Data Min Knowl Disc (2012) 25:420–449
DOI 10.1007/s10618-012-0260-3

Simultaneous classification and community detection on heterogeneous network data

Prakash Mandayam Comar · Pang-Ning Tan · Anil K. Jain

Received: 2 May 2011 / Accepted: 15 February 2012 / Published online: 14 March 2012

Abstract Previous studies on network mining have focused primarily on learning a single task (such as classification or community detection) on a given network. This paper considers the problem of multi-task learning on heterogeneous network data. Specifically, we present a novel framework that enables one to perform classification on one network and community detection in another related network. Multi-task learning is accomplished by introducing a joint objective function that must be optimized to ensure the classes in one network are consistent with the link structure, nodal attributes, as well as the communities detected in another network. We provide both theoretical and empirical analysis of the framework. We also show that the framework can be extended to incorporate prior information about the correspondences between the clusters and classes in different networks. Experiments performed on both real-world and synthetic data sets demonstrate the effectiveness of the joint framework compared to applying classification and community detection algorithms on each network separately.

Keywords Multi-task learning · Community detection · Classification · Link mining

Responsible editor: Fei Wang, Hanghang Tong, Phillip Yu, Charu Aggarwal.

P. M. Comar (✉) · P.-N. Tan · A. K. Jain
Department of Computer Science & Engineering, Michigan State University, East Lansing, MI, USA
e-mail: mandayam@cse.msu.edu

P.-N. Tan
e-mail: ptan@cse.msu.edu

A. K. Jain
e-mail: jain@cse.msu.edu

1 Introduction

The explosive growth of online social media applications have triggered a wave of studies on the analysis of network data using sophisticated data mining and machine learning algorithms. Despite the extensive literature, prior studies have focused mainly on learning a single task (e.g., community detection or classification) on a given network.

This paper departs from previous research by focusing on learning multiple tasks, specifically, classification and community detection, in multiple related networks. The motivation for this study is two-fold. First, the rapid proliferation of online social and information networks raises the question whether we can leverage network data from known information sources (say, Wikipedia) to enhance the network learning tasks. In a previous study (Mandayam Comar et al. 2010b, 2012), we have demonstrated the advantages of combining data from multiple related networks to improve community detection. However, the approach presented in Mandayam Comar et al. (2010b, 2012) considers the same learning task—community detection—on all networks. Here, we extend the analysis to the case when a different learning task is performed on each network. For example, one might be interested in leveraging the knowledge about the classes of Wikipedia editors (based on the articles they have edited) to identify the communities of users at `Digg.com`, a social news Web site. Though the correspondence between the identities of Digg users and Wikipedia editors may not be known, their community structures and classes are potentially well-aligned as they are based on the topics of articles posted or edited by the users.

Second, by comparing the classes and communities across different networks, one could potentially explain the identified communities in one network in terms of the known classes defined in another network, especially when there is high connectivity between nodes that belong to a community and its associated class. Furthermore, it is also possible to perform a comparative network analysis to identify the classes (clusters) found in one network but not the other. For example, given the known classes of users at Facebook, can we identify groups of MySpace users whose link structure and nodal behavior do not correspond to any known groups in Facebook?

One approach to solve this multi-task multi-network learning problem is to combine all the networks into one giant graph and apply the classification or community finding algorithms to the combined graph. However such an approach has many limitations. First, by applying a single algorithm to the combined graph, we have no control over the number of clusters or classes that will be found in each of its underlying networks. In particular, as will be shown in our experiments, this approach does not work well when the number of classes in one network is different than the number of communities in another network. Furthermore, it may lead to a suboptimal solution as it attempts to fit a global model to a graph that contains local networks with their own distinctive properties.

Another approach would be to perform the learning task independently on each network. An obvious limitation of this approach is that it does not fully utilize the link information from other related networks. More importantly, if the learning tasks are somewhat related and there is prior knowledge about the relationships between the clusters and classes in different networks, this approach will not be able to share and

utilize this information. This motivates the need to develop a joint classification and community detection algorithm for multiple related networks.

The main contributions of this work are as follows. First, we present a novel framework for joint classification and community detection in heterogeneous network data. The framework is applicable even when the number of classes in one network differs from the number of communities in another network. Another advantage of using the framework is that it produces a community-class correspondence matrix that represents the degree of association between the communities in one network and the known classes in another. This allows us to incorporate any prior knowledge about the relationship between the classes and communities in different networks.

A shorter version of this paper appeared in Mandayam Comar et al. (2010a). This article differs from the shorter conference version in that we have provided new insights into the workings of the iterative update formula used in our learning algorithm. We have also drawn parallels between the update formula and the label propagation algorithm (Zhu and Ghahramani 2002; Zhou et al. 2004). In addition, the Sect. 6 is updated with new results from using larger data sets. We have also included extensive experiments using synthetic data sets to assess the performance of our proposed framework under different multi-network parameter settings. In particular, these experiments help to shed light into fundamental questions such as (1) Can clustering on one network help to improve classification on another network, and vice-versa? (2) Does combining multiple related networks help in solving a task better than solving the same task independently on individual networks? (3) How does the presence of noisy links in different networks affect performance of each learning task?

The remainder of this paper is organized as follows. Section 2 discusses the past works related to mining multiple networks. Section 3 formulates the multi-task multi-network learning problem while Sect. 4 describes the proposed framework. Section 6 presents the experimental results. Conclusions are given in Sect. 7.

2 Related work

Mining network data has been an active research area in recent years (Getoor and Diehl 2005; Senator 2005). Examples of popular network mining tasks include classification, link prediction, clustering (or community detection), and influence maximization. The type of network being analyzed can be categorized in many ways. Typical categorizations include whether the network is weighted or unweighted, directed or undirected, and static or dynamic.

In this paper, we categorize a network based on the types of nodes and links present in the network. We consider a network to be *homogeneous* if all the nodes in the network are of the same type. Otherwise, it is called a *heterogeneous* network. Furthermore, a network is *mono-relational* if all the links are of the same type and *multi-relational* if the links are of different types or if they connect between nodes of different types. Thus, the types of networks studied in previous research can be classified as (1) *mono-relational homogeneous networks*, (2) *multi-relational homogeneous networks*, or (3) *multi-relational heterogeneous networks*.

Clustering or community detection is a common task performed on mono-relational homogeneous networks. Since networks are typically represented as graphs, most of the early research on community detection has focused on graph partitioning techniques (Ford and Fulkerson 1956; Bollobás 1998; Wei and Cheng 1989). For example, Flake et al. (2000, 2002) have applied such techniques for detecting communities on the World Wide Web. Other community detection techniques include those based on spectral (Newman and Girvan 2004; Newman 2006; Shi and Malik 1997) and discriminative learning approaches (Yang et al. 2009).

Another important network mining task is node classification, also known in the literature as collective classification or link-based classification. Here, given a network with labeled information on some of the nodes, the objective is to assign labels to all other remaining nodes. The node classification problem differs from traditional classification in that the entities to be classified can no longer be considered as independent and identically distributed (iid). Instead, entities (nodes) that are linked together are more likely to share the same class. A popular approach for solving such a problem is the *label propagation* technique, where the class information is propagated iteratively from labeled nodes to the unlabeled ones via a weighted link matrix. The weights on the links are constructed both using the nodal features and the known network link topology (Zhu and Ghahramani 2002; Zhou et al. 2004). Effectively, these methods perform a weighted average of the label information flowing from the neighboring nodes in the network.

The increased availability of contextual information about the nodes in a network have spurred research into the mining of multi-relational homogeneous networks, where different type of links exist between a set of homogeneous nodes. For example, given a collection of documents, one can construct several types of links between the documents; e.g., based on their co-citations, similarity of keywords in abstracts or titles, similarity of authors, etc. There have been recent efforts to identify communities in a multi-relational homogeneous network (Cai et al. 2005). Zhou et al. (2008) proposed a new method to combine multiple graphs to measure document similarities, where different factorization strategies are used based on nature of different graphs. Tang et al. (2009b) proposed a linked matrix factorization approach for fusing information from multiple graph sources. Lin et al. (2009) also investigated a similar problem using a relational hyper graph factorization approach to detect communities of users based on various social contexts and interactions. Tang et al. (2009a) has proposed several methods to integrate different graphs to improve community detection performance. Extending the label propagation technique to a multi-relational homogeneous network is straightforward as long as it is possible to define the similarity measures for different types of data. Kato et al. (2009) developed a label propagation algorithm for multi-relational homogeneous networks that automatically integrates structure information provided by the different types of links.

Network data that contain nodes of different data types are becoming increasingly common. Such heterogeneous nodes often carry rich and diverse content that need to be properly integrated and analyzed for drawing useful inference. A heterogeneous network with links that exist only between nodes of different types can be represented as a k-partite graph. This type of graphs has been successfully used to model relationships such as documents-words, products-users, blogs-bloggers, etc. Clustering bipartite

and k-partite graphs are often referred to as co-clustering or multi-way clustering in the literature. Long et al. (2005) investigated the problem of co-clustering as a matrix factorization problem and derived multiplicative update formulas for identifying the clusters. Dhillon et al. (2003) presented a framework for co-clustering that minimizes the loss in mutual information between the original joint distribution of related data and the corresponding joint distribution of the clustered data. Long et al. (2007, 2008) provided a unified framework for attributes-based clustering, semi-supervised clustering, co-clustering, and graph clustering using a probabilistic framework.

Research on mining data from multi-relational heterogeneous networks has flourished in recent years. Unlike previous work on k-partite graph mining, where links are formed between nodes of different types, a multi-relational heterogeneous network includes links that are established between nodes of the same types. For example, we can construct a multi-relational bibliographical network by combining the co-authorship network with a citation network between articles and a author–article bipartite graph. Various algorithms for mining such networks have been developed recently. Mandayam Comar et al. (2010b, 2012) showed simultaneous clustering of Wikipedia editors and articles gave more cohesive clusters than clustering each network independently. Chen et al. (2009) presented a co-classification framework for detecting Web spam and spammers in social media applications. They formalized the joint detection tasks as a constraint optimization problem, in which the relationships between users and their submitted Web content are represented as constraints in the form graph regularization. A pair of classifiers for detecting Web spam (URL) and spammers (Users) is simultaneously trained by taking into consideration the URL–URL link and user–user link structure. They demonstrated that the co-classification strategy is more effective than training the pair of classifiers independently.

