电 子 科 技 大 学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

学士学位论文

BACHELOR THESIS

论文题目 **Transfer Learning for Community Detection**

专　　业	计算机科学与技术
学　　号	**2014030302014**
作 者 姓 名	黄泽熙
指 导 教 师	**Asst. Prof. Sinno Jialin Pan**

摘 要

复杂网络中的社团检测是最为重要的数据挖掘任务之一。关于多维社团检测的前人研究旨在发现适用于多层乃至所有层的社团划分。基于对现实网络的分析，本文提出另一个新的多维社团检测的目标，即通过从其他层迁移知识来提升某些层的社团划分结果。为了解决这一新的问题，本文提供了一个基于表征的多维社团检测框架：先同时学习所有层共同的特性表征和每层独有的特性表征，之后对学到的两种表征结合并进行聚类，从而得到社团划分。本文设计了一种拓展对称非负矩阵分解方法来学习表征，并利用k-means算法对表征进行聚类，从而实现了这一框架。本文的方法在多个多维网络数据集上与其他基于表征的社团检测方法进行了性能比较。结果显示，本文的方法在以模块度为依据的性能上优于其他方法，尤其是当接受迁移知识的目标层噪声较大（如远比其他层稀疏）时。通过随机去除一些目标层的边，本文还模拟了在输入数据高度不可靠情况下的性能测试。结果表明，本文提供的方法对于噪声的健壮性良好，能在去除目标层一半的边之后还能保持高达83.7%的性能，而其他方法则只能保持约50%。

关键词：复杂网络，多维社团检测，迁移学习，数据挖掘

ABSTRACT

Community detection in complex networks is one of the most fundamental data mining tasks. Previous works on multiplex community detection aim to find a community partition across several or all layers. Here, based on the observation of real-world networks, we propose another novel objective for multiplex community detection, that is, to refine community detection results in some layer with transferred knowledge from other layers. To solve this new problem, we provide a representation-based multiplex community detection framework, which first learns a shared common feature representation and layer-specific feature representations simultaneously and then cluster the combined representations for community partition. The framework is implemented with an extended symmetric non-negative matrix factorization approach for learning representations and k-means for clustering representations. This implementation is compared to other representation-based community detection algorithms on several multiplex network datasets. Experimental results show that our implementation outperforms other methods in terms of modularity, especially when the target layer which receives transferred knowledge contains much noise (e.g. sparser than other layers). We further evaluate our algorithm in highly unreliable conditions by randomly removing a set of edges in the target layer. It is shown that our algorithm is quite robust, retaining as high as 83.7% of its original performance even when half of the edges are removed, while other methods can only retain about 50%.

Keywords: complex networks, multiplex community detection, transfer learning, data mining

Contents

Chapter 1 Introduction

1.1 Background

1.1.1 Network

The modern science of networks has brought significant advances to our understanding of complex systems [1], which spans the natural, social, as well as computer science and engineering [2–4]. Networks consist of nodes (or vertices) which represent entities, and edges (or links) that mimick their interactions. Networks can be found in various real-world contexts. Internet, for example, is the physical network of computers, routers and modems which are linked via cables or wireless signals. Another famous example is Facebook①, a large online social network that connects billions of people virtually. Networks also find their existence in many other areas, include biology [5], biochemistry [6], economics [7], ecology [8], epidemiology [9], political science [10], computer science [11], social science [12], etc.

For decades, network analysis and mining focus on monoplex networks (see Figure 1-1, for example), where all edges represent a single type of interaction between nodes [13].

The most widely used model of the monoplex network is graph, denoted by $G(V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is the collection of n nodes and $E = \{(v_i, v_j)|v_i \text{ is connected to } v_j\}$ is the collection of m edges in the network. For each graph, a corresponding adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ records the connection information. In this thesis, we limit our scope to unweighted and undirected networks that contain no self-loops. For such networks, the entries of the adjacency matrix are specified by

$$\mathbf{A}_{ij} = \begin{cases} 1 & \text{if node } i \text{ and } j \text{ is connected} \\ 0 & \text{otherwise} \end{cases}, 1 \leqslant i, j \leqslant n \tag{1-1}$$

① https://www.facebook.com/

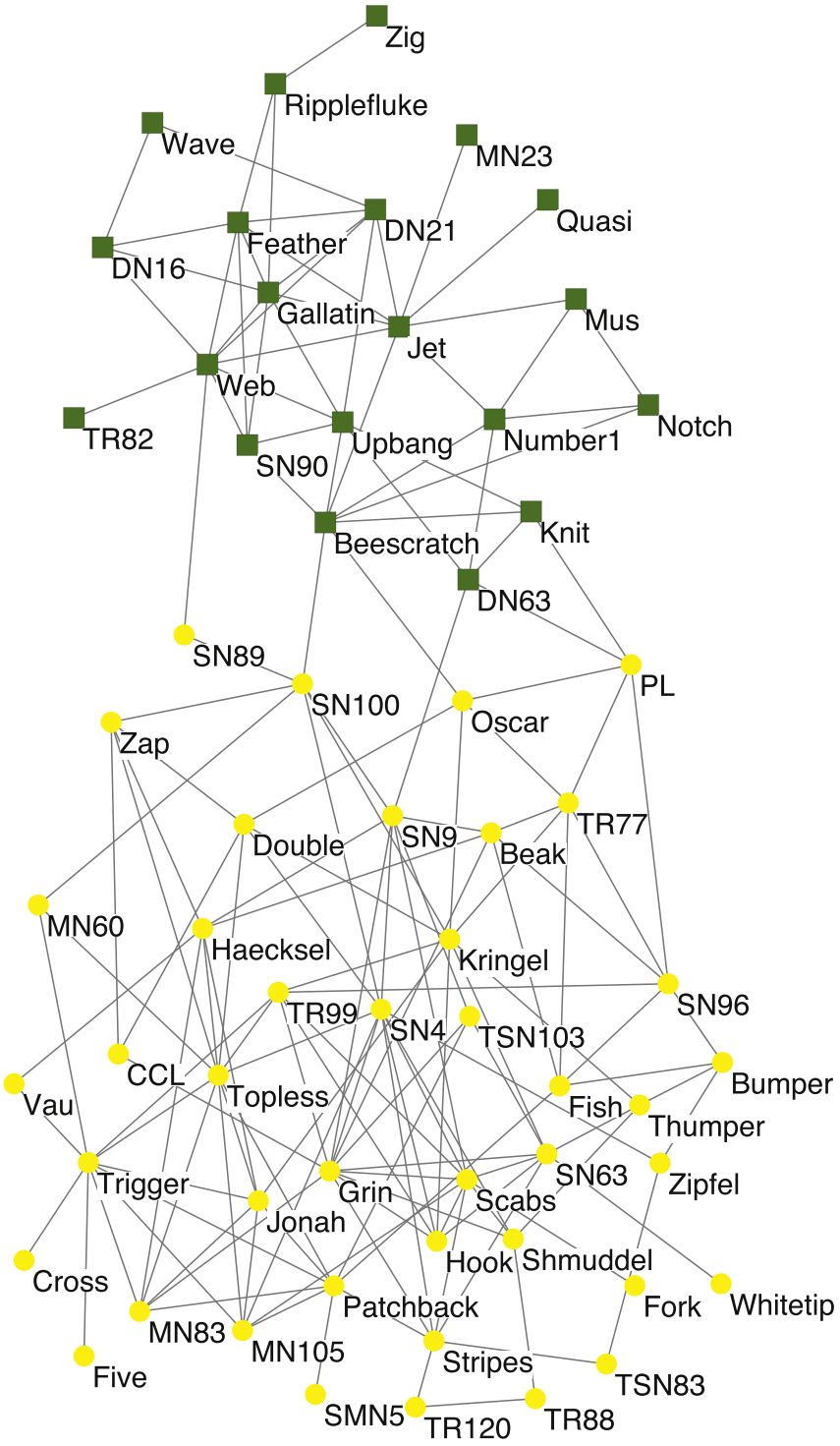


Figure 1-1 Lusseau's network of bottlenose dolphins [14]. Edges connect dolphins which have shown preferred companionship. This figure is generated by [15].

