbf{B}$, making its physical meaning unclear. For undirected networks, using a positive semidefinite (PSD) matrix $\mathbf{X}\mathbf{X}^T$ to approximate the symmetric matrix $\mathbf{G}$ is not very suitable in some sense. For example, the diagonal elements of $\mathbf{X}\mathbf{X}^T$ are usually positive but those of $\mathbf{G}$ are almost zero. Moreover, the eigenvalues of $\mathbf{X}\mathbf{X}^T$ are all nonnegative but $\mathbf{G}$ can have negative eigenvalues. For directed networks, the number of parameters ($2nk$) is much larger than that of the undirected case (nk) and also that of BNMTF ($nk+k^2$). As a result, its model complexity is also higher accordingly. Moreover, since $\mathbf{X}\mathbf{Y}^T = (\mathbf{X}\mathbf{D})(\mathbf{Y}\mathbf{D}^{-1})^T$ where $\mathbf{D}$ can be any diagonal matrix with positive diagonal elements, this factorization also faces the nonidentifiability problem and hence slows down the convergence of the algorithm.

In [19], the authors treated directed and undirected networks differently. For undirected networks, like in [18], the factorization form $\mathbf{X}\mathbf{X}^T$ is used but with the squared loss as in Eq. (2). For directed networks, however, a different form $\mathbf{X}\mathbf{A}\mathbf{X}^T$ is used. On the contrary, BNMTF has a unified formulation for both directed and undirected networks, with the only difference being that $\mathbf{B}$ is symmetric or asymmetric according to whether the network is undirected or directed. With a unified formulation, we can gain a better understanding of our model under different settings. Moreover, from the implementation point of view, having a unified formulation is also favorable due to the high degree of code reusability. For undirected networks, because the model in [19] is similar to that in [18], it has the same drawbacks as discussed above. For directed networks, even though their formulation appears to be similar to ours, there exist some crucial differences. For example, their method implicitly assumes that each row of $\mathbf{X}$ represents the probability distribution that the corresponding vertex belongs to each of the communities because a postprocessing step makes the probabilities sum to 1. They do not impose constraints on $\mathbf{X}$ directly because doing so would make the optimization problem even more difficult due to the nonexistence of an analytical solution. However, in order to model overlapping communities, we believe the probabilities of each vertex belonging to different communities should not be constrained as in [19]. For instance, a vertex may belong to the community 'Politics' with probability 0.9 and the community 'Economics' with probability 0.8, due to the strong relation between these two communities. An entity can be very active in multiple communities, but this scenario cannot be modeled well if we impose the constraint that the row sum be equal to 1.

Besides, the network sparsity problem has not been addressed by both methods in [18, 19].

In summary, compared with [18, 19], BNMTF has some appealing advantages:

1) The use of a tri-factorization form in BNMTF gives clear physical meaning to each factor.
2) A unified formulation is used for both directed and undirected networks.
3) BNMTF addresses the network sparsity problem explicitly by using a loss function, as in Eq. (4) or (5), based only on the observed data.

We note that nonnegative matrix tri-factorization has been investigated by some other researchers. For example, Ding et al. [5] proposed an orthogonal nonnegative matrix tri-factorization method for clustering problems by placing an orthogonal constraint on $\mathbf{U}$. However, their method is not suitable for overlapping community detection because it assumes that each vertex can only belong to one community. To the best of our knowledge, there does not exist any nonnegative matrix tri-factorization method with bounded constraints imposed.

3. PARAMETER LEARNING

In this section, we discuss how to solve the optimization problem (6) efficiently.

In general, three types of optimization methods have been used for solving NMF-based methods, namely, auxiliary function methods [11], which are similar to the majorization-minimization approach [9] in statistics, projected gradient methods [12], and the newly emerging coordinate descent methods [3, 8]. Among these three approaches, auxiliary function and coordinate descent methods are very popular due to their efficiency and the existence of analytical solutions. In what follows, we will develop coordinate descent and auxiliary function methods for the BNMTF model.

3.1 Squared Loss

We first discuss how to solve problem (6) when the loss function is the squared loss as in Eq. (2) or (4). For convenience, let us unify Eqs. (2) and (4) to the following form:

$$L(\mathbf{G}, \mathbf{U}, \mathbf{B}) = \sum_{(i,j)\in\mathcal{I}} (g_{ij} - \hat{g}_{ij})^2, \tag{7}$$

where $\mathcal{I}$ is an index set of the available entries in $\mathbf{G}$. More specifically, for the squared loss defined in Eq. (2), $\mathcal{I}$ denotes the set of all indices in $\mathbf{G}$ for directed networks and the set of all indices in the upper-triangular portion of $\mathbf{G}$ for undirected networks. For the squared loss defined in Eq. (4), $\mathcal{I}$ denotes the set of indices of all the nonzero entries of $\mathbf{G}$ for directed networks and the set of indices of all nonzero entries in the upper-triangular portion of $\mathbf{G}$ for undirected networks.

Here we use the coordinate descent method to solve problem (6) with the unified squared loss function as in Eq. (7).[1]

[1]We have also tried to devise an auxiliary function method. However, due to the existence of l_1 regularization and the bounded constraint on $\mathbf{U}$, we have not been able to find a good auxiliary function, as an upper bound of the objective function of problem (6), with an analytical solution.

For the matrix $\mathbf{U}$, the coordinate descent method considers a one-variable update for problem (6) as

$$\begin{aligned} \min_{t} \quad & f_{pq}^{U}(t) = L(\mathbf{G}, \mathbf{U} + t\mathbf{O}_{nk}^{pq}, \mathbf{B}) + \lambda \mathbf{1}^T(\mathbf{U} + t\mathbf{O}_{nk}^{pq})\mathbf{1} \\ \text{s.t.} \quad & 0 \le u_{pq} + t \le 1, \end{aligned} \tag{8}$$

where $\mathbf{O}_{nk}^{pq}$ denotes an $n \times k$ matrix with all entries equal to 0 except the (p,q)th entry which is equal to 1, and u_{pg} denotes the (p,q)th entry of $\mathbf{U}$. By introducing a weight matrix $\mathbf{W}$ whose (i,j)th entry equals 1 when $(i,j) \in \mathcal{I}$ and 0 otherwise, we rewrite the unified squared loss as

$$L(\mathbf{G}, \mathbf{U}, \mathbf{B}) = \|\mathbf{W} \odot (\mathbf{G} - \mathbf{U}\mathbf{B}\mathbf{U}^T)\|_F^2$$