None of the existing studies considered learning more than one task simultaneously on the multi-relational heterogeneous network. The merits of multi-task learning over single-task learning has been discussed in detail by Caruana (1997). Caruana says "Multi-task learning is an approach to inductive transfer that improves generalization performance by utilizing the domain information contained in the training data of other related tasks. To accomplish this, the tasks are learnt in parallel using a shared representation." Previous work on multi-task learning has primarily focused on learning multiple, related classification tasks on non-relational data (Evgeniou et al. 2005; Xue et al. 2007). To the best of our knowledge, there has not been any work on solving different tasks such as clustering and classification simultaneously on multiple networks, which is the main contribution of this work.

3 Problem statement

Let $\mathcal{G} = (\bigcup_{i=1}^{T} V_i, \mathcal{E}_s \cup \mathcal{E}_d)$ be the graph representation of a multi-relational heterogeneous network, where each V_i corresponds to a set of homogeneous nodes of a specific type and T is the number of node types. Furthermore, $\mathcal{E}_s$ is the set of links connecting the same type of nodes and $\mathcal{E}_d$ is the set of links connecting nodes of different types. The multi-relational heterogeneous network forms a k-partite graph if $\mathcal{E}_s = \emptyset$. Each set of homogeneous nodes V_i also induces a subgraph $\mathcal{G}_i = (V_i, E_i)$,

where $E_i \subseteq V_i \times V_i$ and $\mathcal{E}_s = \bigcup_{i=1}^{T} E_i$. We consider the induced subgraph $\mathcal{G}_i$ a mono-relational homogeneous network.

Definition 1 (*Multi-task Multi-network Learning*) Given a multi-relational heterogeneous network $\mathcal{G} = (\bigcup_{i=1}^{T} V_i,\ \mathcal{E}_s \cup \mathcal{E}_d)$, the multi-task multi-network learning problem is to solve T learning tasks, where each task is associated with an induced subgraph $\mathcal{G}_i = (V_i,\ E_i)$ of $\mathcal{G}$.

The T learning tasks stated in the preceding definition may correspond to the same class of learning problem or different classes of problems (e.g., classification, community detection, or link prediction). For example, one might be interested in classifying the nodes in one network while detecting communities in another. Furthermore, we consider only the case where the learning tasks are related; e.g., the classes or communities in one network are related to the classes or communities in another.

For brevity, the framework presented in this paper focuses on a multi-relational heterogeneous network with $T = 2$, though it can be generalized to graphs containing more than two types of nodes. To simplify the notation, let $\mathcal{G}_1 = (V_1,\ E_1)$ and $\mathcal{G}_2 = (V_2,\ E_2)$ be the induced subgraphs of $\mathcal{G}$ containing nodes from V_1 and V_2, respectively. Also, let $\mathcal{G}_{12} = (V_1 \cup V_2,\ \mathcal{E}_d)$ denote a bipartite graph with links connecting between nodes in V_1 to those in V_2. The heterogeneous nodes V_1 and V_2 may originate from the same domain source (e.g., Wikipedia articles and editors) or from different sources (Wikipedia editors and Digg users). The number of nodes in each network are denoted as $|V_1| = n$ and $|V_2| = m$, respectively.

This paper focuses on a multi-task learning problem in which the tasks involve classification and community detection. Without loss of generality, we assume that the community detection task is performed on network $\mathcal{G}_1$ and the classification task on network $\mathcal{G}_2$. Let k_1 be the number of communities (clusters) in $\mathcal{G}_1$ and k_2 be the number of classes in $\mathcal{G}_2$. Instead of representing the link structure of the multi-relational heterogeneous network $\mathcal{G}$ with a single adjacency matrix, we consider the following three adjacency matrices:

- A, an $n \times n$ the adjacency matrix associated with the network $\mathcal{G}_1 = (V_1,\ E_1)$, where $A_{ij} = 1$ if $(v_{1i},\ v_{1j}) \in E_1$ and zero otherwise.
- B, an $m \times m$ adjacency matrix for the network $\mathcal{G}_2 = (V_2,\ E_2)$, where $B_{ij} = 1$ if $(v_{2i},\ v_{2j}) \in E_2$ and zero otherwise.
- C, an $n \times m$ adjacency matrix for the bipartite graph $\mathcal{G}_{12} = (V_1 \cup V_2,\ \mathcal{E}_d)$, where $C_{ij} = 1$ if $(v_{1i},\ v_{2j}) \in \mathcal{E}_d$ and zero otherwise.

The preceding notation assumes that the links in the networks are unweighted. Our proposed formulation is applicable even for networks with weighted links by allowing the entries within the adjacency matrices to be non-binary valued. We also assume that the first l nodes in $\mathcal{G}_2$ are labeled while the remaining $m - l$ nodes are unlabeled. The true class information is encoded in an $l \times k_2$ binary matrix L, such that $L_{ij} = 1$ if the node $v_{2i} \in V_2$ belongs to class j and zero otherwise. A summary of the notations used throughout the paper is summarized in Table 1.

Our proposed approach generates a pseudo-label matrix $X \in \mathcal{R}^{n \times k_1}$ for the nodes in $\mathcal{G}_1$, where the ith row in X (denoted as $\mathbf{x_i}^T$) represents the degree of membership of

Table 1 Summary of notations based on the multi-relational network shown in Fig.1

Symbol	Description
$\mathcal{G}_1$	A homogeneous network for detecting communities
$\mathcal{G}_2$	A homogeneous network for classification
$\mathcal{G}_{12}$	A bipartite graph connecting nodes between $\mathcal{G}_1$ and $\mathcal{G}_2$
$v_{ij} \in V_i$	A node in network $\mathcal{G}_i$ ($i = 1$ or 2)
$\mathcal{E}_i$	The set of links in network $\mathcal{G}_i$
$\mathcal{E}_d$	The set of links in the bipartite graph $\mathcal{G}_{12}$
k_1	Number of communities in $\mathcal{G}_1$
k_2	Number of classes in $\mathcal{G}_2$
l	Number of labeled nodes in $\mathcal{G}_2$
A	An $n \times n$ adjacency matrix for network $\mathcal{G}_1$
B	An $m \times m$ adjacency matrix for network $\mathcal{G}_2$
C	An $n \times m$ adjacency matrix for bipartite graph $\mathcal{G}_{12}$
X	An $n \times k_1$ pseudo-label matrix for community membership of the nodes in $\mathcal{G}_1$
Y	An $m \times k_2$ pseudo-label matrix for class membership of the nodes in $\mathcal{G}_2$
V	A $k_1 \times k_2$ community-class correspondence matrix
L	An $l \times k_2$ true class membership matrix for the labeled nodes in $\mathcal{G}_2$

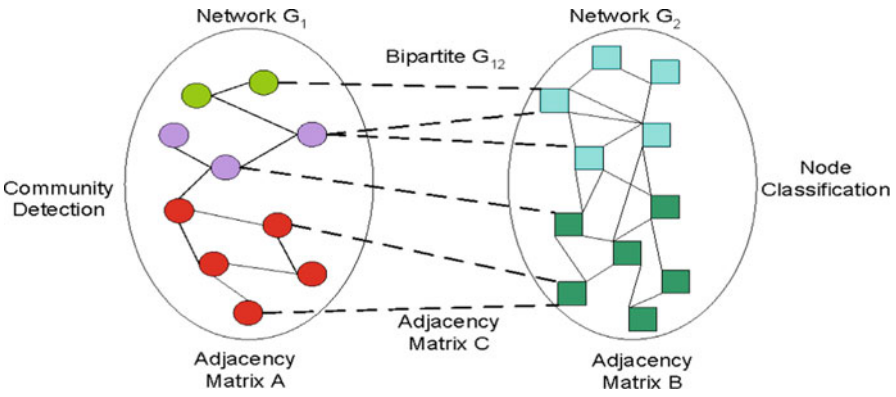

Fig. 1 An example of a multi-relational heterogeneous network

node $v_{1i} \in V_1$ in each of the k_1 clusters. Similarly, a pseudo-label matrix $Y \in \mathcal{R}^{m \times k_2}$ is generated for the nodes in $\mathcal{G}_2$, where the ith row of Y (denoted as $\mathbf{y_i}^T$) represents the degree of membership of node $v_{2i} \in V_2$ belonging to each of the k_2 classes. The pseudo-label matrices are estimated by minimizing an objective function based on the Kullback-Leibler distance defined from these matrices. The Kullback-Leibler distance between any two matrices, P and Q, is defined as

$$D(P\|Q) = \sum_{ij} P_{ij} \log\left(\frac{P_{ij}}{Q_{ij}}\right) - P_{ij} + Q_{ij} \tag{1}$$

4 Proposed joint learning framework

This section outlines our proposed framework for simultaneous clustering and classification of multiple related networks. The key idea in the proposed framework is to project the values contained in the adjacency matrices, which represents the links between nodes, into a latent feature space such that the observed link between two nodes can be expressed as weighted inner products in that space (Koren et al. 2009). In this paper, we construct this latent feature space by considering all the links available in the multi-relational heterogeneous network instead of considering them separately for each homogeneous network. The elements of the latent feature space are interpreted as the cluster and class membership vectors of the nodes in the network. The rationale behind our interpretation of the latent feature space is that a link is more likely to be formed between nodes that belong to the same community (or class) than between those in different communities (or classes).

The projection into latent feature space is accomplished by employing a matrix factorization approach (Lee and Seung 2001; Ding et al. 2006; Ho 2008) to optimize the following joint objective function:

$$\mathcal{L} = \min_{X,U,V,Y,W} D(A \parallel XUX^T) + D(C \parallel XVY^T) + D(B \parallel YWY^T) + D(\beta L \parallel Y_l) \quad (2)$$

The first term in the objective function deals with the clustering of nodes in $\mathcal{G}_1$ by factorizing the adjacency matrix A into a product involving the pseudo-label matrix X. The last two terms deal with the classification of nodes in $\mathcal{G}_2$ by estimating the pseudo-label matrix Y, taking into account both the link structure (B) and class information (L). Thus, Y_l is an $l \times k_2$ sub-matrix of Y consisting on rows of Y corresponding to the labeled data points. Thus, the last term in the objective function, $D(L \parallel Y_l)$, does not apply to unlabeled nodes in network $\mathcal{G}_2$. Meanwhile, the second term in the objective function is used to learn the relationship between the clusters found in network $\mathcal{G}_1$ and the classes obtained for network $\mathcal{G}_2$. The association between the clusters and classes are encoded by the cluster-class correspondence matrix V. The constant β indicates the factor by which Y_l would be fit to the ground truth label L. Since L is fixed binary label matrix, the constant factor β can be absorbed into it. That is, $L_{ij} = \beta$ if the node $v_{2i} \in V_2$ belongs to class j and zero otherwise. In what follows we would not explicitly mention the label factor β.