And because the network is undirected and contains no self-loop, we have

$$\mathbf{A}_{ij} = \mathbf{A}_{ji}, 1 \leqslant i, j \leqslant n \ \& \ \mathbf{A}_{ii} = 0, 1 \leqslant i \leqslant n \tag{1-2}$$

The number of edges that are connected to a node i is called the degree of i, denoted as k_i. We have

$$k_i = \sum_{j=1}^{n} \mathbf{A}_{ij} \tag{1-3}$$

The degree of nodes of real-world networks usually follows a power law distribution [16],

$$f(k) \sim k^{-\gamma} \tag{1-4}$$

where γ is a parameter whose value is typically in the range $2 < \gamma < 3$. Networks having this property is called scale-free networks [17].

Recently, however, much more efforts have been devoted to analysis of multiplex networks [18] (or equivalently referred to as multidimensional networks [19] or multi-layer networks [20]), where the same set of nodes are connected by multiple types of edges, which allows encoding richer and more complex interactions of real-world dynamic systems (see Figure 1-2, for example).

A multiplex network can be represented with several monoplex networks with a shared set of nodes. Each monoplex network is called a layer of the multiplex network. A N-layer multiplex network $G(V, E_1, ..., E_N)$ thus have a set of n nodes and N sets of edges, each with m_K edges, where $1 \leqslant K \leqslant N$. Each layer also has a respective adjacency matrix $\mathbf{A}_K$,

$$\mathbf{A}_{Kij} = \begin{cases} 1 & \text{if node } i \text{ and } j \text{ are connected in } K\text{th layer} \\ 0 & \text{otherwise} \end{cases}, 1 \leqslant i, j \leqslant n, 1 \leqslant K \leqslant N \tag{1-5}$$

Degree of nodes is computed for each layer. The degree of node i in Kth layer is

$$k_{Ki} = \sum_{j=1}^{n} \mathbf{A}_{Kij} \tag{1-6}$$

1.1.2 Community Detection

In real-world networks, a very interesting phenomenon is high concentrations of connections within special groups of nodes, and low concentrations between these groups,

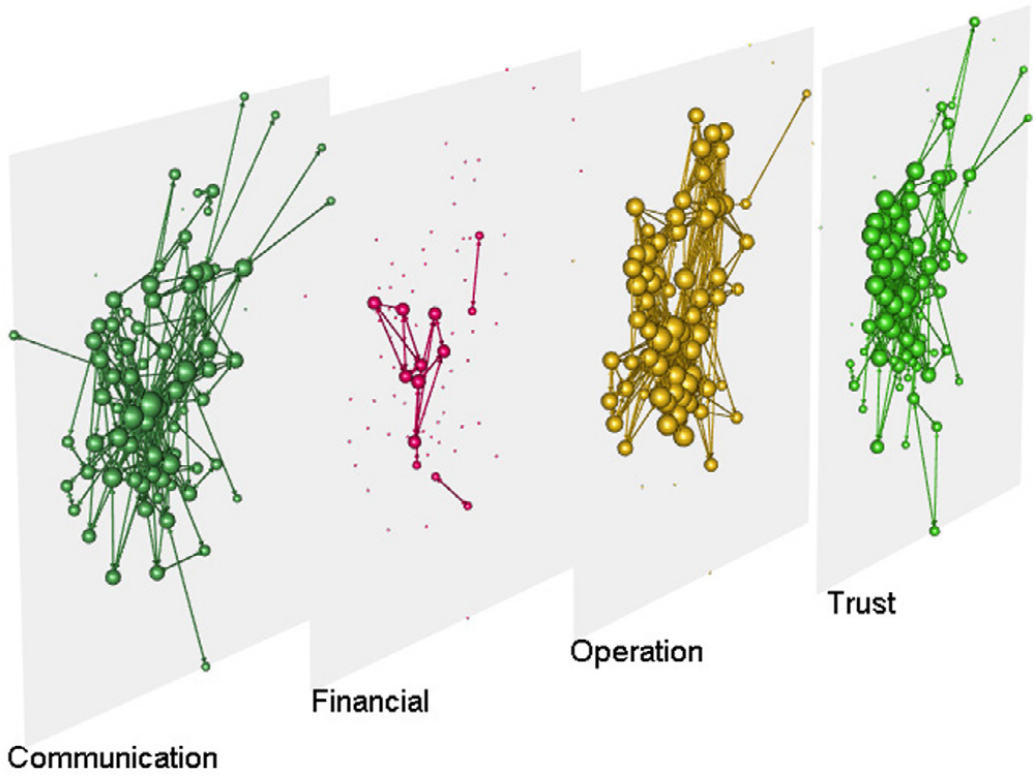


Figure 1-2 Indonesian Noordin Top terrorist multiplex network [21]. Edges in each layer represent exchanged communication, financial involvement, common operations and mutual trust. This figure is generated by [22].

which is referred to as community structure [23]. A vivid example of community structure is shown in Figure 1-3. Scientists in the same research area, i.e. community, generally collaborate more closely, while scientists from different research areas seldom work together. There are various examples of other community structures in our daily life: families, work groups, friend circles, universities, nations, to mention a few. In other networked systems, community structure is also omniscient: group of pages on related topics in World Wide Web [24, 25], collection of proteins with the same function in protein interaction networks [26, 27] and functional modules such as cycles and pathways in metabolic networks [28, 29].