and then simplify $f_{pg}^{U}(t)$ to

$$\begin{aligned} & f_{pq}^{U}(t) \\ =& \left\| \mathbf{W} \odot \left(\mathbf{G} - (\mathbf{U} + t\mathbf{O}_{nk}^{pq})\mathbf{B}(\mathbf{U} + t\mathbf{O}_{nk}^{pq})^T \right) \right\|_F^2 + \lambda t + \lambda \mathbf{1}^T\mathbf{U}\mathbf{1} \\ =& \|\mathbf{W} \odot (\mathbf{Z}_0 + t\mathbf{Z}_1 + t^2\mathbf{Z}_2)\|_F^2 + \lambda t + \lambda \mathbf{1}^T\mathbf{U}\mathbf{1} \\ =& \mathrm{tr}\left(\left(\mathbf{W} \odot (\mathbf{Z}_0 + t\mathbf{Z}_1 + t^2\mathbf{Z}_2)\right) \left(\mathbf{W} \odot (\mathbf{Z}_0 + t\mathbf{Z}_1 + t^2\mathbf{Z}_2)\right)^T \right) \\ & + \lambda t + \lambda \mathbf{1}^T\mathbf{U}\mathbf{1} \\ =& at^4 + bt^3 + ct^2 + dt + e, \end{aligned} \tag{9}$$

where $\mathbf{Z}_0 = \mathbf{U}\mathbf{B}\mathbf{U}^T - \mathbf{G}$, $\mathbf{Z}_1 = \mathbf{O}_{nk}^{pq}\mathbf{B}\mathbf{U}^T + \mathbf{U}\mathbf{B}(\mathbf{O}_{nk}^{pq})^T$, $\mathbf{Z}_2 = \mathbf{O}_{nk}^{pq}\mathbf{B}(\mathbf{O}_{nk}^{pq})^T$, $a = \mathrm{tr}((\mathbf{W} \odot \mathbf{Z}_2)(\mathbf{W} \odot \mathbf{Z}_2)^T)$, $b = 2\mathrm{tr}((\mathbf{W} \odot \mathbf{Z}_1)(\mathbf{W} \odot \mathbf{Z}_2)^T)$, $c = 2\mathrm{tr}((\mathbf{W} \odot \mathbf{Z}_0)(\mathbf{W} \odot \mathbf{Z}_2)^T) + \mathrm{tr}((\mathbf{W} \odot \mathbf{Z}_1)(\mathbf{W} \odot \mathbf{Z}_1)^T)$, $d = 2\mathrm{tr}((\mathbf{W} \odot \mathbf{Z}_0)(\mathbf{W} \odot \mathbf{Z}_1)^T) + \lambda$, and $e = \mathrm{tr}((\mathbf{W} \odot \mathbf{Z}_0)(\mathbf{W} \odot \mathbf{Z}_0)^T) + \lambda \mathbf{1}^T\mathbf{U}\mathbf{1}$. We can further simplify a, b, c and d by noting the following:

$$\begin{aligned} \mathrm{tr}\left((\mathbf{W} \odot \mathbf{Z}_i)^T(\mathbf{W} \odot \mathbf{Z}_j) \right) &= \mathrm{tr}\left((\mathbf{W} \odot \mathbf{Z}_i \odot \mathbf{W})^T\mathbf{Z}_j \right) \\ &= \mathrm{tr}\left((\mathbf{W} \odot \mathbf{Z}_i)^T\mathbf{Z}_j \right), \end{aligned}$$

where the first equation follows from a property of the Hadamard product that $\mathrm{tr}(\mathbf{A}^T(\mathbf{C} \odot \mathbf{D})) = \mathrm{tr}((\mathbf{A} \odot \mathbf{C})^T\mathbf{D})$, and the second equation follows from another property of the Hadamard product that $\mathbf{A} \odot \mathbf{C} \odot \mathbf{D} = \mathbf{A} \odot \mathbf{D} \odot \mathbf{C}$ and a property of $\mathbf{W}$ that $\mathbf{W} \odot \mathbf{W} = \mathbf{W}$ because $\mathbf{W} \in \{0,1\}^{n \times n}$ is a binary matrix. If there is no constraint in problem (8), we can set the derivative of $f_{pg}^{U}(t)$ with respect to t to 0 and get

$$h(t) = 4at^3 + 3bt^2 + 2ct + d = 0, \tag{10}$$

which amounts to solving a root finding problem. It is easy to show that $a, b \ge 0$. Since different values of a, b, c and d affect the result of the root finding problem, we discuss it separately for the following cases:

1) $a > 0$
2) $a = 0, b > 0$
3) $a = 0, b = 0, c \neq 0$
4) $a = 0, b = 0, c = 0, d \neq 0$.

The solutions for these four cases are depicted in Tables 1 to 4 separately.

For the matrix $\mathbf{B}$, we consider two situations depending on whether the network is directed. When it is directed, $\mathbf{B}$ is a general nonnegative matrix and the objective function for each step of the coordinate descent method is formulated as

$$\begin{aligned} \min_{t} \quad & f_{pq}^{B}(t) = L(\mathbf{G}, \mathbf{U}, \mathbf{B} + t\mathbf{O}_{kk}^{pq}) + \lambda \mathbf{1}^T\mathbf{U}\mathbf{1} \\ \text{s.t.} \quad & b_{pq} + t \ge 0, \end{aligned} \tag{11}$$

Table 1: Algorithm for solving problem (8) when $a > 0$.