The optimization problem is solved using an alternating minimization scheme. Since there are five parameters to be estimated (X, Y, U, W, and V), we iteratively update the value of each parameter by fixing the values of the remaining four parameters. The update formula for each parameter is obtained using a gradient descent approach. Taking the partial derivative of $\mathcal{L}$ with respect to X yields the following expression

$$\frac{\partial \mathcal{L}}{\partial X_{ij}} = \sum_{a=1}^{N}\left[\frac{-A_{ia}[XU^T]_{aj}}{[XUX^T]_{ia}} + [XU^T]_{aj} - \frac{A_{ai}[XU]_{aj}}{[XUX^T]_{ai}} + [XU]_{aj}\right] + \sum_{a=1}^{M}\left[\frac{-C_{ia}[YV^T]_{aj}}{[XVY^T]_{ia}} + [YV^T]_{aj}\right]$$

The partial derivative of $\mathcal{L}$ with respect to Y_{ij} depends on whether the node v_{2i} is labeled or unlabeled. For $i \leq l$

$$\frac{\partial \mathcal{L}}{\partial Y_{ij}} = \sum_{a=1}^{M}\left[\frac{-B_{ia}[YW^T]_{aj}}{[YWY^T]_{ia}} + [YW^T]_{aj} - \frac{B_{ai}[YW]_{aj}}{[YWY^T]_{ai}} + [YW]_{aj}\right] + \sum_{a=1}^{N}\left[\frac{-C_{ai}[XV]_{aj}}{[XVY^T]_{ai}} + [XV]_{aj}\right] + \frac{-L_{ij}}{Y_{ij}} + 1$$

For $i > l$ the partial derivative is the same as above except for the last term, which is omitted since there are no label information available for the unlabeled nodes. Given their partial derivatives, the pseudo-label matrices are computed iteratively using the well known gradient descent method. For example, to estimate X, we use $X_{i,j} = X_{i,j} - \eta \frac{\partial L}{\partial X_{ij}}$. However, as shown in Lee and Seung (2001), we may transform this into a multiplicative update formula by choosing an appropriate expression for η. The resulting multiplicative update formula for X is summarized below:

$$X_{ij} = X_{ij}\frac{\sum_a\left(\frac{A_{ia}[XU^T]_{aj}}{[XUX^T]_{ia}} + \frac{A_{ai}[XU]_{aj}}{[XUX^T]_{ai}}\right) + \sum_a \frac{C_{ia}[YV^T]_{aj}}{[XVY^T]_{ia}}}{\sum_a [XU + XU^T]_{aj} + \sum_a [YV^T]_{aj}} \tag{3}$$

The update formula for Y_{ij} depends on whether the node is labeled or not.

$$Y_{ij} = \begin{cases} Y_{ij}\dfrac{\sum_a\left(\frac{B_{ia}[YW^T]_{aj}}{[YWY^T]_{ia}} + \frac{B_{ai}[YW]_{aj}}{[YWY^T]_{ai}}\right) + \sum_a \frac{C_{ai}[XV]_{aj}}{[XVY^T]_{ai}}}{\left(\sum_a [YW + YW^T]_{aj} + \sum_a [XV]_{aj}\right)}, & i > l; \\ Y_{ij}\dfrac{\sum_a\left(\frac{B_{ia}[YW^T]_{aj}}{[YWY^T]_{ia}} + \frac{B_{ai}[YW]_{aj}}{[YWY^T]_{ai}}\right) + \sum_a \frac{C_{ai}[XV]_{aj}}{[XVY^T]_{ai}} + \frac{-L_{ij}}{Y_{ij}}}{\left(\sum_a [YW + YW^T]_{aj} + \sum_a [XV]_{aj}\right) + 1}, & i \leq l. \end{cases} \tag{4}$$

Meanwhile, the update formula for matrices U, W, and V are computed as follows:

$$V_{ij} = V_{ij}\left[\frac{\sum_{a=1}^{M}\sum_{b=1}^{N}\frac{C_{ab}}{[XVY^T]_{ab}}X_{ai}Y_{bj}}{\sum_{a=1}^{M}\sum_{b=1}^{N}X_{ai}Y_{bj}}\right] \tag{5}$$

$$U_{ij} = U_{ij}\left[\frac{\sum_{a=1}^{M}\sum_{b=1}^{N}\frac{A_{ab}}{[XUX^T]_{ab}}X_{ai}X_{bj}}{\sum_{a=1}^{M}\sum_{b=1}^{N}X_{ai}X_{bj}}\right] \tag{6}$$

$$W_{ij} = W_{ij}\left[\frac{\sum_{a=1}^{M}\sum_{b=1}^{N}\frac{B_{ab}}{[YWY^T]_{ab}}Y_{ai}Y_{bj}}{\sum_{a=1}^{M}\sum_{b=1}^{N}Y_{ai}Y_{bj}}\right] \tag{7}$$

Algorithm 1 Multi-network Classification/Clustering Algorithm

Input: Adjacency Matrices A, B, C and Label Matrix L.
Output: Clustering Membership Matrices X, Y
Initialize X^{old}, Y^{old}, U^{old}, V^{old}, and W^{old} to to random matrices.
for $i = 1$ to MaxIter
  Update X^{new} according to (3) using X^{old}, Y^{old}, U^{old}, V^{old}, and W^{old}
  Update Y^{new} according to (4) using X^{new}, Y^{old}, U^{old}, V^{old}, and W^{old}
  Update V^{new} according to (5) using X^{new} and Y^{new}
  Update U^{new} according to (6) using X^{new}
  Update W^{new} according to (7) using Y^{new}
  Set $X^{old} \leftarrow X^{new}$; $Y^{old} \leftarrow Y^{new}$; $U^{old} \leftarrow U^{new}$;
    $W^{old} \leftarrow W^{new}$; $V^{old} \leftarrow V^{new}$;
end for

A summary of the pseudocode of our algorithm is given in Algorithm 1. We first randomly initialize the pseudo-label matrices X and Y to some non-negative entries. We then iteratively update the matrices X, Y, U, W, and V according to the update formula given above. The process is repeated until a specified maximum number of iterations is reached.

Theorem 1 *For a fixed Y, U, V and W the update formula for X, given in Eq. 3 monotonically decreases the objective function of the objective function defined in* (2).

The proof of the above theorem is given in the `Appendix` section. It uses the well known bound optimization technique, where the objective function (2) is bounded by an smooth auxiliary function which is then minimized. Similarly, for fixed X, U, V and W the update formula for Y given in (4) monotonically decreases the objective function and the same is true for other three update formula. Since *Algorithm 1* iteratively reduces the value of the objective function, it is guaranteed to converge to a to a stationary point. More results on convergence of the multiplicative update formulas can be found in Lin (2007). Further, assuming all the networks have similar number of nodes, communities and link densities, the complexity of the proposed algorithm is $O(n^2kT)$, where n is number of nodes, k is number of communities and T is number of iterations. The complexity of the update formulas is discussed in detail in Mandayam Comar et al. (2012).

As will be shown later, the iterative update formula presented in this section is somewhat similar to the label propagation algorithm. We highlight the advantages of our proposed multi-task learning technique compared to single-task learning with label propagation and cut-based graph partitioning algorithms in the next section.

5 Joint learning versus independent learning

In this section, we will give an insight into the exact mechanism by which the proposed framework performs the iterative clustering and classification between the networks until the community structure and classes crystalize in each of the network. We explain this from the point of view of classification problem, though similar explanation can

be made for the clustering problem. We also compared our joint learning approach to independent learning using the well-known label propagation (for classification) and graph partitioning (for clustering) algorithms.

5.1 Joint factorization versus label propagation

Consider a binary classification problem in a network where each node belongs to either class 1 or class 2. Suppose the class information is encoded in a two dimensional row vector, where $[1,\ 0]^T$ represents a node assigned to class 1 and $[0,\ 1]^T$ represents a node assigned to class 2. Unlabeled nodes are assigned a vector $[0,\ 0]^T$. A node classification algorithm can be designed to propagate a fraction of the class information from a labeled node to its neighbors at each iteration. An unlabeled node will sum up the label vector it receives from each of its neighbor. After a sufficient number of iterations, an unlabeled node in the network whose label vector is denoted by $[l_i^1,\ l_i^2]$ will be assigned to class 1 if $l_i^1 > l_i^2$ and to class 2 otherwise. This is the learning strategy employed by a broad class of *label propagation* algorithms (Zhu and Ghahramani 2002; Zhou et al. 2004).

Two important parameters that govern the propagation of labels between nodes in a network are (1) *propagation structure* and (2) *rate of propagation*. The former refers to the link structure that defines the neighborhood structure of each node in the network while the latter determines the fraction of amount by which the label information is propagated to neighboring nodes at each iteration. The *rate of propagation* is usually modeled as a function of the weight associated with the link between an adjacent pair of nodes. More formally, the update formula for the label propagation algorithm is given by

$$Y^{(t)} \leftarrow \alpha \mathbf{P} Y^{(t-1)} + (1-\alpha)\mathbf{L}, \tag{8}$$

or equivalently,

$$Y_{ij}^{(t)} = \alpha \sum_a \mathbf{P}_{ia} Y_{aj}^{(t-1)} + (1-\alpha) L_{ij} \tag{9}$$

where $\mathbf{P}$ is the propagation matrix constructed from the adjacency matrix of the network. There are several forms of label propagation matrices proposed in the literature, including $\mathbf{P}=\mathbf{D}^{-1}\mathbf{B}$ (Zhu and Ghahramani 2002) and $\mathbf{P}=\mathbf{D}^{-\frac{1}{2}}\mathbf{B}\mathbf{D}^{-\frac{1}{2}}$ (Zhou et al. 2004), where $\mathbf{B}$ is the adjacency matrix and $\mathbf{D}$ is a diagonal matrix who diagonal entries are $\mathbf{D}_{ii} = \sum_j \mathbf{B}_{ij}$. Next, we show that the multiplicative update formula given by our matrix factorization framework have similar mathematical form, which suggests that our multiplicative update formula can be viewed as a form of multi-task multi-network label propagation.