Community detection in networks has various applications. For example, identifying clusters of customers with similar interests in the network of purchase relationships

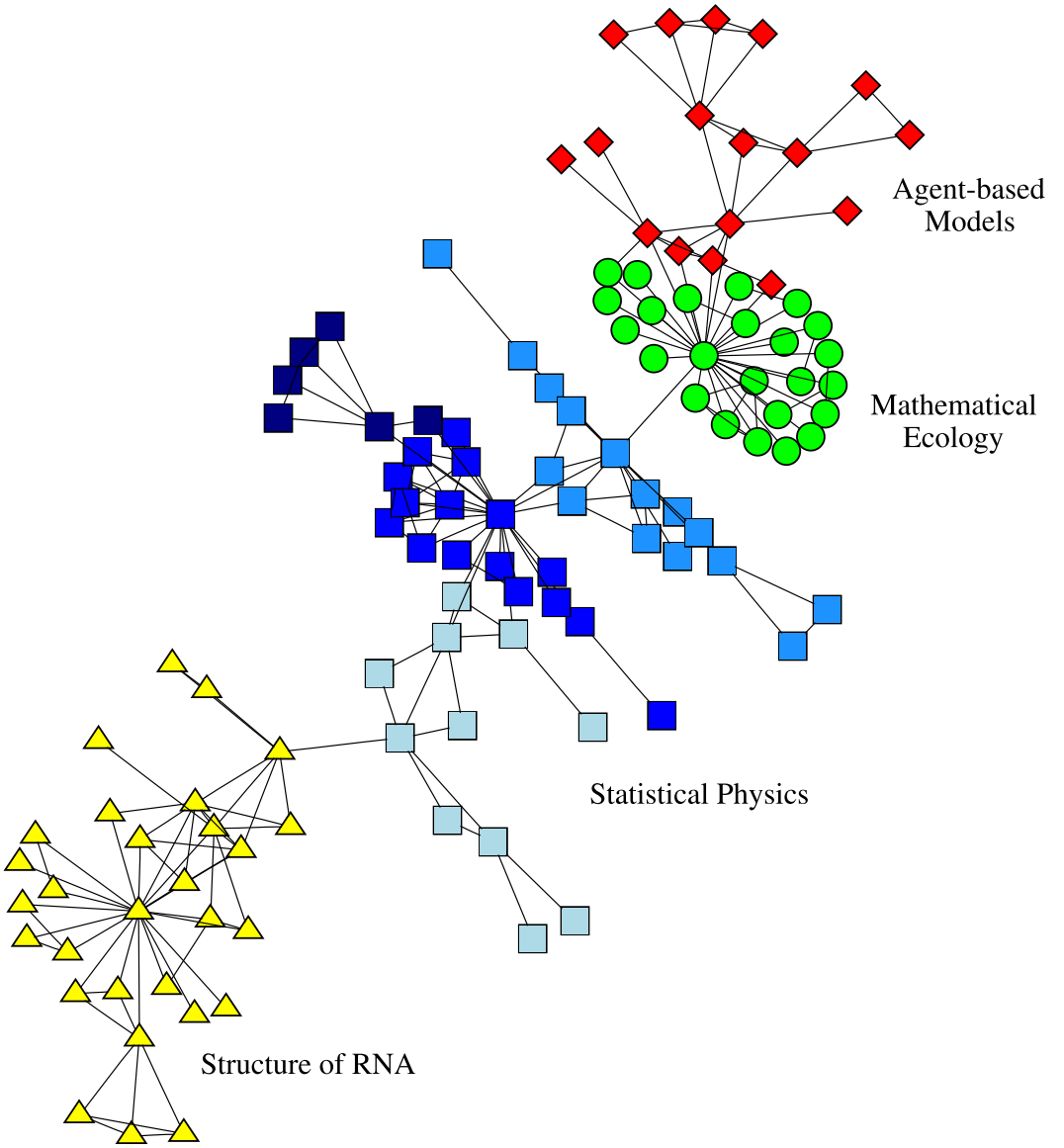


Figure 1-3 Collaboration network of scientists working at the Santa Fe Institute. Edges connect scientists who have co-authored at least one paper. Different symbols indicate different research areas of the scientists. This figure is generated by [23].

can help online retailers (e.g. Amazon[①]) set up efficient and effective recommendation systems [30]. Community detection is also important for social science analysis [31], handling navigation queries [32], establishing dedicated mirror server [33], etc.

Due to wide applicability of community detection, huge research efforts have been put into developing effective community detection algorithms. For monoplex networks, the task is to find a community partition $\mathtt{C} = \{C_1, ..., C_{k_{com}}\}$, where k_{com} is the number of communities in the network and $C_i = \{n_{i1}, ..., n_{iq_i}\}, 1 \leqslant i \leqslant k_{com}$ is a collection of nodes which are densely connected to each other, compared to nodes that don't belong to this collection. Several measures have been proposed to evaluate community detection effectiveness, such as normalized cut [34] and modularity [35]. Unfortunately, exactly optimizing these two metrics is NP-hard, which leads to numerous algorithms to heuristically solve the problem.

Traditional methods for monoplex community detection include graph partitioning (e.g. Kernighan Lin algorithm [36]) which tries to minimize number of edges between communities, hierarchical clustering [37] which iteratively merge clusters of nodes if their similarity is high (agglomerative) or iteratively split cluster of nodes by removing edges connecting vertices with low similarity (divisive) and partitional clustering (e.g. k-means [38]) which embeds nodes into a metric space and then minimize the distance between nodes and their centroids. The number of community detection algorithms is still growing fast, with representative ones such as modularity optimization [39], Louvain [40] and Infomap [41].

Unlike community detection in monoplex networks, multiplex community detection only gains its popularity recently. The objective of multiplex community detection is often defined as to find a community partition $\mathtt{C}$ for all layers of a multiplex network [42], though some recent work has argued that community structure can be significant only for a subset of layers [43].

To find community for all layers, several methods have been proposed. A straightforward idea is to to aggregate all layers and apply monoplex algorithms on the aggregated network [44]. For example, in binary aggregation, nodes are connected in the aggregated network if they are connected in any layer of the multiplex network. Frequency-based aggregation counts the number of connection in all layers and use this number as weight

① `https://www.amazon.com/`

in the aggregated network. Another approach is ensemble clustering, in which monoplex algorithms are applied to each layer of the multiplex network, and later a ensemble strategy is used to find a consensus across layers [45]. More recently, some extended versions of monoplex algorithms have sprung up to handle the multiplex problem. [46] extended the InfoMap algorithm, [47] was based on the Quick algorithm and [48] took an extended seed-centric approach. Tensor factorization approach [49] can also be viewed as an extended version of spectral algorithms for monoplex networks.

1.1.3 Transfer Learning

Transfer learning aims to extract the knowledge from one or more source task and apply the knowledge to a target task. Compared to traditional machine laerning methods which try to learn each task from scratch, transfer learning techniques try to transfer the knowledge from some source tasks to a target task (see Figure 1-4). Its application can be found in classification of web documents[50], sentiment [51] and image [52], WiFi localization [53], computed aided design [54], name-entity recognition [55], to mention a few.