calculate $r = \frac{c}{2a} - \frac{3b^2}{16a^2}$;
calculate $s = \frac{b^3}{32a^3} - \frac{bc}{8a^2} + \frac{d}{4a}$;
calculate $\Delta = \frac{r^3}{27} + \frac{s^2}{4}$;
if $\Delta > 0$
  calculate $t_1 = \sqrt[3]{\sqrt{\Delta} - \frac{s}{2}} - \sqrt[3]{\sqrt{\Delta} + \frac{s}{2}} - \frac{b}{4a}$;
  $t = \max(-u_{pq}, \min(1 - u_{pq}, t_1))$;
elseif $\Delta == 0$
  $t_1 = -2\sqrt[3]{\frac{s}{2}} - \frac{b}{4a}$;
  $t_2 = \sqrt[3]{\frac{s}{2}} - \frac{b}{4a}$;
  if $s == 0$
    $t = \max(-u_{pq}, \min(1 - u_{pq}, t_1))$;
  else
    if $t_1 \in [-u_{pq}, 1 - u_{pq}]$ and $t_2 \in [-u_{pq}, 1 - u_{pq}]$
      $t = \begin{cases} t_1 & \text{if } f_{pq}^{U}(t_1) <= f_{pq}^{U}(t_2) \\ t_2 & \text{otherwise} \end{cases}$;
    elseif $t_1 \in [-u_{pq}, 1 - u_{pq}]$
      $t = t_1$;
    elseif $t_2 \in [-u_{pq}, 1 - u_{pq}]$
      $t = t_2$;
    elseif $t_1 < -u_{pq}$
      $t = -u_{pq}$;
    else
      $t = 1 - u_{pq}$;
    end
  end
else
  calculate $\rho = \sqrt{\frac{s^2}{4} - \Delta} = \sqrt{-\frac{r^3}{27}}$;
  calculate ϕ according to $cos(\phi) = -\frac{s}{2\rho}$;
  calculate $t_i = 2\rho^{\frac{1}{3}} \cos\left(\frac{\phi + 2(i-1)\pi}{3}\right) - \frac{b}{4a}$, for $i = 1, 2, 3$,
  assumed $t_1 < t_2 < t_3$;
  if $t_1 \in [-u_{pq}, 1 - u_{pq}]$ and $t_3 \in [-u_{pq}, 1 - u_{pq}]$
    $t = \begin{cases} t_1 & \text{if } f_{pq}^{U}(t_1) <= f_{pq}^{U}(t_3) \\ t_3 & \text{otherwise} \end{cases}$;
  elseif $t_1 \in [-u_{pq}, 1 - u_{pq}]$
    $t = t_1$;
  elseif $t_3 \in [-u_{pq}, 1 - u_{pq}]$
    $t = t_3$;
  elseif $t_2 \in [-u_{pq}, 1 - u_{pq}]$
    $t = \begin{cases} -u_{pq} & \text{if } f_{pq}^{U}(-u_{pq}) <= f_{pq}^{U}(1 - u_{pq}) \\ 1 - u_{pq} & \text{otherwise} \end{cases}$;
  elseif $t_1 > 1 - u_{pq}$
    $t = 1 - u_{pq}$;
  elseif $t_3 < -u_{pq}$
    $t = -u_{pq}$;
  elseif $t_2 > 1 - u_{pq}$
    $t = -u_{pq}$;
  else
    $t = 1 - u_{pq}$;
  end
end

where b_{pq} is the (p,q)th entry of $\mathbf{B}$. We simplify $f_{pq}^{B}(t)$ as

$$\begin{aligned} & f_{pq}^{B}(t) \\ =& \left\| \mathbf{W} \odot \left(\mathbf{G} - \mathbf{U}(\mathbf{B} + t\mathbf{O}_{kk}^{pq})\mathbf{U}^T \right) \right\|_F^2 + \lambda \mathbf{1}^T\mathbf{U}\mathbf{1} \\ =& \|\mathbf{W} \odot (t\mathbf{P}_0 + \mathbf{P}_1)\|_F^2 + \lambda \mathbf{1}^T\mathbf{U}\mathbf{1} \\ =& t^2\mathrm{tr}((\mathbf{W} \odot \mathbf{P}_0)^T\mathbf{P}_0) + 2t\mathrm{tr}((\mathbf{W} \odot \mathbf{P}_1)^T\mathbf{P}_0) + \mathrm{tr}((\mathbf{W} \odot \mathbf{P}_1)^T\mathbf{P}_1) \\ & + \lambda \mathbf{1}^T\mathbf{U}\mathbf{1}, \end{aligned}$$

where $\mathbf{P}_0 = \mathbf{U}\mathbf{O}_{kk}^{pq}\mathbf{U}^T$ and $\mathbf{P}_1 = \mathbf{U}\mathbf{B}\mathbf{U}^T - \mathbf{G}$. We note that $f_{pq}^{B}(t)$ is a quadratic function of t. It is easy to show that the solution to problem (11) can be calculated as

$$\begin{aligned} t &= \max\left(-b_{pq}, -\frac{\mathrm{tr}((\mathbf{W} \odot \mathbf{P}_1)^T\mathbf{P}_0)}{\mathrm{tr}((\mathbf{W} \odot \mathbf{P}_0)^T\mathbf{P}_0)} \right) \\ &= -\min\left(b_{pq}, \frac{\mathrm{tr}((\mathbf{W} \odot \mathbf{P}_1)^T\mathbf{P}_0)}{\mathrm{tr}((\mathbf{W} \odot \mathbf{P}_0)^T\mathbf{P}_0)} \right). \end{aligned}$$

Table 2: Algorithm for solving problem (8) when $a = 0, b > 0$.

```
if c^2 < 3bd
    t = -u_pq;
elseif c^2 > 3bd
    calculate t_1 = (-c+sqrt(c^2-3bd))/3b;
    calculate t_2 = (-c-sqrt(c^2-3bd))/3b;
    if t_1 ∈ [-u_pq, 1-u_pq]
        t = t_1;
    elseif t_2 ∈ [-u_pq, 1-u_pq]
        t = { -u_pq      if f^U_pq(-u_pq) <= f^U_pq(1-u_pq)
            { 1-u_pq     otherwise                          ;
    elseif t_1 < -u_pq or t_2 > 1-u_pq
        t = -u_pq;
    else
        t = 1-u_pq;
    end
else
    calculate t_1 = -c/3b;
    if t_1 ∈ [-u_pq, 1-u_pq]
        t = t_1;
    else
        t = -u_pq;
    end
end
```

Table 3: Algorithm for solving problem (8) when $a = 0, b = 0, c \neq 0$.

```
calculate t_1 = -d/2c;
if c > 0
    t = max(-u_pq, min(1-u_pq, t_1));
else
    t = { -u_pq      if |-u_pq-t_1| >= |1-u_pq-t_1|
        { 1-u_pq     otherwise                      ;
end
```

Table 4: Algorithm for solving problem (8) when $a = 0, b = 0, c = 0, d \neq 0$.

$$t = \begin{cases} -u_{pq} & \text{if } d > 0 \\ 1 - u_{pq} & \text{otherwise} \end{cases};$$