Consider the problem of classification on a single network $\mathcal{G}_2$ without using the information from the bipartite graph $\mathcal{G}_{12}$. The objective function involves minimizing the terms $D(B \parallel YWY^T) + D(L \parallel Y_l)$. The update formula for Y can be obtained by considering only the relevant terms in Eq. 4. Assuming the links in the network are

undirected, this imposes symmetry restriction on the matrices B and W. Furthermore, let $M_{ij}=\frac{B_{ij}}{[YWY^T]_{ij}}$, which represents a weighted adjacency matrix, whose large weights are associated with links that were incorrectly predicted by YWY^T in the previous iteration. Hence, the update formula Y for labeled nodes can be re-written as

$$\begin{aligned} Y_{ij} &= Y_{ij}\frac{\sum_a \left(\frac{B_{ia}[YW^T]_{aj}}{[YWY^T]_{ia}} + \frac{B_{ai}[YW]_{aj}}{[YWY^T]_{ai}}\right) + \frac{L_{ij}}{Y_{ij}}}{\sum_a (YW + YW^T) + 1} \\ &= Y_{ij}\frac{\sum_a M_{ia}[YW]_{aj} + \frac{L_{ij}}{Y_{ij}}}{\left(\sum_a [YW]_{aj}\right) + 1} \end{aligned} \tag{10}$$

Notice that the denominator term approximately normalizes the columns of the matrix YW. Let $\mathbf{D}$ be the diagonal matrix with diagonal entries consisting of the column sums of YW matrix. That is, $D_{ii} = \sum_a [YW]_{ai} + 1$. Then the column normalized label matrix is given by $\tilde{Y} = YW\mathbf{D}^{-1}$. Let Y^j represent the jth column of Y (that is, the label information of j class), then the above update formula can be written as

$$Y_{ij} = Y_{ij}\sum_a M_{ia}\tilde{Y}_{aj} + \frac{L_{ij}}{D_{jj}} \tag{11}$$

Comparing the Eqs. 9 and 11, we notice that the multiplicative update formula has the same mathematical form as the label propagation algorithm. The key difference between them is that the multiplicative updates have a varying propagation matrix M, instead of a fixed propagation matrix $\mathbf{P}$. The role of α in (9), is to establish consistency between the actual and predicted labels on the labeled data set. This is done by the factor β in (11). Recall that β is absorbed into L, i.e $L_{ij} = \beta$ if the node $v_{2i} \in V_2$ belongs to class j and zero otherwise.

The update formula for unlabeled nodes can be similarly shown to have same mathematical form as (9) albeit with different propagation matrix. This aspect makes it very adaptive in that, the propagation matrix can be updated at each iteration to accommodate more information from the other network. At each iteration, the partial cluster information from network $\mathcal{G}_1$ and the partial label information from $\mathcal{G}_2$ determines the propagation structure and the propagation rate for the classification problem. The label information thus propagated would minimize the class ambiguity among the unlabeled nodes. The enriched class information is again propagated back for performing the clustering in $\mathcal{G}_1$.

5.2 Joint factorization versus graph cuts

The earliest algorithms on identifying communities in a given network were borrowed from the graph theory literature, where, subset of the edges were removed to induce partition on the vertex set. Given a graph $\mathbf{G}=(\mathbf{V},\mathbf{E})$, a cut $\langle M, N\rangle$ is a partitioning of the vertex set $\mathbf{V} = M \bigcup N$ of graph by removing edges between the vertex sets M and N. The size of the cut is defined as number of edges removed from the edge set $\mathbf{E}$.

Let A be the adjacency matrix of the graph **G**.

$$\text{cut}_{\langle M,\, N\rangle} = \sum_{v\in M,\, u\in N} A_{uv} \tag{12}$$

A min-cut of the graph is a cut that has a smallest cut size. The well known algorithm for solving min-cut problem is based on the *max-flow min-cut* theorem by Ford and Fulkerson (1956). The flow based algorithm for identifying graph cuts has been used to solve the community detection problem in large World Wide Web graphs (Flake et al. 2000; Flake et al. 2002).

A key hinderance in applying the cut based algorithm for community detection in a multi-relational heterogeneous network setting is that the *link density* in different networks (data sources) are different. As a result one may get trivial partitions or sub-optimal partitions. For example, consider a multi-relational heterogeneous network $\mathcal{G}$, whose adjacency matrix is constructed as follows

$$G_{ij} = \begin{cases} A_{ij} & i,\ j = 1,\ 2, \ldots, n, \\ C_{ij} & i = 1,\ 2, \ldots, n \text{ and } j = n+1, \ldots, m \\ C_{ji} & j = n+1, \ldots, n+m \text{ and } i = 1, \ldots, n \\ B_{ij} & i = n+1, \ldots, n+m \text{ and } j = n+1, \ldots, n+m. \end{cases} \tag{13}$$

If the link density in bipartite graph $\mathcal{G}_{12}$ is very low, then the cut based algorithm would chop the multi-network into individual networks and subsequently split each of the individual networks independently, thus ignoring the valuable information given in the bipartite graph $\mathcal{G}_{12}$. On the other hand, if the link density within one of the two networks is much higher than the other (including the bipartite graph), then the cut based algorithm would treat it as a dense community and split the other network repeatedly. This is illustrated graphically in Fig. 2. Such problems would not arise in the joint learning framework as the proposed objective function factorizes each network separately into their respective communities (and classes) and these latent factor variables in turn induce partition on the bipartite graph that link the two networks. Nevertheless, the link density does affect the working of the update formula in two ways. If the link density in one of the network is higher than the other, then it contributes more to the objective function than the other network. For example, in the objective function (2), if the matrix A is denser than the other two matrices, then more weights would be given in minimizing the first term making the algorithm more focussed in determining a better estimate of value X than for Y. This problem can be addressed by appropriately weighting the terms in the objective function or scaling the adjacency matrices.

5.3 Incorporating prior information

In the previous section, we described the similarity between the iterative update formulas of our joint learning framework and the label propagation algorithms. We also

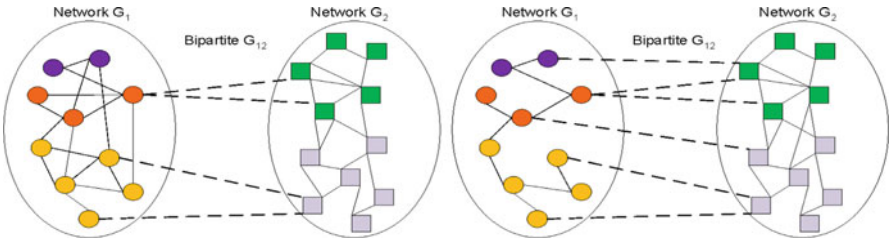

Fig. 2 Sample multi-network to illustrate the functioning of graph cut based partitioning techniques. *Left* the link density in the bipartite graph $\mathcal{G}_{12}$ is sparse. As as result, the graph cut based algorithm would first disintegrate the multi-network into individual networks and then continue to split individual networks based on their link densities. *Right* the link density is very different between network $\mathcal{G}_1$ from $\mathcal{G}_2$, as a result, one of them may get split many times while the other remain intact. Such problem wont arise in proposed joint factorization

presented the advantage of the proposed joint learning method over the traditional graph cut based methods. In this section, we present another distinct aspect of the proposed framework that allows one to incorporate the available prior knowledge about the relationship between the classes and clusters in the different networks to guide the learning process. In what follows we give a motivation for incorporating prior knowledge into the objective function.

First, we interpret the role of the V matrix in the objective function. This matrix is estimated from the KL divergence term, $D(C \parallel XVY^T)$, where C is the adjacency matrix representation of the links in the bipartite graph $\mathcal{G}_{12}$. Since X encodes the community membership of the nodes in $\mathcal{G}_1$ and Y encodes the class membership of the nodes in $\mathcal{G}_2$, we could interpret the role of matrix V as capturing the relationship between the clusters in $\mathcal{G}_1$ and the classes in $\mathcal{G}_2$. Let $\mathbf{P}$ be the prior matrix that contains the information about the link distribution between the communities (in $\mathcal{G}_1$) and the classes (in $\mathcal{G}_2$). One way to incorporate the prior information would be to add the constraint $D(V \parallel \mathbf{P})$ to the objective function (2). However, solving a quadratic optimization with quadratic constraints is a time consuming task. Instead we introduce the prior factor right into the objective function as follows.

$$\min_{X,U,V,Y,W} D(A \parallel XUX^T) + D(B \parallel YWY^T) + D(L \parallel Y_l) + D(C \parallel X(\lambda V + (1-\lambda)\mathbf{P})Y^T). \tag{14}$$

Here λ is a user-specified parameter that controls the tradeoff between fitting V directly to the data and fitting the data to the prior matrix $\mathbf{P}$. If $\lambda = 0$, then the correspondence between classes and communities is entirely determined by the prior matrix. If $\lambda = 1$, then the formulation reduces to the original framework given in (2). In situations where the actual proportion of links between the nodes from each community in $\mathcal{G}_1$ and each class in $\mathcal{G}_2$ is unknown, we may use a simple prior with P_{ij} is 0 or 1 indicating whether the ith community is expected to be related to the jth class. For example, consider a problem where there are five communities in $\mathcal{G}_1$ and three classes in $\mathcal{G}_2$. Let the prior matrix be

$$P = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

The value $P_{ij} = 1$ indicates a non empty correspondence between the ith community and the jth class. However, for the prior matrix to be effective, its entries should be scaled up according to the magnitude of V matrix that is obtained without using the prior information.

Another important use of the prior information is that often times the link structure in the network is so noisy that the algorithm fails to identify the exact correspondences between the classes and clusters. When this happens, one or more columns of the V matrix becomes a vector with all entries equal to zero. Adding the prior matrix will constrain the V matrix to have at least one non-zero entry in each column. We illustrate this in our experiment using real network data sets from Digg and Wikipedia.

6 Experimental evaluation

This section presents the results of applying the proposed framework to the multi-task learning problem on both synthetic and real-world network data.

6.1 Data sets

We have designed a multi-network simulator to generate the synthetic data for our experiments. In addition, network data from two real-world domains, Wikipedia.org and Digg.com, are used to evaluate the performance of our proposed framework.

6.1.1 Synthetic data

Our multi-network simulator constructs two networks, $\mathcal{G}_1$ and $\mathcal{G}_2$. The parameters for constructing each network is summarized in Table 2 below. We assume there is a correspondence between the clusters/classes in $\mathcal{G}_1$ and the clusters/classes in $\mathcal{G}_2$. A bipartite graph $\mathcal{G}_{12}$ is also constructed by linking the nodes in $\mathcal{G}_1$ to those in $\mathcal{G}_2$. By default, each cluster contains 100 nodes (so a network with 4 clusters will contain 400 nodes).

In this paper we investigate three main configurations of the network parameter settings. The configurations are given in Table 3. The "Noisy Network" configuration varies the proportion of noisy links (i.e., P_{12}) in $\mathcal{G}_1$ while fixing all other parameters. The "Noisy Bipartite" configuration varies the proportion of noisy links (i.e., Q_2) in the bipartite graph. In both configurations, there is a one-to-one correspondence between the clusters/classes in the two networks. For the "Uneven" configuration, the networks contain different number of clusters/classes. We have 5 clusters in $\mathcal{G}_1$ and 3 clusters in $\mathcal{G}_2$. Thus, a many-to-one correspondence is established between clusters/classes in $\mathcal{G}_1$ to those in $\mathcal{G}_2$.