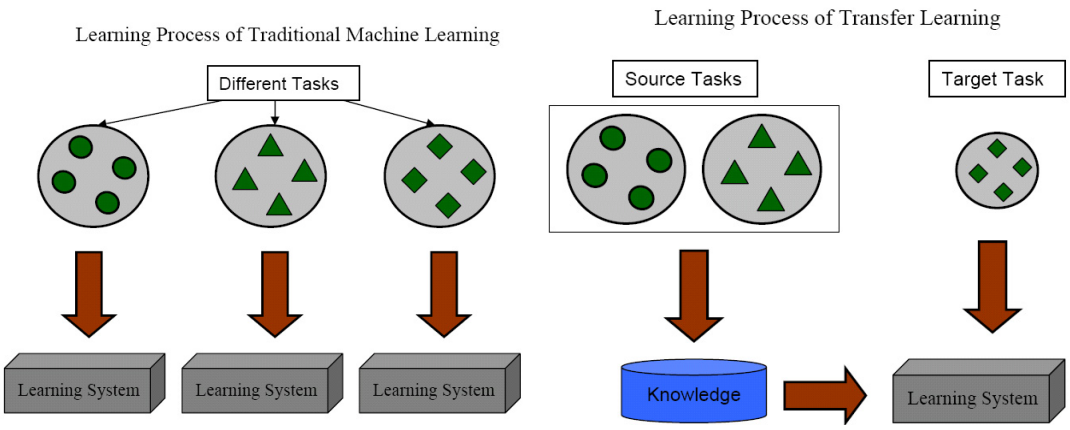


Figure 1-4 Different learning processes between traditional machine learning and transfer learning. This figure is generated by [56].

Depending on whether source and target domains or tasks are the same, transfer learning can be categorized into three sub-setting [56]. In inductive transfer learning setting, the target task is different from the source task, no matter when the source and target domains are the same or not. Classical algorithms for inductive transfer learning include

TrAdaBoost [57], MT-IVM [58] and TAMAR [59]. In the transductive transfer learning setting, the source and target tasks are the same, while the source and target domains are different. KMM [60], SCL [61] and MMDE [53] are some widely-used algorithms for transductive transfer learning. In the unsupervised transfer learning setting, the target task is different from but related to the source task, and it focuses on unsupervised learning tasks such as clustering. For unsupervised transfer learning, STC [62] is proposed to transfer clustering problem and TDA [63] is developed for transfer dimensionality reduction problem.

1.2 Contribution

The major contributions of our work are fourfold:

1. We propose a novel objective for multiplex community detection, i.e., to refine community detection results in some layer with transferred knowledge from other layers, based on our observation of real-world networks. To the best of our knowledge, we are the first to formulate the multiplex community detection problem in this way.
2. To achieve this new objective, we design a representation-based multiplex community detection framework, which first learns a shared common feature representation and layer-specific feature representations simultaneously and then cluster the combined representations for community partition. This framework can easily accommodate to different scenarios by selecting different learning and clustering algorithms.
3. We provide an implementation of the framework, which includes an extended symmetric non-negative matrix factorization approach for learning representations and k-means for clustering representations. We theoretically prove that under the update rules of our learning algorithm, the solution converges to a Karush-Kuhn-Tucker stationary point, if it converges.
4. We compare our algorithm to other representation-based community detection algorithms on several multiplex network datasets, which shows that our algorithm outperforms the others, especially when the target layer is noisy (e.g. much sparser

than other layers). We further verify this property by testing our algorithm on networks with removed edges.

1.3 Thesis Structure

The rest of this thesis is structured as follows: we unify the representation-based community detection methods with a shared procedure and introduce several different models of representation-based community detection in our next chapter. Chapter 3 presents our own work, which includes the formulation of our objective, our framework and its implementation, as well as experiments and results. Finally, we summarize this thesis and discuss some future research directions in Chapter 4.

Chapter 2 Representation-based Community Detection

2.1 A Shared Procedure

Representation-based community detection techniques are a collection of methods for monoplex community detection. They have a shared procedure as follows: First, they define some objective function on the network $G(V, E)$ to optimize. Then, some utility matrix $\mathbf{U}$ of the graph depending on the objective function is projected into some latent space in the optimization process, resulting in some latent feature representation $\mathbf{H}$. Finally, some clustering algorithm is applied to the learned latent representation $\mathbf{H}$ to get the community partition result $\mathtt{C}$. This shared procedure is illustrated in Figure 2-1.

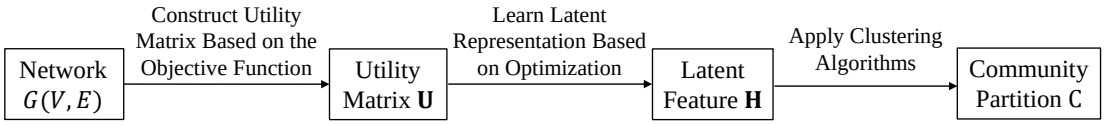


Figure 2-1 The shared procedure of representation-based community detection methods.

In the sections that follow, we will introduce different models of representation-based community detection.

2.2 Stochastic Model

In stochastic model [64], the entry of adjacency matrix $\mathbf{A}_{ij}$ is viewed as the probability that node i and j are connected, which can be further considered to be determined by the probability that this pair of nodes generate edges belonging to the same community. Denote $\mathbf{H} \in \mathbb{R}^{n \times r}$ as the latent variables (where r is the number of dimensions in the latent space), such that $\mathbf{H}_{ik}$ represents the probability that node i generates an edge belonging to kth community. Then, the probability that node i and j is connected by an edge in kth community is $\mathbf{H}_{ik}\mathbf{H}_{jk}$, and the probability they are connected can thus be expressed as

$$\mathbf{A}_{ij} = \sum_{k=1}^{r} \mathbf{H}_{ik}\mathbf{H}_{jk} \tag{2-1}$$

As a result, the community detection problem can be formulated as a non-negative matrix factorization,

$$\min_{\mathbf{H}\geqslant 0} L = ||\mathbf{A} - \mathbf{H}\mathbf{H}^T||_F^2 \tag{2-2}$$

Then, clustering algorithms can be applied to the latent variables $\mathbf{H}$ to obtain the community partition. Note that the utility matrix in stochastic model is simply the adjacency matrix $\mathbf{A}$.

2.3 Latent Space Model

The latent space model [65] maps the nodes in a network into a low-dimensional Euclidean space such that the proximity between the nodes based on network connectivity are kept in the latent space. Introduce $\mathbf{P} \in \mathbb{R}^{n\times n}$ as the proximity matrix for the network with $\mathbf{P}_{ij}$ denoting the distance between node i and and node j. When the distance measure is specified (e.g. geodesic distance [66]), $\mathbf{P}$ can be computed from the network adjacency matrix. Now, denote $\mathbf{H} \in \mathbb{R}^{n\times r}$ as the coordinates of nodes in the latent space, we have [67]

$$\mathbf{H}\mathbf{H}^T = -\frac{1}{2}(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T)(\mathbf{P}\circ\mathbf{P})(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T) = \widetilde{\mathbf{P}} \tag{2-3}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{1} = (1, ..., 1)$ and $\circ$ represents entrywise multiplication. This can be formulated as the matrix factorization problem below,

$$\min L = ||\widetilde{\mathbf{P}} - \mathbf{H}\mathbf{H}^T||_F^2 \tag{2-4}$$

Then, clustering algorithms can be applied to the coordinates in latent space $\mathbf{H}$ to obtain the community partition. Note that the utility matrix in stochastic model is $\widetilde{\mathbf{P}}$.