When the network is undirected, $\mathbf{B}$ is symmetric and the objective function for a non-diagonal entry $b_{pq}\,(p \neq q)$ is given by

$$\begin{aligned} \min_{t} \quad & f^B_{pq}(t) = L(\mathbf{G}, \mathbf{U}, \mathbf{B} + t\mathbf{O}^{pq}_{kk} + t\mathbf{O}^{qp}_{kk}) + \lambda \mathbf{1}^T\mathbf{U}\mathbf{1} \\ \text{s.t.} \quad & b_{pq} + t \geq 0, \end{aligned} \tag{12}$$

where the objective function can be simplified to

$$\begin{aligned} & f^B_{pq}(t) \\ =& \left\| \mathbf{W} \odot \left(\mathbf{G} - \mathbf{U}(\mathbf{B} + t\mathbf{O}^{pq}_{kk} + t\mathbf{O}^{qp}_{kk})\mathbf{U}^T \right) \right\|^2_F + \lambda \mathbf{1}^T\mathbf{U}\mathbf{1} \\ =& \|\mathbf{W} \odot (t\mathbf{P}_2 + \mathbf{P}_1)\|^2_F + \lambda \mathbf{1}^T\mathbf{U}\mathbf{1} \\ =& t^2 \mathrm{tr}((\mathbf{W} \odot \mathbf{P}_2)^T\mathbf{P}_2) + 2t\,\mathrm{tr}((\mathbf{W} \odot \mathbf{P}_1)^T\mathbf{P}_2) + \mathrm{tr}((\mathbf{W} \odot \mathbf{P}_1)^T\mathbf{P}_1) \\ & + \lambda \mathbf{1}^T\mathbf{U}\mathbf{1}, \end{aligned}$$

with $\mathbf{P}_2$ defined as $\mathbf{P}_2 = \mathbf{U}(\mathbf{O}^{pq}_{kk} + \mathbf{O}^{qp}_{kk})\mathbf{U}^T$. Then the optimal t can be calculated as

$$t = -\min\left(b_{pq}, \frac{\mathrm{tr}((\mathbf{W} \odot \mathbf{P}_1)^T\mathbf{P}_2)}{\mathrm{tr}((\mathbf{W} \odot \mathbf{P}_2)^T\mathbf{P}_2)} \right).$$

For each diagonal entry b_{pp}, the update rule is identical to that for directed networks and we omit the derivation here.

3.1.1 Some Implementation Issues

For the update of $\mathbf{U}$, we need to efficiently compute the coefficients in Eq. (9) or, equivalently, the matrices $\mathbf{Z}_0$, $\mathbf{Z}_1$ and $\mathbf{Z}_2$. According to the definition of $\mathbf{Z}_2$, it is a zero matrix with only one nonzero element indexed by (p, p) as b_{qq}. $\mathbf{Z}_0$ measures the approximation residual and we can keep track of $\mathbf{Z}_0$ and update it whenever each entry in $\mathbf{U}$ and $\mathbf{B}$ changes. Since $\mathbf{Z}_1$ is related to $\mathbf{UB}$ and $\mathbf{BU}^T$, similar to $\mathbf{Z}_0$, we also keep track of $\mathbf{UB}$ and $\mathbf{BU}^T$. By taking advantage of the extreme sparsity of the matrices involved, we can compute the coefficients in Eq. (9) efficiently.

For the update of $\mathbf{B}$, we need to compute $\mathbf{P}_0$, $\mathbf{P}_1$ and $\mathbf{P}_2$. Note that $\mathbf{P}_1$ is equal to $\mathbf{Z}_0$ which is kept as a global variable and needs no additional computation. For $\mathbf{P}_0$, it is easy to show that $\mathbf{P}_0 = \mathbf{u}_p\mathbf{u}_q^T$ according to its definition, where $\mathbf{u}_p$ is the pth column of $\mathbf{U}$, and hence we can compute $\mathbf{P}_0$ more efficiently. Similar to $\mathbf{P}_0$, $\mathbf{P}_2$ can be computed as $\mathbf{P}_2 = \mathbf{u}_p\mathbf{u}_q^T + \mathbf{u}_q\mathbf{u}_p^T$ in an efficient way.

3.2 Generalized KL-Divergence

Similar to the squared loss, we first give a unified form of the generalized KL-divergence as follows

$$L(\mathbf{G}, \mathbf{U}, \mathbf{B}) = \sum_{(i,j)\in\mathcal{I}} \left(g_{ij} \ln \frac{g_{ij}}{\hat{g}_{ij}} - g_{ij} + \hat{g}_{ij} \right). \tag{13}$$

According to [3, 8], the coordinate descent method has no closed-form solution and so we resort to an auxiliary function method. We first give the definition of an auxiliary function.

Definition 1. *$G(\mathbf{h}, \mathbf{h}_0)$ is an auxiliary function for a function $F(\mathbf{h})$ if the conditions*

$$G(\mathbf{h}, \mathbf{h}_0) \geq F(\mathbf{h}); \quad G(\mathbf{h}, \mathbf{h}) = F(\mathbf{h})$$

are satisfied.

For the minimization of $F(\mathbf{h})$, we can minimize $G(\mathbf{h}, \mathbf{h}_0)$ instead based on the following lemma from [9, 11].

Lemma 1. *For an estimate $\mathbf{h}_0$ of $F(\mathbf{h})$, we can update it to $\mathbf{h}_1$ with $F(\mathbf{h}_1) \leq F(\mathbf{h}_0)$ where $\mathbf{h}_1$ satisfies*

$$\mathbf{h}_1 = \arg\min_{\mathbf{h}} G(\mathbf{h}, \mathbf{h}_0).$$

With $\mathbf{B}$ fixed, the objective function for $\mathbf{U}$ is formulated as

$$F(\mathbf{U}) = \sum_{(i,j)\in\mathcal{I}} ([\mathbf{UBU}^T]_{ij} - g_{ij} \ln[\mathbf{UBU}^T]_{ij}) + \lambda \mathbf{1}^T\mathbf{U}\mathbf{1},$$

where $[\mathbf{A}]_{ij}$ denotes the (i, j)th element of $\mathbf{A}$. Then we define the auxiliary function for $F(\mathbf{U})$ in the following theorem.