Table 2 Parameters of multi-network generator

Parameter	Explanation
k_1	Number of clusters/classes in $\mathcal{G}_1$
P_{11}	Probability of within cluster link in $\mathcal{G}_1$
P_{12}	Probability of between-cluster link in $\mathcal{G}_1$
k_2	Number of clusters/classes in $\mathcal{G}_2$
P_{21}	Probability of within cluster link in $\mathcal{G}_2$
P_{22}	Probability of between-cluster link in $\mathcal{G}_2$
Q_1	Probability of link between nodes in two corresponding clusters in $\mathcal{G}_{12}$
Q_2	Probability of link between nodes in two non-corresponding clusters in $\mathcal{G}_{12}$

Table 3 Configuration for generating synthetic data

Configuration	Network $\mathcal{G}_1$			Network $\mathcal{G}_2$			Links in bipartite network $\mathcal{G}_{12}$	
	k_1	P_{11}	P_{12}	k_2	P_{21}	P_{22}	Q_1	Q_2
Noisy Network	4	0.30	**0.05–0.25**	4	0.20	0.10	0.12	0.03
Noisy Bipartite	4	0.20	0.12	4	0.20	0.15	0.30	**0.05–0.25**
Uneven	**3**	0.20	0.12	**5**	0.20	0.15	0.30	**0.05–0.25**

For each configuration, the parameter being varied is displayed in bold font

Table 4 Data category and sub-category

Category User clusters	Sub-categories Article clusters
Political Science	Civil-Rights Liberties (1068), Imperialism (1322), Nationalism (688),
Natural Science	Physics (709), Earth Sciences (607), Astronomy (780)
Computer Science	Algorithms (135), Operating Systems (454), Computer Architecture (889)
Biology	Zoology (429), Anatomy (931), Cell-Biology (806)

6.1.2 Wikipedia data

We use the Wikipedia dump from 9 Oct 2009 for our experiments. The Wikipedia articles are chosen from four broad topics—Biology, Natural Science, Computer Science and Social Science. Each of the topics are further subdivided into subtopics which are shown in Table 4. We randomly sampled 8,818 articles and 13,037 of their corresponding editors to form the Wikipedia multi-network. A visual representation of the adjacency matrices of the article and editor networks is shown in Fig. 3. Our task is to classify each article into one of the 12 possible sub-categories and to partition the editors into 4 clusters.

The Wikipedia data set is particularly challenging. Firstly, the editors do not seem to have a fixed domain of interest. As shown in the spy plot in Fig. 3, a good proportion of

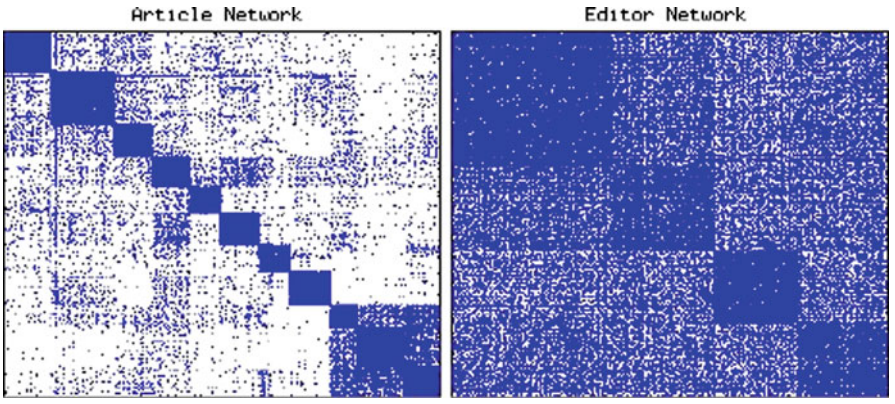

Fig. 3 Spy plot for article and user networks in Wikipedia

editors have contributed articles in all four categories. We assign a ground truth label to each user based on the category for which the user has made the most number of contributions. Secondly, the links between articles tend to be noisy. The article–article adjacency matrix given in Fig. 3 shows 9 visually distinct groups even though there are 12 sub-categories of articles. This is because the computer science articles (from the topics Algorithms, Operating Systems, and Computer Architectures) are highly connected to each other.

6.1.3 Digg data

The web site www.Digg.com is a popular social bookmarking web site where each individual users bookmark URLs and share them publicly with other users. The goal is to identify different user community based on the topics of the bookmarked URLs. As mentioned in the Sect. 1, the objective of this work is to investigate whether it is possible to combine information from several networks to improve performance of clustering and classification tasks. However, one of the challenges in demonstrating this effect empirically is the lack of ground truth information about the true clusters of various domains. So, the main reason we choose Wikipedia and Digg data sets for our experiments is the availability of the ground truth class labels that can be used to evaluate the performance of our joint community detection framework. Furthermore, though there may not be a direct correspondence between Wikipedia editors and Digg users, their community structures are often defined based on the topics of the articles they have edited or posted. Since there exists common topics among articles in both networks, this information that can be harnessed to improve their community detection tasks. Our experiment suggests that Wikipedia is a potentially useful source to improve clustering of users in a domain such as Digg.com as well. The exact data collection mechanism is described below.

We first sampled 5,670 Digg users who have bookmarked URLs on the following three topics: *Politics*, *Computer Science*, and *Natural Science*. We formed a user–user adjacency matrix (F) from the user–URL matrix. Two users are linked if they have

at least τ URLs in common. The multi-network constructed in this approach contains 4 types of nodes: Wikipedia editors, Wikipedia articles, Digg users, and Digg URLs. Each URL bookmarked at Digg.com has a title and a short description about the content of the web site. The Digg URL-word matrix and Wikipedia article-word matrix are used to establish a "weighted link" between a Digg URL and a Wikipedia article. Specifically, the weight of the link corresponds to the cosine similarity between the words in the title and description of a URL and the words that appear in the content of a Wikipedia article. In this experiment, we considered only those Wikipedia users who have edited articles in the three topics mentioned above.

6.2 Baseline algorithms and evaluation metrics

As a baseline, we use the normalized cut (`Ncut`) algorithm by Shi and Malik (1997) for clustering and the label propagation algorithm with local and global consistency (`LGC`) by Zhou et al. (2004) for classification. For a fair comparison, we applied each baseline algorithm on the entire multi-network $\mathcal{G}$ (instead of $\mathcal{G}_1$ and $\mathcal{G}_2$ separately). This is accomplished by constructing a single adjacency matrix G for the entire network using (13). In each experiment, we set the proportion of labeled nodes for the classification problem in one of the two network to be 0.2. The label factor β is set to 0.5 for synthetic data and set to 50 for Wikipedia, Digg data. We use the normalized mutual information (NMI) measure to evaluate clustering results and accuracy to evaluate classification results. Since we perform the classification task as semi-supervised clustering, the accuracy measure is computed differently. Here , each estimated cluster is assigned to the ground truth class to which is most frequent in the cluster. If rows of the confusion matrix represents the ground truth class and columns represent the estimated clusters, then we sum the maximum values across each column and divide by total number of points. Note that if the confusion matrix is diagonal heavy (maximum of each row/column occurs in diagonal) then this measure is same as regular accuracy measure.

We denote our proposed multi-task, multi-network learning framework as `Joint` or `Joint + Prior` in the remainder of this section. Since the iterative update formula converges to a locally optimal solution, the `Joint` and `Joint + Prior` algorithms were run several times, each with different random initialization values for the pseudo-label matrices. We report the results produced by the run which minimizes the objective function (2) (instead of choosing the run that maximizes the accuracy of inferring the class and cluster membership of the nodes). For the synthetic data, we show the scatter plot of all the values from 15 different runs and highlight the values of the run with minimum error. We denote this run as `Joint(ME)`

6.3 Results on synthetic data

In this section, we evaluate the performance of our algorithm on the synthetic data set under four configurations that are listed in Table 3. These four configurations evaluate different aspects of learning multiple related networks.

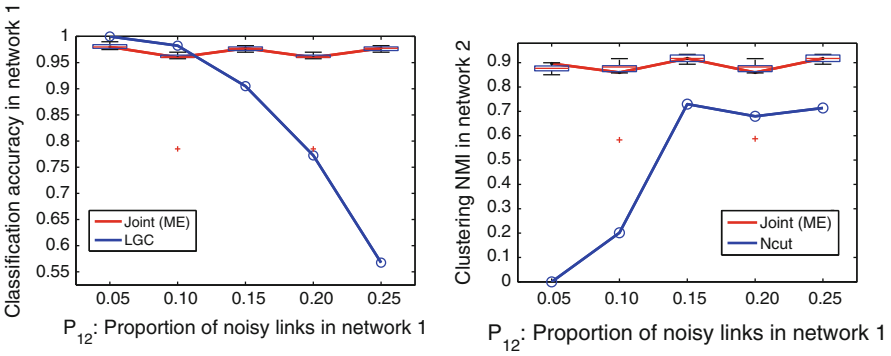

Fig. 4 Effect of varying the between-cluster link probabilities (P_{12}) on $\mathcal{G}_1$. *Left* accuracy on $\mathcal{G}_1$. *Right* NMI on $\mathcal{G}_2$

6.3.1 Varying noisy links within a network

We use the "Noisy Network" configuration in which we vary the inter cluster noise (P_{12}) in network $\mathcal{G}_1$ from 0.05 to 0.25 (with a step size of 0.05). we the performed clustering on $\mathcal{G}_1$ (classification on $\mathcal{G}_2$) and classification on $\mathcal{G}_1$ (clustering on $\mathcal{G}_2$) to study the effect of noise on both the learning task.

Classification on $\mathcal{G}_1$ Fig. 4 compares the results for `Joint` learning against the `LGC` and `Ncut` on multi-graph. The left panel shows classification accuracy on $\mathcal{G}_1$ and the right panel shows the NMI of clusters obtained from $\mathcal{G}_2$.

When the noise level is low, the `LGC` algorithm performs slightly better than `Joint`. However, at higher noise levels, the performance of `LGC` drops significantly and the joint learning performs much better. This result suggests that the explicit community information (from variable X) in $\mathcal{G}_2$ helps to discern between the classes in $\mathcal{G}_1$, whereas `LCG` is unable to use such information. Notice the inter quartile range for the box plot of accuracy values is very small, suggesting the robustness of the classification problem against the random initializations.