2.4 Spectral Clustering Model

Spectral clustering [68] is related to graph partition which minimizes the number of edges between communities, with some constraints to avoid singletons (i.e. communities only consisting of a single node). Their objective can be formulated as a min-trace problem,

$$\min_{\mathbf{H}^T\mathbf{H}=\mathbf{I}} L = \mathrm{Tr}(\mathbf{H}^T\widetilde{\mathbf{L}}\mathbf{H}) \tag{2-5}$$

where $\mathbf{H} \in \mathbb{R}^{n \times r}$ represent latent features and $\widetilde{\mathbf{L}}$ is graph Laplacian and can be computed by the adjacency matrix based on different types of constraints. For example,

$$\widetilde{\mathbf{L}} = \begin{cases} \mathbf{D} - \mathbf{A} & \text{(Ratio Cut)} \\ \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} & \text{(Normalized Cut)} \end{cases} \tag{2-6}$$

where $\mathbf{D}$ is the degree matrix of the network

$$\mathbf{D}_{ij} = \begin{cases} k_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \quad 0 \leqslant i, j \leqslant n \tag{2-7}$$

$\widetilde{\mathbf{L}}$ is the utility matrix for spectral clustering and rows of $\mathbf{H}$ can be clustered to get the community partition.

2.5 Modularity Optimization Model

Modularity is a widely-accepted measure of community detection effectiveness. Maximizing modularity on a network is equivalent to the following max-trace problem [69],

$$\max_{\mathrm{Tr}(\mathbf{H}^T\mathbf{H})=n} Q = \mathrm{Tr}(\mathbf{H}^T\mathbf{M}\mathbf{H}) \tag{2-8}$$

where $\mathbf{H} \in \mathbb{R}^{n \times r}$ is the community indicator matrix and $\mathbf{M}$ is the modularity matrix, with

$$\mathbf{M}_{ij} = \mathbf{A}_{ij} - \frac{k_i k_j}{2m}, 1 \leqslant i, j \leqslant n \tag{2-9}$$

Modularity matrix $\mathbf{M}$ is the utility matrix for modularity optimization. Note that the indicator matrix $\mathbf{H}$ can also be viewed as a latent feature matrix, and thus clustering algorithms can be applied to its rows to obtain the community partition.

2.6 Nonlinear Mapping Model

While previous methods introduced in this chapter all find linear embedding of the utility matrix as the feature matrix, in deep reconstruction model nonlinear embedding of the utility matrix is generated. In [70], modularity matrix $\mathbf{M}$, as input of an auto-encoder, is mapped to a low-dimensional representation $\mathbf{H} \in \mathbb{R}^{r \times n}$. The ith column of $\mathbf{H}$, which

represents node i in the latent space, is mapped as

$$\mathbf{h}_i = g(\mathbf{W}_H \mathbf{m}_i + \mathbf{d}_H) \tag{2-10}$$

where $\mathbf{W}_H \in \mathbb{R}^{r \times n}$, $\mathbf{d}_H \in \mathbb{R}^{r \times 1}$ are the parameters to be learned in the encoder, $\mathbf{m}_i$ is the ith column of the modularity matrix $\mathbf{M}$, and $g(\cdot)$ is an element-wise nonlinear mapping, such as sigmoid function

$$g_{sigmoid}(x) = \frac{1}{1+e^{-x}} \tag{2-11}$$

or tanh function

$$g_{tanh}(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2-12}$$

The decoder maps the latent representation $\mathbf{H}$ back into the original data space,

$$\mathbf{r}_i = l(\mathbf{W}_M \mathbf{h}_i + \mathbf{d}_M) \tag{2-13}$$

where $\mathbf{r}_i$ is the ith column of the reconstructed data $\mathbf{R} \in \mathbb{R}^{n \times n}$, $\mathbf{W}_M \in \mathbb{R}^{n \times r}$, $\mathbf{d}_M \in \mathbb{R}^{n \times 1}$ are the parameters to be learned in the decoder and $l(\cdot)$ is another element-wise nonlinear mapping similar to $g(\cdot)$. The objective of the auto-encoder is to minimize the reconstruction error between original $\mathbf{M}$ and reconstructed $\mathbf{R}$,

$$\min L = \sum_{i=1}^{n} L_\theta(\mathbf{m}_i, \mathbf{r}_i) \tag{2-14}$$

where $L_\theta(\mathbf{m}_i, \mathbf{r}_i)$ is a distance function (e.g. Euclidean distance or sigmoid cross-entropy distance). When the parameters of the auto-encoder is learned, the columns of the latent nonlinear embedding $\mathbf{H}$ can be clustered to find the community partition. In practice, a series of auto-encoders can be stacked for the reconstruction task to take advantage of a deep structure.

Chapter 3 Transfer Learning for Multiplex Community Detection

3.1 Problem

3.1.1 Observation

While in Chapter 1 we introduced various methods and algorithms for community detection in multiplex networks, all of them are based on a plausible assumption that the underlying community partitions for all layers are identical, and their objective is thus to find that partition. However, considering the fact that different layers of a multiplex network encode different types of relationships, we argue that for each layer, there is a distinct community structure behind it. And while the community partitions for different layers can be related to each other as nodes may share some common behaviors across layers (e.g. high-degree), they should be intrinsically distinct as they are actually different types of communities (e.g. friend circles vs. work groups). This argument is based on our observation of real-world multiplex networks.

Consider, for example, the Aarhus Computer Science Department Network (AUCS)[71], which is an unweighted and undirected five-dimensional multiplex network representing interactions between employees of the computer science department at Aarhus University in Aarhus, Denmark. It consists of 61 employees (administrative staff, faculty, research associates, Ph.D. students, and postdocs) belonging to eight work groups. The five interaction dimensions are lunch (1), Facebook (2), co-authors (3), leisure (4) and work (5). A community partition found for all the layers by MDLPA algorithm [43] is shown in Figure 3-1. Although the community structures identified correspond well to the ground-truth work groups, it is easy for us to find that edges in the first dimension dominate intra-community connection for all work groups. Meanwhile, the second dimension accounts for the majority of inter-community connection. This finding suggests that while the underlying community structure for the first layer may be highly correlated to real-world work groups, it is not the case for second layer. Actually, the second layer (Facebook interaction) may encode friendship relations that are independent of work-based relations.

This observation is consistent with our argument that different layers have different community structures. On the other hand, we also note that nodes do share some common features across different dimensions. For example, consider two specific nodes, U141 and U92 in the work group colored by yellow. U141 is only connected to three other nodes U48, U92 and U68, and only in the first dimension, which may indicate that he or she is a new-comer or a taciturn individual who avoids socializing with others. U92, on the contrary, is connected to 7 nodes in first dimension, which may indicate that he or she is more out-going. And this characteristic is also reflected in other dimensions as U92 is also connected to other nodes in the second, fourth and fifth dimension.