Theorem 1.

$$\begin{aligned} & G(\mathbf{U}, \hat{\mathbf{U}}) \\ =& \sum_{(i,j)\in\mathcal{I}} \left(\sum_{r=1}^{k}\sum_{s=1}^{k} b_{rs} \left(\frac{(\hat{u}_{js} + \epsilon)(u_{ir} + \epsilon)^2}{2(\hat{u}_{ir} + \epsilon)} + \frac{(\hat{u}_{ir} + \epsilon)(u_{js} + \epsilon)^2}{2(\hat{u}_{js} + \epsilon)} \right) \right. \\ & \left. - \sum_{r=1}^{k}\sum_{s=1}^{k} \left(\epsilon b_{rs} u_{js} + \epsilon b_{rs} u_{ir} + \epsilon^2 b_{rs} \right) - g_{ij} \sum_{r,s} \alpha^{ij}_{rs} \ln \frac{u_{ir} b_{rs} u_{js}}{\alpha^{ij}_{rs}} \right) \\ & + \lambda \mathbf{1}^T\mathbf{U}\mathbf{1}, \end{aligned}$$

where ϵ is a positive constant, $\hat{u}_{ij}$ is the (i,j)th element of $\hat{\mathbf{U}}$, and $\alpha_{rs}^{ij} = \frac{\hat{u}_{ir} b_{rs} \hat{u}_{js}}{\sum_{r,s} \hat{u}_{ir} b_{rs} \hat{u}_{js}}$ is the auxiliary function of $F(\mathbf{U})$.

In Theorem 1, we add ϵ to each $\hat{u}_{rs}$ to increase numerical stability because $\hat{u}_{rs}$ may be equal or very close to 0. Then we need to minimize $G(\mathbf{U}, \hat{\mathbf{U}})$ with respect to $\mathbf{U}$ where $\hat{\mathbf{U}}$ is the current estimate of $\mathbf{U}$. For each entry of $\mathbf{U}$, the optimization problem is formulated by rewriting $G(\mathbf{U}, \hat{\mathbf{U}})$ as

$$\begin{aligned} \min_{u_{pq}} \quad & f(u_{pq}) = \frac{a}{2} u_{pq}^2 + b u_{pq} - c \ln u_{pq} \\ \text{s.t.} \quad & 0 \le u_{pq} \le 1, \end{aligned} \tag{14}$$

where

$$\begin{aligned} \mathcal{I}_p^+ &= \{j \,|\, (p,j) \in \mathcal{I}\}, \quad \mathcal{I}_p^- = \{i \,|\, (i,p) \in \mathcal{I}\} \\ a &= \sum_{j \in \mathcal{I}_p^+} \sum_{s=1}^{k} \frac{(\hat{u}_{js} + \epsilon) b_{qs}}{\hat{u}_{pq} + \epsilon} + \sum_{i \in \mathcal{I}_p^-} \sum_{r=1}^{k} \frac{b_{rq} (\hat{u}_{ir} + \epsilon)}{\hat{u}_{pq} + \epsilon} \\ b &= \epsilon (a - \sum_{i \in \mathcal{I}_p^-} \sum_{r=1}^{k} b_{rq} - \sum_{j \in \mathcal{I}_p^+} \sum_{s=1}^{k} b_{qs}) + \lambda \\ c &= \sum_{j \in \mathcal{I}_p^+} \sum_{s=1}^{k} g_{pj} \alpha_{qs}^{pj} + \sum_{i \in \mathcal{I}_p^-} \sum_{r=1}^{k} g_{ip} \alpha_{rq}^{ip}. \end{aligned}$$

If $a = 0$ and $b = 0$, $f(u_{pq})$ degenerates to $f(u_{pq}) = -c \ln u_{pq}$. It is easy to show that the optimal solution is $u_{pq} = 1$ because $c \ge 0$.

If $a = 0$ and $b > 0$, we set the derivative of $f(u_{pq})$ with respect to u_{pq} to 0 and get the solution as $u_{pq,1} = \frac{c}{b}$. If $c \le b$, we can get $u_{pq,1} \in [0,1]$ and since the second-order derivative $f''(u_{pq,1}) = \frac{c}{u_{pq,1}^2} \ge 0$, $u_{pq,1}$ is the optimal solution. If $c > b$, we can get $f'(u_{pq}) = b - \frac{c}{u_{pq}} < 0$ for $u_{pq} \in [0,1]$, implying that $f(u_{pq})$ is nonincreasing and hence 1 is the optimal solution.

If $a = 0$ and $b < 0$, $f'(u_{pq}) = b - \frac{c}{u_{pq}} < 0$ over $[0,1]$, which implies that $f(u_{pq})$ is nonincreasing and hence $u_{pq} = 1$ is the optimal solution.

When $a > 0$,[2] the situation is more complicated. If there is no constraint in problem (14), we can set the derivative of $f(u_{pq})$ with respect to u_{pq} to 0 and obtain the solutions for u_{pq} as

$$u_{pq,1} = \frac{-b + \sqrt{b^2 + 4ac}}{2a}, \; u_{pq,2} = \frac{-b - \sqrt{b^2 + 4ac}}{2a}.$$

It is easy to show that $u_{pq,1} \ge 0$ and $u_{pq,2} \le 0$ since $a > 0$ and $c \ge 0$. When $u_{pq,1} \in [0,1]$, we have $f''(u_{pq}) = a + \frac{c}{u_{pq}} > 0$, implying that $u_{pq,1}$ is the optimal solution of problem (14). When $u_{pq,1} \notin [0,1]$ or equivalently $u_{pq,1} > 1$ since $u_{pq,1} \ge 0$, the derivative of $f(\cdot)$ denoted by $f'(u_{pq})$ is negative over $[0,1]$ since $f'(u_{pq}) = \frac{a}{u_{pq}}(u_{pq} - u_{pq,1})(u_{pq} - u_{pq,2})$ due to $u_{pq,1} > 1$ and $u_{pq,2} \le 0$. This implies that $f(u_{pq})$ is nonincreasing over $[0,1]$ and hence $u_{pq} = 1$ is the optimal solution to problem (14).

When $\mathbf{U}$ is fixed, the objective function for $\mathbf{B}$ is formulated as

$$F(\mathbf{B}) = \sum_{(i,j) \in \mathcal{I}} ([\mathbf{U}\mathbf{B}\mathbf{U}^T]_{ij} - g_{ij} \ln [\mathbf{U}\mathbf{B}\mathbf{U}^T]_{ij}).$$

For the auxiliary function of $F(\mathbf{B})$, we have the following result.

[2]By definition a cannot be negative.

THEOREM 2.

$$G(\mathbf{B}, \hat{\mathbf{B}}) = \sum_{(i,j) \in \mathcal{I}} \left(\sum_{r=1}^{k} \sum_{s=1}^{k} b_{rs} u_{ir} u_{js} - g_{ij} \sum_{r=1}^{k} \sum_{s=1}^{k} \beta_{rs}^{ij} \ln \frac{b_{rs} u_{ir} u_{js}}{\beta_{rs}^{ij}} \right),$$

where $\hat{b}_{rs}$ is the (r,s)th element of $\hat{\mathbf{B}}$ and $\beta_{rs}^{ij} = \frac{\hat{b}_{rs} u_{ir} u_{js}}{\sum_{r,s} \hat{b}_{rs} u_{ir} u_{js}}$ is the auxiliary function of $F(\mathbf{B})$.