Surprisingly, as the noise level in $\mathcal{G}_1$ increases, the NMI of the `Ncut` algorithm also increases in $\mathcal{G}_2$. This is because of difference in the link densities between the two networks (see Sect. 5.2). When $P_{11} = 0.3$ and $P_{12} = 0.05$, $\mathcal{G}_1$ has four distinct clusters, while in comparison, the network $\mathcal{G}_2$ with high link density (with $P_{21} = 0.20$, $P_{22} = 0.10$) by itself appears as a single community in the multi-network. The `Ncut` algorithm tries to identify four communities in the multi-network. Due the difference in the link density between the two networks, the `Ncut` algorithm assign the entire $\mathcal{G}_2$ as one community and partitioned the network $\mathcal{G}_1$ into three communities. This results in lower NMI value on $\mathcal{G}_2$. The increase in the noise level in $\mathcal{G}_1$ increases the link density in the network resulting in identifying better cuts by the `Ncut` algorithm on the multi-network. When the noise parameter is $P_{12} = 0.15$, the link densities in both the network becomes comparable and the proposed `Joint` algorithm still performs better than `Ncut`.

Clustering on $\mathcal{G}_1$ For the same "Noisy Network" configuration, we performed community detection $\mathcal{G}_1$ and classification accuracy in $\mathcal{G}_2$. The results are shown in Fig. 5.

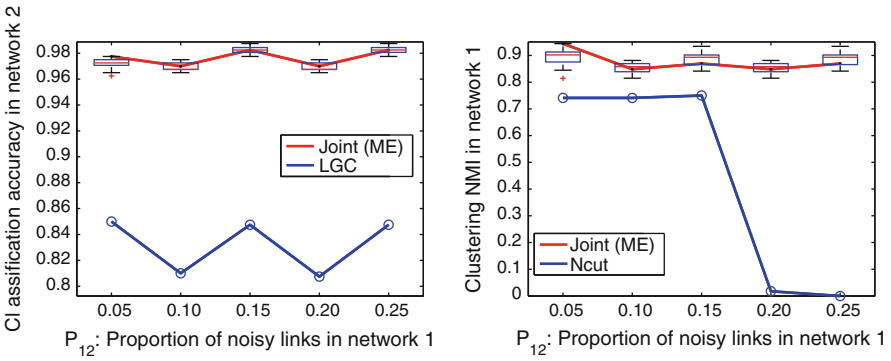

Fig. 5 Effect of varying the between-cluster link probabilities (P_{12}) on $\mathcal{G}_1$. *Left* accuracy on $\mathcal{G}_2$. *Right* NMI on $\mathcal{G}_1$

In both the cases, the `Joint` performs better than the respective baseline algorithms. As explained earlier, when the noise level is low the `Ncut` on multigraph identifies three communities on network $\mathcal{G}_1$ and the whole of $\mathcal{G}_2$ as fourth community. This resulted in NMI of 0.75. As the noise exceeded certain threshold ($P_{12} > 0.15$), the `Ncut` on multi-graph identified cuts in network $\mathcal{G}_2$ instead of $\mathcal{G}_1$. This resulted in decrease of NMI value in $\mathcal{G}_1$. Since the parameters for $\mathcal{G}_2$ are not varied, the classification results on $\mathcal{G}_2$ from `LCG` remains almost constant at 0.85 while the classification accuracy from `Joint` also remains pretty much constant at 0.98.

6.3.2 Varying noisy links between networks

We use the "Noisy Bipartite" configuration and vary q_2 from 0.05 to 0.25 in step size of 0.05. Varying the noise between the network will have no bearing on the independent clustering and classification tasks on either network. Here again, we performed both clustering and classification on the multi-relational network $\mathcal{G}$ given by (13). We then recorded the classification accuracy on subgraph $\mathcal{G}_1$ and clustering NMI on subgraph $\mathcal{G}_2$. The results are shown in Fig. 6. At lower noise levels, the `Joint` classification outperforms both `LGC` and `Ncut`. For higher noise levels in $\mathcal{G}_{12}$, the learned class information from $\mathcal{G}_1$ are not successfully transferred to $\mathcal{G}_2$ for community detection and vice versa. In terms of the objective function, the adjacency matrix C influences both the cluster membership (X) and class membership (Y) factors. As C becomes noisier, it propagates noise into factors X and Y, thus bringing down both accuracy and NMI.

6.3.3 Effect of unequal number of clusters and classes

We use the "Uneven" configuration by varying q_2 from 0.05 to 0.25 in step size of 0.05. We apply `LGC` to the entire multi-network $\mathcal{G}$ (with number of classes equal to 3) and `Ncut` to $\mathcal{G}$ (with number of clusters equal to 5). We compare their performance against the classification accuracy of `Joint` on $\mathcal{G}_1$ and NMI of `Joint` on $\mathcal{G}_2$. The results are shown in Fig. 7. Here again, the proposed `Joint` learning algorithm outperforms

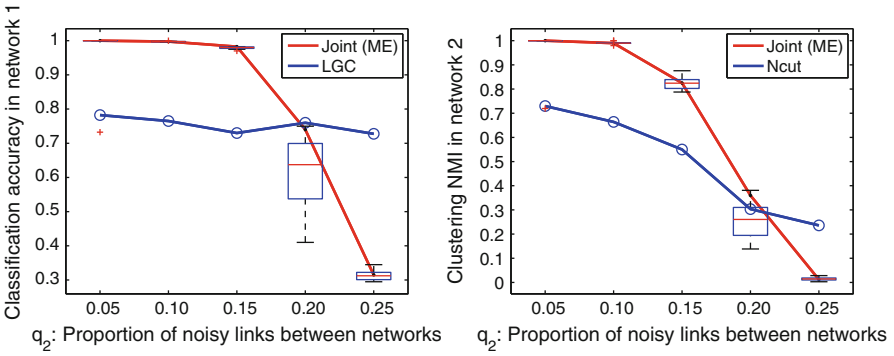

Fig. 6 Effect of varying the between-cluster link probabilities (q_2) of bipartite network $\mathcal{G}_{12}$ with same number of clusters/classes in both $\mathcal{G}_1$ and $\mathcal{G}_2$. *Left* accuracy on $\mathcal{G}_1$. *Right* NMI on $\mathcal{G}_2$

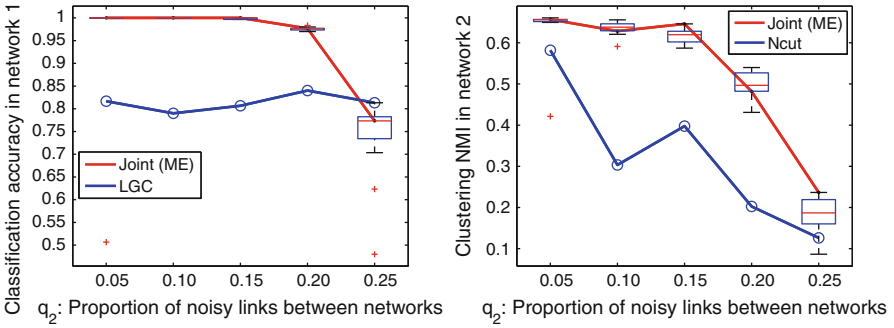

Fig. 7 Effect of varying between-cluster link probabilities (q_2) of bipartite network $\mathcal{G}_{12}$ with uneven number of clusters/classes. *Left* accuracy on $\mathcal{G}_1$. *Right* NMI on $\mathcal{G}_2$

`LGC` and `Ncut` at lower noise levels. The performance of the `Joint` is comparable to `LGC` and `Ncut` at higher noise levels.

6.4 Results on Wikipedia data

In this experiment, we perform classification task on the article network and the community detection task on the editor network. As mentioned earlier, there are four large communities in the editor network and 12 sub-topics in the article network. Here we perform four sets of experiments:

1. Apply `LGC` on article network only (with $k_1 = 12$) and `Ncut` on editor network only (with $k_2 = 4$).
2. Apply `LGC` (with 12 classes) and `Ncut` (with 4 clusters) to the entire multi-network $\mathcal{G}$.
3. Apply `Joint` to the article network (with $k_1 = 12$) and editor network (with $k_2 = 4$) without using prior information.
4. Apply `Joint` to the article network (with $k_1 = 12$) and editor network (with $k_2 = 4$) with using prior information.

Table 5 Clustering results of Wikipedia editors

User network	4 Clusters (NMI)
Ncut on editor network only	0.07
Ncut to entire multi-network	0.02
Joint(ME) without Prior	0.32
Joint without Prior	0.30 ± 0.0389
Joint(ME) with Prior	0.39
Joint with Prior	0.36 ± 0.0215

Here (ME) refers to the run with minimum error

Table 6 Classification results of Wikipedia articles

Article network	12 Classes (accuracy)
LGC on article network only	0.87
LGC on entire multi-network	0.85
Joint without Prior (ME)	0.84
Joint without Prior	0.80 ± 0.054
Joint with Prior (ME)	0.88
Joint with Prior	0.85 ± 0.021

Here (ME) refers to the run with minimum error

The results are shown in Tables 5 and 6. As shown in Table 5, the independent clustering of user network gives very bad results compared to the `Joint` approach. The cluster NMI increases from 0.07 to 0.32. Using the prior information further boosts NMI to 0.39. However, this additional gain comes at the expense of slightly reduced classification accuracy on the article network. Applying `LGC` on article network alone gives an accuracy of 0.87 which reduces to 0.85, when applied to the entire multi-network. This is because the class information provided by the article network is more useful than the "coarse-level" cluster information provided by the editor network. Furthermore, a user typically contributes to articles across different categories which makes it difficult to decide his/her actual class label. We currently assigned the user to the category to which he/she has made the most contributions. In fact, it is because of this problem, it is difficult to acquire the label information in user network, and thus, clustering becomes a necessary task.

The `Joint` approach gives an accuracy of 0.84 on the article network. The loss in accuracy in the article classification can also be explained by examining the resulting confusion matrix, as shown in Table 7. The `Joint` approach identifies four communities in the editor network. Each of these four communities have major correspondence with three article sub-categories. Therefore, the `Joint` approach settles for a local minimum solution in such a way that the resulting confusion matrix is block diagonal instead of pure diagonal. That is to say, the identification of four communities in the editor network would propagate coarse-level cluster information from the editor network to article network. This makes it harder to discern the sub-categories in the article network and hence the drop in accuracy However, using the right prior information improves the accuracy to 0.88.