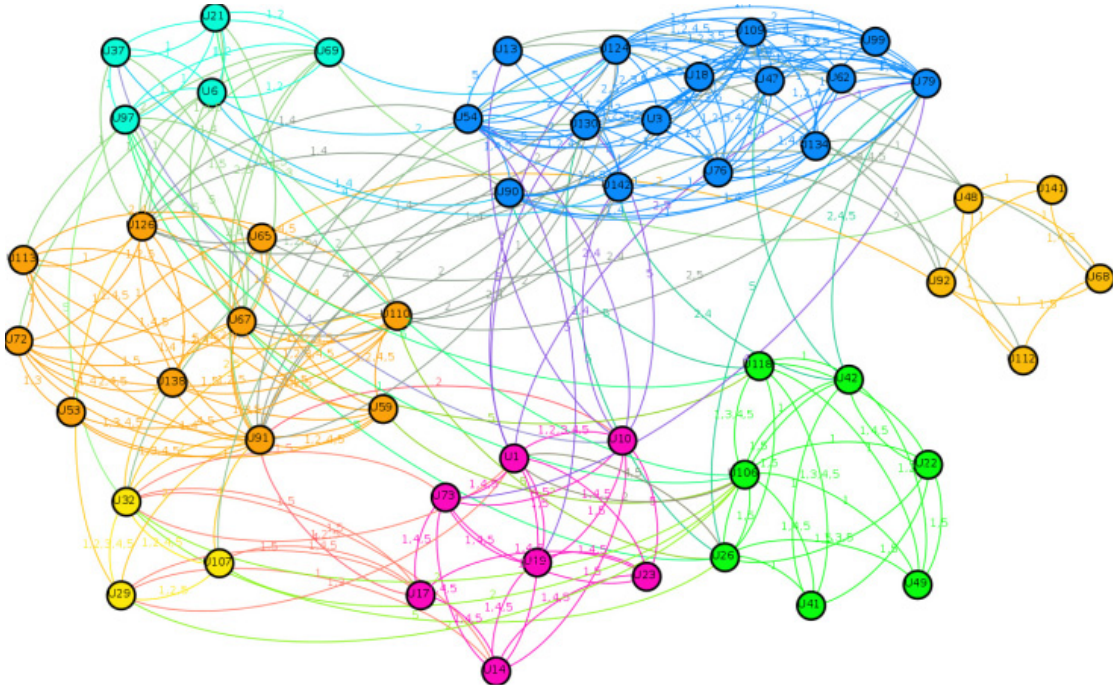


Figure 3-1 The community structures (indicated by different colors) of the Aarhus computer science department network as identified by MDLPA [43]. Numbers on edges designate dimension IDs.

Based on our observation, we propose a different and more realistic objective for multiplex community detection, i.e. finding distinct community partitions for each layer of a multiplex network, while exploiting the common features of nodes in different layers (i.e. transferred knowledge). We will formally define this objective in the following subsection.

3.1.2 Formulation

We borrow the notations of multiplex community detection from Chapter 1 except for the community part. Instead of finding a community partition $\mathtt{C} = \{C_1, ..., C_{k_{com}}\}$ for all the layers, we aim to find a distinct community partition for each layer, i.e. $\mathtt{C}_K = \{C_{K1}, ..., C_{Kk_{com_K}}\}$ for Kth layer, where k_{com_K} is the number of communities to be found in Kth layer. Unlike monoplex methods which find the community partition $\mathtt{C}_K$ with topological features only in Kth layer (i.e. layer-specific features), we hope to make use of those from other layers as well (i.e. common features). Formally, we define our problem as follows.

Problem 3.1 *Given a N-layer multiplex network $G(V, E_1, ..., E_N)$ with adjacency matrices $\mathbf{A}_1, ..., \mathbf{A}_N$, find distinct community partitions in each layer, $\mathtt{C}_1, ..., \mathtt{C}_N$, so that both common features and layer-specific features of nodes are exploited.*

Note that this definition of our problem is in the multi-task learning setting, as the community partition results of all layers are found simultaneously. In reality, only some layer may be of interest, which leads to the following definition of our problem in the transfer learning setting,

Problem 3.2 *Given a N-layer multiplex network $G(V, E_1, ..., E_N)$ with adjacency matrices $\mathbf{A}_1, ..., \mathbf{A}_N$, find distinct community partition in some layer K', $\mathtt{C}_{K'}$, so that both transferred knowledge from other layers and layer-specific features of K'th layer are exploited.*

In the following section when we develop our framework, we shall see that it can accommodate both settings of the problem, with little modification.

3.2 Algorithm

3.2.1 Framework

Since we are the first to define the objective of multiplex community detection as to finding a community partition for some layer with transferred knowledge from other layers, we can't use multiplex community detection techniques introduced in Chapter 1.

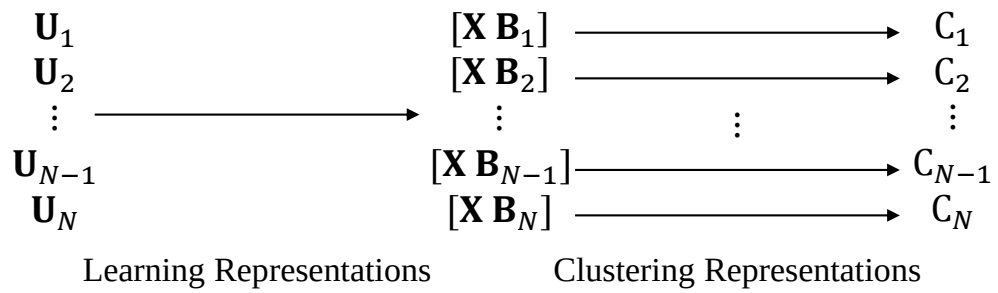


Figure 3-2 Framework of our algorithm. $\mathbf{U}_1, ..., \mathbf{U}_N$ are the utility matrix of each layer, $\mathbf{X}$ is the common feature matrix shared by all layers, $\mathbf{B}_1, ..., \mathbf{B}_N$ are the layer-specific feature matrices, and $\mathtt{C}_1, ..., \mathtt{C}_N$ are the found community partitions for each layer.

Here, we consider extending the representation-based community detection methods in Chapter 2 by incorporating shared features. Our framework is illustrated in Figure 3-2.

Unlike previous representation-based community detection methods which work on monoplex networks and only learn a latent representation for each layer separately, our framework takes the utility matrices $\mathbf{U}_1, ..., \mathbf{U}_N$ of all layers as input and collectively finds a common feature matrix $\mathbf{X}$ ($\mathbf{X} \in \mathbb{R}^{n \times c}$, where c is the number of dimensions of common features), along with layer-specific feature matrices $\mathbf{B}_1, ..., \mathbf{B}_N$ ($\mathbf{B}_K \in \mathbb{R}^{n \times c_K}$, where c_K is the number of dimensions of layer-specific features of Kth layer). $\mathbf{X}$ and $\mathbf{B}_K$ then collectively form the combined feature matrix $[\mathbf{X}\ \mathbf{B}_K]$ for Kth layer. In this way, the combined feature matrix for each layer incorporates shared features, or transferred knowledge from other layers, which can help improve the community detection results. Once the latent representations $[\mathbf{X}\ \mathbf{B}_1], ..., [\mathbf{X}\ \mathbf{B}_N]$ are learned, we can apply clustering algorithms on the latent representation of each layer separately. And the community partition for each layer is the result of the clustering algorithm. We summarize our framework of transfer learning for multiplex community detection in Algorithm 1.

Note that our framework in Figure 3-2 and Algorithm 1 solve the problem in the multi-task learning setting. To accommodate it to the transfer learning setting, we apply the representation clustering algorithm to the target layer K' only, instead of all the layers, while keeping our representation learning process unchanged.