Then we need to minimize $G(\mathbf{B}, \hat{\mathbf{B}})$, where $\hat{\mathbf{B}}$ is the current estimate of $\mathbf{B}$, with the objective function for each entry b_{pq} of $\mathbf{B}$ formulated as

$$\begin{aligned} \min_{b_{pq}} \quad & f(b_{pq}) = b_{pq} \sum_{(i,j) \in \mathcal{I}} u_{ip} u_{jq} - \left(\sum_{(i,j) \in \mathcal{I}} g_{ij} \beta_{pq}^{ij} \right) \ln b_{pq} \\ \text{s.t.} \quad & b_{pq} \ge 0. \end{aligned}$$

When the network is directed, $\mathbf{B}$ is asymmetric and we set the derivative of $f(b_{pq})$ with respect to b_{pq} to 0 to get the solution as

$$b_{pq} = \frac{\sum_{(i,j) \in \mathcal{I}} g_{ij} \beta_{pq}^{ij}}{\sum_{(i,j) \in \mathcal{I}} u_{ip} u_{jq}},$$

which is nonnegative and hence satisfies the constraint. When the network is undirected, $\mathbf{B}$ is symmetric. We set the derivative of $f(b_{pq})$ with respect to b_{pq} and b_{qp} to 0 and get the solution as

$$b_{pq} = \begin{cases} \frac{\sum_{(i,j) \in \mathcal{I}} g_{ij} \beta_{pq}^{ij}}{\sum_{(i,j) \in \mathcal{I}} u_{ip} u_{jq}} & \text{if } p = q \\ \frac{\sum_{(i,j) \in \mathcal{I}} g_{ij} (\beta_{pq}^{ij} + \beta_{qp}^{ij})}{\sum_{(i,j) \in \mathcal{I}} (u_{ip} u_{jq} + u_{iq} u_{jp})} & \text{otherwise.} \end{cases}$$

3.2.1 Some Implementation Issues

Even though the derivation above appears complicated, it can actually be summarized as an elegant matrix formulation which can be implemented efficiently using some matrix-based software such as MATLAB.

For the update of $\mathbf{U}$, we need to compute the coefficients in the objective function of problem (14). We stack the coefficients for all $\{u_{pq}\}$ in three matrices, namely, $\mathbf{M}_a$, $\mathbf{M}_b$ and $\mathbf{M}_c$, which are all of size $n \times k$. Then we can compute them as

$$\begin{aligned} \mathbf{M}_a &= \frac{\mathbf{W}(\mathbf{U} + \epsilon \mathbf{E}_{nk}) \mathbf{B}^T + \mathbf{W}^T (\mathbf{U} + \epsilon \mathbf{E}_{nk}) \mathbf{B}}{\mathbf{U} + \epsilon \mathbf{E}_{nk}} \\ \mathbf{M}_b &= \epsilon (\mathbf{M}_a - \mathbf{W}^T \mathbf{E}_{nk} \mathbf{B} - \mathbf{W} \mathbf{E}_{nk} \mathbf{B}^T) + \lambda \mathbf{E}_{nk} \\ \mathbf{M}_c &= \mathbf{U} \odot \left(\frac{\mathbf{G}}{\mathbf{U}\mathbf{B}\mathbf{U}^T} \mathbf{U}\mathbf{B}^T + \left(\frac{\mathbf{G}}{\mathbf{U}\mathbf{B}\mathbf{U}^T} \right)^T \mathbf{U}\mathbf{B} \right), \end{aligned}$$

where $\mathbf{E}_{nk}$ denotes an $n \times k$ matrix of all ones, and $\frac{\mathbf{M}_1}{\mathbf{M}_2}$ for matrices $\mathbf{M}_1$ and $\mathbf{M}_2$ denotes elementwise division. When we utilize all the elements in $\mathbf{G}$, $\mathbf{W}$ becomes $\mathbf{E}_{nn}$ and we can further simplify the formulation. After getting $\mathbf{M}_a$, $\mathbf{M}_b$ and $\mathbf{M}_c$, we can easily obtain the estimate of $\mathbf{U}$.

For the update of $\mathbf{B}$, we unify the update rule for directed and undirected networks as

$$\tilde{\mathbf{B}} = \frac{\mathbf{B} \odot \left(\mathbf{U}^T \frac{\mathbf{G}}{\mathbf{U}\mathbf{B}\mathbf{U}^T} \mathbf{U} \right)}{\mathbf{U}^T \mathbf{W} \mathbf{U}},$$

where $\tilde{\mathbf{B}}$ is the updated solution for $\mathbf{B}$.

4. EXPERIMENTS

In this section, we report the empirical performance of BNMTF by comparing it with the two related NMF-based methods mentioned before, denoted by NMF_{sq} [19], which uses the squared loss, and by NMF_{kl} [18], which uses the generalized KL-divergence. The two variants of BNMTF are denoted by BNMTF_{sq} and BNMTF_{kl} which use the squared loss and generalized KL-divergence, respectively. The implementation of our method can be downloaded from `http://www.cse.ust.hk/~dyyeung/code/BNMTF.zip`.

The first performance measure is modularity which was proposed by Newman and Girvan in [14]. The modularity measure Q is defined as

$$Q(\mathbf{Y}) = \frac{1}{2m}\sum_{i,j,l}\Big(g_{ij} - \frac{d(i)d(j)}{2m}\Big)y_{il}y_{jl} = \frac{1}{2m}\text{tr}(\mathbf{Y}^T\mathbf{X}\mathbf{Y}),$$

where $d(i)$ denotes the degree of node i, y_{il} is the (i,l)th element of the membership matrix $\mathbf{Y}$, and $\mathbf{X}$ is a matrix with the (i,j)th element $x_{ij} = g_{ij} - \frac{d(i)d(j)}{2m}$. The larger the modularity, the better the performance. Since modularity was originally designed for non-overlapping community detection, only one entry in each row of $\mathbf{Y}$ is 1 while all other entries are 0. So all the methods compared need a postprocessing step to obtain $\mathbf{Y}$ as

$$y_{ik} = \begin{cases} 1 & k = \arg\max_t u_{it} \\ 0 & \text{otherwise} \end{cases}$$

where $\mathbf{U}$ is the membership matrix returned by any method compared and u_{it} is an element of $\mathbf{U}$. The modularity measure may be viewed as a criterion to measure the confidence of the community structure returned by each method. Besides modularity, we also propose another measure which is the area under curve (AUC) score based on modularity. AUC score is to measure the accuracy of the finding of multiple communities by each method. We first scale the entries in each column of the membership matrix $\mathbf{U}$ to $[0,1]$ to make the membership value of the most active node in one community as 1.We then vary a threshold from 0 to 1 and set all those entries in $\mathbf{U}$ that exceed the threshold to 1 and 0 otherwise. Finally we compute the modularity $Q(\mathbf{U})$ and also the AUC score. We use several benchmark datasets[3] (see Table 5) from real-world applications for the experiments.