Table 7 Confusion matrix—article network using `Joint` algorithm

Political Science			Natural Science			Computer Science			Biology		
753	21	29	9	27	3	215	1	1	6	1	2
14	1064	27	32	38	11	96	1	8	29	2	0
1	6	622	7	0	3	25	9	0	4	7	4
0	4	2	680	1	17	1	0	3	1	0	0
11	2	0	41	514	4	16	5	0	13	0	1
1	7	0	16	0	747	1	0	1	7	0	0
0	0	0	2	0	2	22	102	5	1	1	0
0	1	0	3	0	2	27	420	1	0	0	0
3	80	0	8	0	6	7	3	769	0	0	13
1	4	14	1	0	4	4	0	0	345	32	24
0	0	2	2	0	2	1	0	0	17	870	37
1	7	11	1	0	3	5	1	0	139	49	589

Table 8 Confusion matrix for Digg clustering

Ncut on Digg			Ncut on Digg + Wikipedia		
1171	3	474	167	1481	0
1149	0	486	997	638	0
392	0	1995	1952	435	0
NMI −0.143			NMI −0.185		

The *left panel* shows the result of `Ncut` on Digg user network. The *right panel* shows the clustering result on the multi-graph consists of both Digg users and Wikipedia editors. Only two dominant clusters were found on both

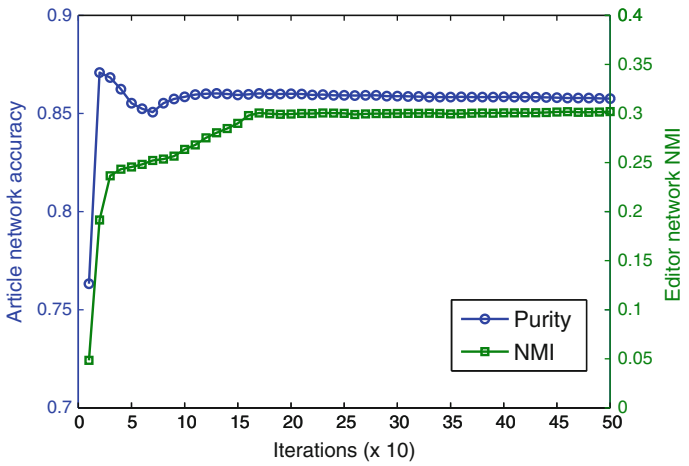

Fig. 8 Plot showing the variation of classification accuracy and clustering NMI for every 10 iterations (best viewed in color)

Table 9 Confusion matrix for Wikipedia classification

LGC on Wikipedia			LGC on Digg + Wikipedia		
1463	203	150	1688	78	50
350	690	159	710	416	73
232	100	859	450	56	685
Accuracy −0.71			Accuracy −0.66		

The *left panel* gives the confusion matrix on applying LGC on Wikipedia editor network and the *right panel* gives the confusion matrix on applying LGC on multi-graph consisting of both Digg user network and Wikipedia editor network

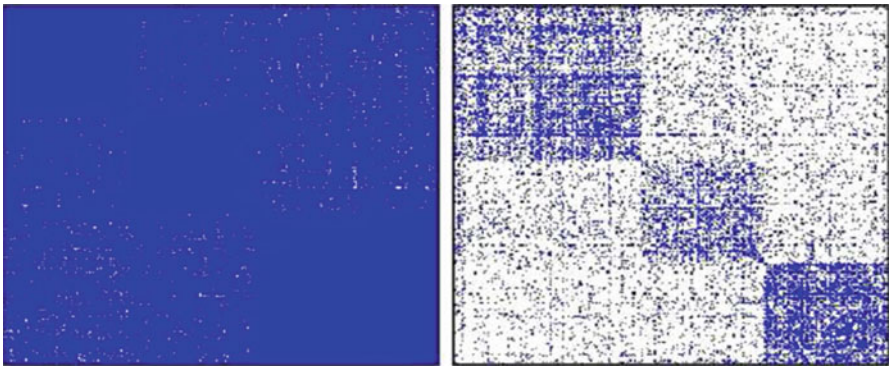

Fig. 9 Adjacency matrix plot for Digg users (*right*) and Wikipedia editors (*left*) networks (best viewed in color)

6.4.1 Number of iterations

In this section ,we discuss the effect of number of iterations on the proposed joint factorization algorithm. Figure 8 shows the variation of the Wikipedia article network accuracy and the editor network NMI with every 10 iterations. Both the values tend to stabilize in long run. It should be noted that they do not necessarily exhibit a monotone property with respect to the number of iterations. It is not easy to determine the exact number of iterations required for convergence. We run the algorithm until the error between two successive iterations is less than a specified threshold or maximum of 1,500 steps.

6.4.2 Results on Wikipedia and Digg data

Here, we linked the Digg users with Wikipedia editors based on the similarity between the words in the bookmarked URLs and words in the edited Wikipedia articles. There are three clusters in each network. The adjacency plot for these two networks is shown in Fig. 9.

We first performed Ncut on the Digg data alone and Ncut on the overall network (Digg + Wikipedia + links between them). The confusion matrix is given in Table 8.

Table 10 Confusion matrix for `Joint` with Prior information for the run with minimum error

Digg—cluster			Wiki—classify		
1246	280	122	1710	84	22
136	1278	221	298	770	131
227	103	2057	55	118	1018
NMI = 0.44			Accuracy −0.83		

As can be seen in the adjacency matrix plot, the first two clusters are noisy and heavily interlinked. So we obtain only two predominant clusters.

We apply the `LGC` algorithm to propagate labels in the Wikipedia data set. The results are summarized in Table 9. The noisy Digg data has degraded the performance of `LGC` on the Wikipedia network. Propagating labels only on Wikipedia data set gives an accuracy of 0.71, which reduces to 0.66 when applied to the multi-network.

The presence of noise on the Digg user network combined with noisy links between the Wikipedia and Digg networks resulted in poor performance of the joint learning algorithm. The number of clusters obtained is less than the number we expect. However, incorporating the prior information ($P = I_3$), ensured that we obtain three clusters on each network. The results are shown in Table 10. Clearly, the `Joint + Prior` results are significantly better than both Ncut and LGC applied to multi-network.

7 Conclusion

In this paper we have given a framework to perform mutual learning on multiple related networks. Through various set of experiments, we infer that on a collection of noisy networks, multi-task learning performs better than independent task learning on individual networks. We have also introduced the idea of using a prior to guide the clustering process. We have demonstrated a practical use of our algorithm by identifying similar communities on different network domains namely Digg and Wikipedia. Recently, researchers have used Wikipedia as knowledge source or auxiliary information source to perform several data mining operations like document clustering, web page classification, tagging and semantic relationship mining etc. (Banerjee and Ramanathan 2007; Wang et al. 2008; Wang and Domeniconi 2008; Grineva et al. 2008; Hu et al. 2009). In this paper, we show yet another novel use of Wikipedia where in we use the link information between the Wikipedia articles to perform clustering of users in other networks. In particular, we use the link information in Wikipedia to cluster users in Digg network. Incorporating nodal attributes and improving scalability of the algorithm to networks having millions of nodes are two potential directions for future research.

Acknowledgements This material is based upon work supported in part by ONR grant number N00014-09-1-0663 and ARO grant number W911NF-09-I-0566.

8 Appendix

In this section we provide the formal proofs that show the update formula (3)–(7) monotonically decrease the objective function (2). The proofs shown here has been reproduced from our previous work (Mandayam Comar et al. 2010b) of joint community detection across multiple networks.

Definition 1 $G(w,\ \acute{w})$ is an auxiliary function for F(w) if following two conditions are satisfied

$$G(w,\ \acute{w}) \geq F(w), \quad G(w,\ w) = F(w) \tag{15}$$

Lemma 1 *If* $G_1(w,\ \acute{w})$ *and* $G_2(w,\ \acute{w})$ *are auxiliary functions for* $F_1(w)$ *and* $F_2(w)$ *respectively, then* $G = G_1 + G_2$ *is the auxiliary function for* $F = F_1 + F_2$. *Further, F is non decreasing under the update formula*

$$w^{t+1} = arg \min_{w} G(w,\ w^t) \tag{16}$$

Proof The proof for G as an auxiliary function for F follows directly from definition. Below, we give the proof for second part.

$$\begin{aligned} F(w^{t+1}) &= F_1(w^{t+1}) + F_2(w^{t+1}) \\ &\leq G_1(w^{t+1},\ w^t) + G_2(w^{t+1},\ w^t) \\ &\leq G_1(w^t,\ w^t) + G_2(w^t,\ w^t) \\ &= F(w^t) \end{aligned}$$

The third line follows from the fact that w^{t+1} minimizes the auxiliary function G. Thus, $G(w^{t+1},\ w^t) \leq G(w^t,\ w^t)$ and $F(w^{t+1}) \leq F(w^t)$, which completes the proof.

Since Eq. 2 involves a summation of various distance functions $D(\cdot\|\cdot)$, Lemma 1 suggests that it is sufficient to show that minimizing the auxiliary function of the summation decreases the overall objective function to prove the correctness and convergence of our algorithm. Here we give the auxiliary function for the term $D(A\|XUX^T)$ which is quadratic in X and the auxiliary function for other terms can be similarly derived. The update formula for X which minimizes $D(A\|XUX^T)$ is given by

$$X_{ij} = X_{ij} \left(\frac{\sum_{a=1}^{N} \left(\frac{A_{ia}[XU^T]_{aj}}{[XUX^T]_{ia}} + \frac{A_{ai}[XU]_{aj}}{[XUX^T]_{ai}} \right)}{\left(\sum_{a=1}^{N} [XU + XU^T]_{aj} \right)} \right) \tag{17}$$

The objective function can be written as

$$F_1 = \sum_{i,\ j} A_{ij} \log \frac{A_{ij}}{\sum_{kl} X_{ik} U_{kl} X_{lj}^T} - A_{ij} + \sum_{kl} X_{ik} U_{kl} X_{lj}^T \tag{18}$$

Now define

$$\alpha_{k,l} = \frac{X_{ik}^{(t)} U_{kl} X_{lj}^{(t)T}}{\sum_{rs} X_{ir}^{(t)} U_{ks} X_{sj}^{(t)T}} \tag{19}$$

Clearly, $\sum_{rs} \alpha_{rs} = 1$. Using Jensens inequality we get

$$-\log \textstyle\sum_{kl} X_{ik} U_{kl} X_{lj}^T \leq -\sum_{kl} \alpha_{kl} \frac{X_{ik} U_{kl} X_{lj}^T}{\alpha_{kl}} \tag{20}$$