To implement our framework, the representation learning method and the representation clustering method must be specified. In the following subsections, we will intro-

duce a possible implementation of these methods. However, it is noteworthy that our framework can easily accommodate different sets of learning and clustering methods.

Algorithm 1 Our Framework of Transfer Learning for Multiplex Community Detection

Input: Utility matrices of a N-layers multiplex network $\mathbf{U}_1, ..., \mathbf{U}_N$

Output: Community partitions for each layer $\mathtt{C}_1, ..., \mathtt{C}_N$

1: $\mathbf{X}, \mathbf{B}_1, ..., \mathbf{B}_N \leftarrow$ *learn_representation*$(\mathbf{U}_1, ..., \mathbf{U}_N)$

2: **for** $K \leftarrow 1$ to N **do**

3: $\quad \mathtt{C}_K \leftarrow$ *cluster_representation*$(\mathbf{X}, \mathbf{B}_K)$

4: **end for**

5: **return** $\mathtt{C}_1, ..., \mathtt{C}_N$

3.2.2 Learning Representations

While in Chapter 2 several different types of representation learning methods are available for monoplex networks, none of them are directly applicable to our problem since they only deal with monoplex networks. Here, we consider extending the stochastic model in Section 2.2, which is formulated as a symmetric non-negative matrix factorization problem.

The utility matrices $\mathbf{U}_1, ..., \mathbf{U}_N$ are the adjacency matrices $\mathbf{A}_1, ..., \mathbf{A}_N$ as in the stochastic model. For each layer K, our objective is to find the combined feature matrix $[\mathbf{X}\ \mathbf{B}_K]$ that best reconstructs the original adjacency matrix $\mathbf{A}_K$. Adopting the square loss function to quantify the reconstruction error, we have

$$L_K = ||\mathbf{A}_K - [\mathbf{X}\ \mathbf{B}_K][\mathbf{X}\ \mathbf{B}_K]^T||_F^2 = ||\mathbf{A}_K - \mathbf{X}\mathbf{X}^T - \mathbf{B}_K\mathbf{B}_K^T||_F^2 \tag{3-1}$$

for Kth layer. Note that if we minimize L_K separately for each layer,

$$\min_{\mathbf{X}\geqslant 0, \mathbf{B}_K \geqslant 0} L_K = ||\mathbf{A}_K - \mathbf{X}\mathbf{X}^T - \mathbf{B}_K\mathbf{B}_K^T||_F^2 \tag{3-2}$$

then this formulation reduces to monoplex stochastic model and $[\mathbf{X}\ \mathbf{B}_K]$ is the latent feature matrix $\mathbf{H}$. Since we want to transfer knowledge from other layers through the common feature matrix $\mathbf{X}$ to help improve the latent representations, we simultaneously minimize the reconstruction error of all layers,

$$\min_{\mathbf{X}\geqslant 0, \mathbf{B}_1, ..., \mathbf{B}_N \geqslant 0} L = \sum_{K=1}^{N} w_K ||\mathbf{A}_K - \mathbf{X}\mathbf{X}^T - \mathbf{B}_K\mathbf{B}_K^T||_F^2 \tag{3-3}$$

where w_K is the weight for the reconstruction error of Kth layer. If there is no prior knowledge about which layer should contribute more to the common feature matrix (e.g. some layer is less noisy), we can simply set all weights to be equal. Then Equation 3-3 reduces to

$$\min_{\mathbf{X}\geqslant 0,\mathbf{B}_1,...,\mathbf{B}_N\geqslant 0} L = \sum_{K=1}^{N} ||\mathbf{A}_K - \mathbf{X}\mathbf{X}^T - \mathbf{B}_K\mathbf{B}_K^T||_F^2 \tag{3-4}$$

For this optimization problem, we have the following theorem.

Theorem 3.1 *The sum of residuals in Equation 3-4, L, can't be minimized to zero.*

Proof We only need to prove that $||\mathbf{A}_K - \mathbf{X}\mathbf{X}^T - \mathbf{B}_K\mathbf{B}_K^T||_F^2 > 0$, which is equivalent to $\mathbf{A}_K \neq \mathbf{X}\mathbf{X}^T + \mathbf{B}_K$. We prove this by showing that $\mathbf{X}\mathbf{X}^T + \mathbf{B}_K\mathbf{B}_K^T$ is positive semi-definite and that $\mathbf{A}_K$ is not.

First, for any non-zero vector $\mathbf{s} \in \mathbb{R}^{n\times 1}$, we have

$$\begin{aligned}\mathbf{s}^T(\mathbf{X}\mathbf{X}^T + \mathbf{B}_K\mathbf{B}_K^T)\mathbf{s} &= \mathbf{s}^T\mathbf{X}\mathbf{X}^T\mathbf{s} + \mathbf{s}^T\mathbf{B}_K\mathbf{B}_K^T\mathbf{s} \\ &= \mathbf{g}^T\mathbf{g} + \mathbf{h}^T\mathbf{h} \\ &= ||\mathbf{g}||^2 + ||\mathbf{h}||^2 \geqslant 0\end{aligned} \tag{3-5}$$

where $\mathbf{g} = \mathbf{X}^T\mathbf{s}, \mathbf{h} = \mathbf{B}_K^T\mathbf{s}$. This shows that $\mathbf{X}\mathbf{X}^T + \mathbf{B}_K\mathbf{B}_K^T$ is positive semi-definite.

Then, for undirected networks with no self-loops, we have $\mathbf{A}_K$ to be symmetric and

$$\sum_{i=1}^{n} \lambda_i = \mathrm{Tr}(\mathbf{A_K}) = \sum_{i=1}^{n} \mathbf{A}_{Kii} = 0 \tag{3-6}$$

where $\lambda_1, ..., \lambda_n$ are the n eigenvalues of $\mathbf{A}_K$. Therefore, either there is at least some $\lambda_i < 0$ or $\lambda_1 = ... = \lambda_n = 0$. However, since $\mathbf{A}_K$ is symmetric, $\lambda_1 = ... = \lambda_n = 0$ directly leads to $\mathbf{A}_K = 0$. Thus, $\lambda_i < 0$ holds, which means $\mathbf{A}_K$ can't be positive semi-definite. ■

Theorem 3.1 states that the reconstruction error can't be minimized to zero, which means that it is impossible to perfectly reconstruct the adjacency matrices with the latent feature matrices.

Now that we have our objective function for learning representations, we need a procedure to optimize it. While Equation 3-4 resembles the objective function of symmetric non-negative matrix factorization, it can't be solved accordingly. Consider the simplest

case when both $\mathbf{X}$ and $\mathbf{B}_K$ ($K \neq K'$) are fixed and we want to find the optimal $\mathbf{B}'_K$. Then, Equation 3-4 reduces to

$$\min_{\mathbf{B}_{K'} \geqslant 0} L_{K'} = ||\mathbf{A}_{K'} - \mathbf{X}\mathbf{X}^T - \mathbf{B}_{K'}\mathbf{B}_{K'}^T||_F^2 \tag{3-7}$$

However, since $\mathbf{A}'_K - \mathbf{X}\mathbf{X}^T \geqslant 0$ doesn't always hold, we can't apply standard symmetric non-negative matrix factorization procedures (such as [72], [73] and [74]). In the following, we design a new set of alternating multiplicative update rules for optimizing our objective, based on semi-NMF techniques [75].