Table 5: Characteristics of 12 datasets.

Dataset	n	m
Book US politics	105	441
C. elegans metabolic	453	2025
American college football	115	613
Dolphins	62	159
Jazz musicians	198	2742
Les Misérables	77	254
Word adjacencies	112	425
Neural network	297	2345
Email	1133	5451
Coauthorships in network science	1589	2742
Power grid	4941	6594
High-energy theory collaborations	8361	15751

[3] `http://www-personal.umich.edu/~mejn/netdata/`

4.1 Comparison with NMF-based Methods

We first compare BNMTF with NMF_{sq} and NMF_{kl} based on the modularity and AUC score. The results are summarized in Tables 6 and 7. Using the generalized KL-divergence, BNMTF_{kl} is either comparable to or better than NMF_{kl} on every dataset with respect to both performance measures. The result is similar when the squared loss is used. We also compare the two variants of BNMTF. Out of 12 benchmark datasets, BNMTF_{kl} outperforms BNMTF_{sq} on eight in terms of modularity measure while for AUC score, BNMTF_{kl} outperforms BNMTF_{sq} on seven datasets. The result seems to suggest that both loss functions are comparable in performance.

4.2 Comparison of Two Strategies to deal with Sparsity

The methods BNMTF_{kl} and BNMTF_{sq} utilize the whole graph. Here we also consider another setting in which the loss function is defined only on the edges with nonzero weights, i.e., Eqs. (4) and (5). The corresponding methods are denoted by sBNMTF_{sq} and sBNMTF_{kl}. The results are also shown in Tables 6 and 7. We note that sBNMTF_{kl} and sBNMTF_{sq} are generally better than BNMTF_{kl} and BNMTF_{sq} in terms of the AUC score but worse in terms of modularity. This suggests that sBNMTF_{kl} and sBNMTF_{sq} seem to be better for identifying the community structure but worse for finding the most confident communities. Moreover, the performance of sBNMTF_{kl} and sBNMTF_{sq} is unsatisfactory in some cases, e.g., on the 'Word adjacencies' dataset. One possible reason for this is that the network is so sparse that there are only very few edges with nonzero weights.

4.3 Sensitivity Analysis

The free parameters in BNMTF include the rank parameter k, the regularization parameter λ, and ϵ which is added to the denominator in BNMTF_{kl}. Here we conduct some experiments on the 'Dolphins' and 'American college football' datasets to study the sensitivity of these parameters.

We vary the rank parameter k from 3 to 100 and the results are shown in Figures 1(a) and 1(b). When k does not exceed 10, the performance change is small. The performance deteriorates very fast as k increases. This is not surprising because the total number of communities is generally very small.

For the regularization parameter λ, we vary it from 0.01 to 100. Figures 2(a) and 2(b) show the results. We find that the performance is very stable except with some slight degradation as λ approaches 100. Because the performance is not sensitive to λ, setting its value is quite easy.

Figures 3(a) and 3(b) show the results when ϵ varies from 0 to 1. The performance increases significantly as it increases from 0 to 0.2, showing that a nonzero value can help to enhance the numerical stability of BNMTF. Beyond 0.2, the performance is generally insensitive to its value and hence its value is also easy to set.

5. CONCLUSION

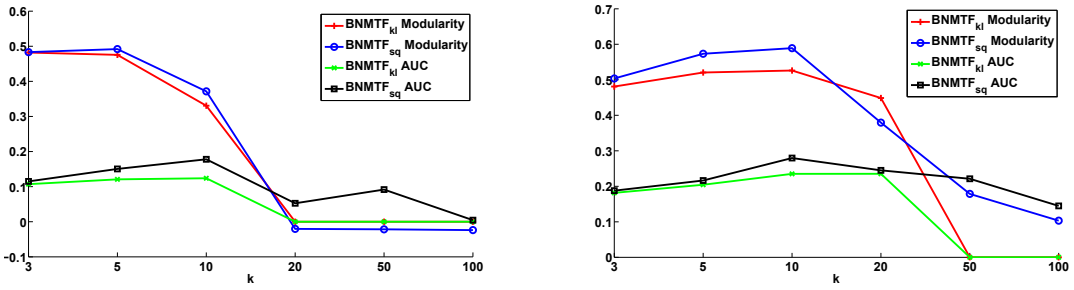
By using matrix tri-factorization, BNMTF explicitly models the community membership of each node and the interaction among communities, making it outperform other NMF-based methods in our empirical comparative study.

Table 6: Comparison in terms of modularity.

Dataset	NMF_{kl}	BNMTF_{kl}	sBNMTF_{kl}	NMF_{sq}	BNMTF_{sq}	sBNMTF_{sq}
Book US politics	0.4051	**0.4802**	0.3893	0.4613	**0.4924**	0.4530
C. elegans metabolic	0.1146	**0.1399**	0.1111	**0.1445**	0.1135	0.0925
American college football	0.5566	**0.5584**	0.4789	0.5584	**0.5733**	0.4315
Dolphins	0.4125	**0.4740**	0.3544	**0.5067**	**0.5067**	0.4125
Jazz musicians	0.2024	**0.2184**	0.2070	0.1133	0.1118	**0.1655**
Les Misérables	0.1565	**0.2146**	0.1772	0.1247	0.1031	**0.1763**
Word adjacencies	-0.0948	**0.1459**	0.0099	0.2539	**0.2677**	-0.0003
Neural network	0.1544	**0.1877**	0.1365	0.0647	**0.1021**	0.0654
Email	0.4992	**0.5108**	0.4210	0.4854	**0.4950**	-0.0020
Coauthorships in network science	0.6936	**0.7827**	0.6835	0.6607	**0.7413**	0.5552
Power grid	0.1530	**0.4646**	0.1217	0.3417	**0.3682**	0.3012
High-energy theory collaborations	0.4803	**0.6053**	0.4444	0.5648	**0.6004**	0.5413

Table 7: Comparison in terms of AUC score.