Substituting (20) in (18) we get $D(A \parallel XUX^T) \leq$

$$\sum_{ij} \left[A_{ij} \log A_{ij} - A_{ij} - A_{ij} \sum_{kl} \alpha_{kl} \log \frac{X_{ik} U_{kl} X_{lj}^T}{\alpha_{kl}} + \sum_{kl} X_{ik} U_{kl} X_{lj}^T \right] \tag{21}$$

plugging in α_{kl} in above equation gives the following bound on the objective function

$$\sum_{ij} \left[A_{ij} \log A_{ij} - A_{ij} - A_{ij} \sum_{kl} \frac{X_{ik}^{(t)} U_{kl} X_{lj}^{(t)T}}{\sum_{rs} X_{ir}^{(t)} U_{rs} X_{sj}^{(t)T}} \times \left(\log X_{ik} U_{kl} X_{lj}^T - log \frac{X_{ik}^{(t)} U_{kl} X_{lj}^{(t)T}}{\sum_{rs} X_{ir}^{(t)} U_{rs} X_{sj}^{(t)T}} \right) + \sum_{kl} X_{ik} U_{kl} X_{lj}^T \right]$$

which is the auxiliary function for F_1. We denote it as $G_1(X, X^{(t)})$. Now taking derivative of G_1 with respect to X_{pq} we get,

$$\frac{\partial G}{\partial X_{pq}} = -\sum_{jl} A_{pj} \frac{X_{pq}^{(t)} U_{ql} X_{lj}^{(t)T}}{\sum_{rs} X_{pr}^{(t)} U_{rs} X_{sj}^{(t)T} X_{pq}} + \sum_{jl} U_{ql} X_{lj}^{(t)T} - \sum_{ik} A_{ip} \frac{X_{ik}^{(t)} U_{kq} X_{qp}^{(t)T}}{\sum_{rs} X_{ir}^{(t)} U_{rs} X_{sp}^{(t)T} X_{pq}} + \sum_{ik} X_{ik}^{(t)} U_{kq} = 0$$

We get

$$X_{pq} = X_{pq}^{(t)} \left[\frac{\sum_j \frac{A_{pj} [U X^{(t)T}]_{qj}}{[X^{(t)} U X^{(t)T}]_{pj}} + \sum_i \frac{A_{ip} [X^{(t)} U]_{iq}}{[X^{(t)} U X^{(t)T}]_{ip}}}{\sum_{j=1}^N [U X^{(t)T}]_{qj} + \sum_{i=1}^N [X^{(t)} U]_{iq}} \right] \tag{22}$$

which is same as (17). Similarly, the term $D(C \parallel XVY^T)$ can be written as

$$F_2 = \sum_{ij} C_{ij} \log \frac{C_{ij}}{\sum_{kl} X_{ik} V_{kl} Y_{lj}^T} - C_{ik} + \sum_{kl} X_{ik} V_{kl} Y_{lj}^T$$

Now define

$$\beta_{kl} = \frac{X_{ik}^{(t)} V_{kl} Y_{lj}^{(t)T}}{\sum_{rs} X_{ir}^{(t)} V_{rs} Y_{sj}^{(t)T}} \tag{23}$$

Once again $\sum_{kl} \beta_{kl} = 1$ and following the same procedure as above, we get the auxiliary function for $D(C \parallel XVY^T)$ to be $G_2(X, X^{(t)}) =$

$$\sum_{ij} \left[C_{ij} \log C_{ij} - C_{ij} - C_{ij} \sum_{kl} \frac{X_{ik}^{(t)} V_{kl} Y_{lj}^{(t)T}}{\sum_{rs} X_{ir}^{(t)} V_{rs} Y_{sj}^{(t)T}} \log X_{ik} V_{kl} Y_{lj}^T - log \frac{X_{ik}^{(t)} V_{kl} Y_{lj}^{(t)T}}{\sum_{rs} X_{ir}^{(t)} V_{rs} Y_{sj}^{(t)T}} + \sum_{kl} X_{ik} V_{kl} Y_{lj}^T \right]$$

Taking derivative of this with respect to X_{pq} and equating it to zero, we get

$$X_{pq} = X_{pq}^{(t)} \left(\frac{\sum_{i=1}^{M} \frac{C_{pi} [YV^T]_{iq}}{[XVY^T]_{pi}}}{\left(\sum_{i=1}^{M} [YV^T]_{iq} \right)} \right) \tag{24}$$

Minimizing the original objective function (2) with respect to X, we have

$$\begin{aligned} \min_X \mathcal{J}(X) &= \min_X F_1(X) + F_2(X) \\ &\leq \min_X G_1(X, X^{(t)}) + G_2(X, X^{(t)}) \end{aligned} \tag{25}$$

Taking derivative of $G_1(X, X^{(t)}) + G_2(X, X^{(t)})$ with respect to X and equating it to zero we get update formulae (3). By Lemma 1, this objective function monotonically decreases the objective function (20). Thus we have proved Theorem 1

References

Banerjee S, Ramanathan K, Gupta A (2007) Clustering short texts using Wikipedia. In: SIGIR, pp 787–788

Bollobás B (1998) Modern graph theory, graduate text in mathematics. Springer, New York

Cai D, Shao Z, He X, Yan X, Han J (2005) Community mining from multi-relational networks. In: Proceedings of the 9th European conference on principles and practice of knowledge discovery in databases

Caruana R (1997) Multitask learning. Mach Learn 28:41–75

Chen F, Tan PN, Jain AK (2009) A co-classification framework for detecting web spam and spammers in social media web sites. In: CIKM

Dhillon IS, Mallela S, Modha DS (2003) Information-theoretic co-clustering. In: Proceedings of the 9th ACM SIGKDD int'l conf on knowledge discovery and data mining. ACM Press, New York, pp 89–98

Ding C, Li T, Peng W, Park H (2006) Orthogonal nonnegative matrix tri-factorizations for clustering. In: Proceedings of the 12th ACM SIGKDD. ACM Press, New York, pp 126–135
Evgeniou T, Micchelli CA, Pontil M (2005) Learning multiple tasks with kernel methods. J Mach Learn Res 6:615–637
Flake GW, Lawrence S, Giles CL (2000) Efficient identification of web communities. In: Sixth ACM SIGKDD international conference on knowledge discovery and data mining. ACM Press, New York, pp 150–160
Flake GW, Lawrence S, Giles CL, Coetzee FM (2002) Self-organization and identification of web communities. IEEE Comput 35:66–71
Ford L, Fulkerson D (1956) Maximal flow through a network. Can J Math 8:399–404
Getoor L, Diehl CP (2005) Link mining: a survey. SIGKDD Explor Newsl 7:3–12
Grineva M, Grinev M, Turdakov D, Velikhov P (2008) Harnessing Wikipedia for smart tags clustering. In: Proceedings of the international workshop on knowledge acquisition from the social web (KASW2008)
Ho N-D (2008) Nonnegative matrix factorization algorithms and applications. PhD Thesis, Catholic University of Louvain, June 2008
Hu X, Zhang X, Lu C, Park EK, Zhou X (2009) Exploiting Wikipedia as external knowledge for document clustering. In: Proceedings of the international conference on knowledge discovery and data mining (KDD)
Kato T, Kashima H, Sugiyama M (2009) Robust label propagation on multiple networks. IEEE Trans Neural Netw 20(1):35–44
Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. IEEE Comput 42(8):30–37
Lee DD, Seung HS (2001) Algorithms for non-negative matrix factorization. In: NIPS, vol 13, pp 556–562
Lin C-J (2007) On the convergence of multiplicative update algorithms for nonnegative matrix factorization. IEEE Trans Neural Netw 18(6):1589–1596
Lin Y-R, Sun J, Castro P, Konuru R, Sundaram H, Kelliher A (2009) MetaFac: community discovery via relational hypergraph factorization. In: Proceedings of the 15th ACM SIGKDD int'l conf on knowledge discovery and data mining, pp 527–536
Long B, Zhang ZM, Yu PS (2005) Co-clustering by block value decomposition. In: Proceedings of the 11th ACM SIGKDD int'l conf on knowledge discovery and data mining. ACM Press, New York, pp 635–640
Long B, Zhang ZM, Yu PS (2007) A probabilistic framework for relational clustering. In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining, pp 470–479
Long B, Yu PS, Zhang Z (2008) A general model for multiple view unsupervised learning. In: Proceedings of the 2008 SIAM international conference on data mining
Mandayam Comar P, Tan PN, Jain AK (2010a) Multi task learning on multiple related networks. In: Proceedings of the 19th ACM international conference on information and knowledge management, CIKM '10, pp 1737–1740
Mandayam Comar P, Tan P-N, Jain AK (2010b) Identifying cohesive subgroups and their correspondences in multiple related networks. In: Proceedings of the 2010 IEEE/WIC/ACM international conference on web intelligence and intelligent agent technology, vol 01, pp 476–483
Mandayam Comar P, Tan P-N, Jain AK (2012) A framework for joint community detection across multiple related networks. Neurocomputing 76(1):93–104
Newman M (2006) Modularity and community structure in networks. Proc Natl Acad Sci USA 103: 8577–8582
Newman MEJ, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E 69(2):026113
Senator TE (2005) Link mining applications: progress and challenges. SIGKDD Explor Newsl 7:76–83
Shi J, Malik J (1997) Normalized cuts and image segmentation. In: Proceedings of CVPR
Tang L, Wang X, Liu H (2009a) Uncovering groups via heterogeneous interaction analysis. In: IEEE international conference on data mining (ICDM '09)
Tang W, Lu Z, Dhillon I (2009b) Clustering with multiple graphs. In: ICDM, Miami, pp 1016–1021
Wang P, Domeniconi C (2008) Building semantic kernels for text classification using Wikipedia. In: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '08
Wang P, Domeniconi C, Hu J (2008) Using Wikipedia for co-clustering based cross-domain text classification. In: Proceedings of the 2008 eighth IEEE international conference on data mining, pp 1085–1090

Wei Y-C, Cheng C-K (1989) Towards efficient hierarchical designs by ratio cut partitioning. In: IEEE International Conference on Computer-Aided Design, 1989. ICCAD-89, pp 298–301. doi:10.1109/ICCAD.1989.76957

Xue Y, Liao X, Carin L, Krishnapuram B (2007) Multi-task learning for classification with Dirichlet process priors. J Mach Learn Res 8:35–63

Yang T, Jin R, Chi Y, Zhu S (2009) Combining link and content for community detection: a discriminative approach. In: Proceedings of the 15th ACM int'l conf on knowledge discovery and data mining, pp 927–936

Zhou D, Bousquet O, Lal TN, Weston J, Schölkopf B (2004) Learning with local and global consistency. In: Advances in Neural Information Processing Systems 16, MIT Press, pp 321–328

Zhou D, Zhu S, Yu K, Song X, Tseng BL, Zha H, Giles CL (2008) Learning multiple graphs for document recommendations. In: WWW '08, pp 141–150

Zhu X, Ghahramani Z (2002) Learning from labeled and unlabeled data with label propagation. CMU