Dataset	NMF_{kl}	BNMTF_{kl}	sBNMTF_{kl}	NMF_{sq}	BNMTF_{sq}	sBNMTF_{sq}
Book US politics	0.0981	0.1024	**0.2352**	0.1415	0.1527	**0.2000**
C. elegans metabolic	0.0028	0.0044	**0.0188**	0.0036	0.0026	**0.0268**
American college football	0.2135	0.2103	**0.2178**	0.2125	**0.2158**	0.0180
Dolphins	0.1118	0.1284	**0.1984**	0.1504	0.1501	**0.1911**
Jazz musicians	0.0416	0.0848	**0.1179**	0.0710	0.0620	**0.2219**
Les Misérables	0.0339	0.0809	**0.0877**	0.0679	0.0529	**0.0896**
Word adjacencies	-0.0262	**0.0208**	-0.0025	0.0513	**0.0560**	-0.0009
Neural network	0.0261	0.0285	**0.0344**	0.0061	0.0127	**0.0278**
Email	0.0782	0.0790	**0.1477**	**0.0829**	0.0808	0.0706
Coauthorships in network science	0.0394	0.0430	**0.0431**	0.0175	0.0162	**0.0487**
Power grid	0.0187	**0.0385**	0.0101	0.0159	**0.0168**	0.0149
High-energy theory collaborations	0.0231	**0.0258**	0.0160	0.0053	**0.0181**	0.0172



(a) Dolphins (b) American college football

Figure 1: Sensitivity analysis with respect to rank parameter k on 'Dolphins' and 'American college football' datasets.

Currently, BNMTF requires the maximum number of communities to be set in advance. One possible extension in the future is to allow the actual number of communities to be determined automatically. For example, we may pursue a probabilistic reformulation of BNMTF so that methods such as automatic relevance determination [1] can be incorporated to learn the number of communities. Moreover, the current paper focuses on the static community detection problem. Another interesting direction is to extend BNMTF to dynamic community detection [15] by modeling dynamic network evolution over time.

Acknowledgments

This research has been supported by General Research Fund 621310 from the Research Grants Council of Hong Kong.

6. REFERENCES

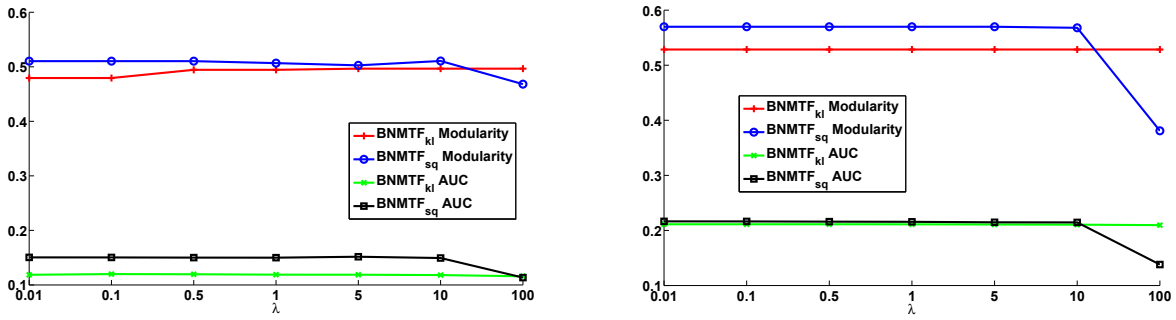
[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[2] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[3] A. Cichocki and A.-H. Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Transaction on Fundamentals*, E92-A(3):708–721, 2009.

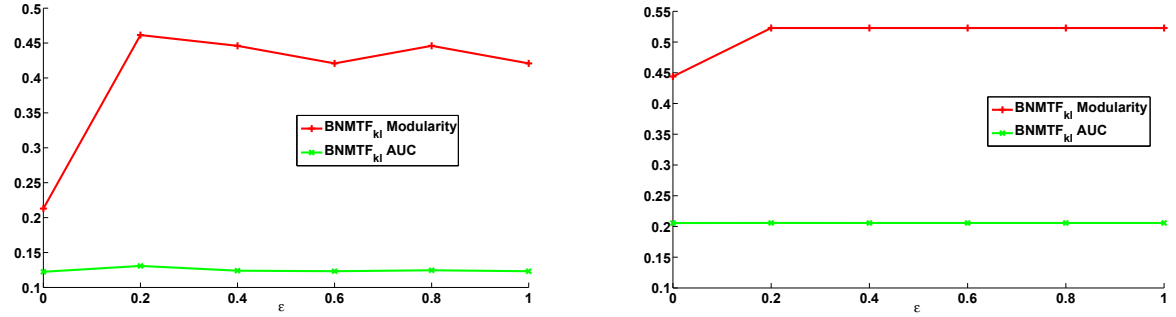
[4] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111, 2004.

[5] C. Ding, T. Li, W. Peng, and H. Park. Orthogonal



(a) Dolphins (b) American college football

Figure 2: Sensitivity analysis with respect to regularization parameter λ on 'Dolphins' and 'American college football' datasets.



(a) Dolphins (b) American college football

Figure 3: Sensitivity analysis with respect to ϵ on 'Dolphins' and 'American college football' datasets.

nonnegative matrix tri-factorizations for clustering. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 126–135, Philadelphia, PA, USA, 2006.

[6] I. J. Farkas, D. Ábel, G. Palla, and T. Vicsek. Weighted network modules. *New Journal of Physics*, 9:180, 2007.

[7] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[8] C.-J. Hsieh and I. S. Dhillon. Fast coordinate descent methods with variable selection for non-negative matrix factorization. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1064–1072, San Diego, CA, USA, 2011.

[9] K. Lange, D. R. Hunter, and I. Yang. Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics*, 9(1):1–59, 2000.

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

[11] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 556–562, Denver, CO, USA, 2000.

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

[13] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133, 2004.

[14] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113, 2004.

[15] G. Palla, A.-L. Barabási, and T. Vicsek. Quantifying social group evolution. *Nature*, 446:664–667, 2007.

[16] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435:814–818, 2005.

[17] G. Palla, I. J. Farkas, P. Pollner, I. Derényi, and T. Vicsek. Directed network modules. *New Journal of Physics*, 9:186, 2007.

[18] I. Psorakis, S. Roberts, M. Ebden, and B. Sheldon. Overlapping community detection using Bayesian non-negative matrix factorization. *Physical Review E*, 83, 2011.

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