We first compute the partial derivatives of our objective function L with respect to $\mathbf{X}$ and $\mathbf{B}_K$,

$$\frac{\partial L}{\partial \mathbf{X}} = 4N\mathbf{X}\mathbf{X}^T\mathbf{X} + 4\sum_{K=1}^{N}(\mathbf{B}_K\mathbf{B}_K^T - \mathbf{A}_K)\mathbf{X} \tag{3-8}$$

$$\frac{\partial L}{\partial \mathbf{B}_K} = 4\mathbf{B}_K\mathbf{B}_K^T\mathbf{B}_K + 4(\mathbf{X}\mathbf{X}^T - \mathbf{A}_K)\mathbf{B}_K \tag{3-9}$$

Since we require both $\mathbf{X} \geqslant 0$ and $\mathbf{B}_K \geqslant 0$, we introduce the respective multipliers:

$$\mathbf{\Lambda} \in \mathbb{R}^{n \times c}, \mathbf{\Gamma}_K \in \mathbb{R}^{n \times c_K}, 1 \leqslant K \leqslant N \tag{3-10}$$

Then, we have the following Lagrangian:

$$\mathcal{L}(\mathbf{X}, \mathbf{B}_K) = L - \mathrm{Tr}(\mathbf{\Lambda}\mathbf{X}^T) - \sum_{K=1}^{N}\mathrm{Tr}(\mathbf{\Gamma}_K\mathbf{B}_K^T) \tag{3-11}$$

From the stationarity of Karush-Kuhn-Tucker conditions we have

$$\frac{\partial \mathcal{L}}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{X}} - \mathbf{\Lambda} = 0, \frac{\partial \mathcal{L}}{\partial \mathbf{B}_K} = \frac{\partial L}{\partial \mathbf{B}_K} - \mathbf{\Gamma}_K = 0 \tag{3-12}$$

which leads to

$$\mathbf{\Lambda} = 4N\mathbf{X}\mathbf{X}^T\mathbf{X} + 4\sum_{K=1}^{N}(\mathbf{B}_K\mathbf{B}_K^T - \mathbf{A}_K)\mathbf{X} \tag{3-13}$$

$$\mathbf{\Gamma}_K = 4\mathbf{B}_K\mathbf{B}_K^T\mathbf{B}_K + 4(\mathbf{X}\mathbf{X}^T - \mathbf{A}_K)\mathbf{B}_K \tag{3-14}$$

Then, from the complementary slackness of Karush-Kuhn-Tucker conditions

$$\mathbf{\Lambda}_{ij}\mathbf{X}_{ij} = 0, \mathbf{\Gamma}_{Kij}\mathbf{B}_{Kij} = 0 \tag{3-15}$$

Substituting Equation 3-13 and Equation 3-14 into Equation 3-15, we have

$$(-\sum_{K=1}^{N}(\mathbf{A}_K-\mathbf{B}_K\mathbf{B}_K^T)\mathbf{X}+N\mathbf{X}\mathbf{X}^T\mathbf{X})_{ij}\mathbf{X}_{ij}=0 \tag{3-16}$$

$$(-(\mathbf{A}_K-\mathbf{X}\mathbf{X}^T)\mathbf{B}_K+\mathbf{B}_K\mathbf{B}_K^T\mathbf{B}_K)_{ij}\mathbf{B}_{Kij}=0 \tag{3-17}$$

Based on Equation 3-16 and Equation 3-17, we design the following update rules for $\mathbf{X}$ and $\mathbf{B}_K$:

$$\mathbf{X}_{ij}\leftarrow\mathbf{X}_{ij}\sqrt{\frac{((\sum_{K=1}^{N}(\mathbf{A}_K-\mathbf{B}_K\mathbf{B}_K^T))^+\mathbf{X})_{ij}}{((\sum_{K=1}^{N}(\mathbf{A}_K-\mathbf{B}_K\mathbf{B}_K^T))^-\mathbf{X})_{ij}+(N\mathbf{X}\mathbf{X}^T\mathbf{X})_{ij}}} \tag{3-18}$$

$$\mathbf{B}_{Kij}\leftarrow\mathbf{B}_{Kij}\sqrt{\frac{((\mathbf{A}_K-\mathbf{X}\mathbf{X}^T)^+\mathbf{B}_K)_{ij}}{((\mathbf{A}_K-\mathbf{X}\mathbf{X}^T)^-\mathbf{B}_K)_{ij}+(\mathbf{B}_K\mathbf{B}_K^T\mathbf{B}_K)_{ij}}} \tag{3-19}$$

where we denote the positive and negative parts of a matrix $\mathbf{E}$ as

$$\mathbf{E}^+=(|\mathbf{E}|+\mathbf{E})/2 \tag{3-20}$$

$$\mathbf{E}^-=(|\mathbf{E}|-\mathbf{E})/2 \tag{3-21}$$

To minimize our objective, we first initialize $\mathbf{X}$ and $\mathbf{B}_1,...,\mathbf{B}_N$ to matrices with entries that are uniformly distributed in $[0,1]$. Then, we alternatingly apply Equation 3-18 to $\mathbf{X}$ and Equation 3-19 to $\mathbf{B}_K, 1\leqslant K\leqslant N$ until convergence.

For this set of update rules, we have the following theorem.

Theorem 3.2 *The limiting solutions of the update rules in Equation 3-18 and Equation 3-19 satisfy Karush-Kuhn-Tucker conditions in Equation 3-16 and Equation 3-17.*

Proof We only prove that Equation 3-18 satisfies Equation 3-16 at convergence, and that Equation 3-19 satisfies Equation 3-17 can be proved in a similar manner.

At convergence, we have $\mathbf{X}^\infty=\mathbf{X}^{t+1}=\mathbf{X}^t=\mathbf{X}$. Then, from Equation 3-18

$$\mathbf{X}_{ij}=\mathbf{X}_{ij}\sqrt{\frac{((\sum_{K=1}^{N}(\mathbf{A}_K-\mathbf{B}_K\mathbf{B}_K^T))^+\mathbf{X})_{ij}}{((\sum_{K=1}^{N}(\mathbf{A}_K-\mathbf{B}_K\mathbf{B}_K^T))^-\mathbf{X})_{ij}+(N\mathbf{X}\mathbf{X}^T\mathbf{X})_{ij}}} \tag{3-22}$$

Square both sides, multiply it by the denominator, and transpose it, we get

$$(((\sum_{K=1}^{N}(\mathbf{A}_K-\mathbf{B}_K\mathbf{B}_K^T))^- -(\sum_{K=1}^{N}(\mathbf{A}_K-\mathbf{B}_K\mathbf{B}_K^T))^+)\mathbf{X}+N\mathbf{X}\mathbf{X}^T\mathbf{X})_{ij}\mathbf{X}_{ij}^2=0 \tag{3-23}$$

Note that

$$(\sum_{K=1}^{N}(\mathbf{A}_K-\mathbf{B}_K\mathbf{B}_K^T))^+-(\sum_{K=1}^{N}(\mathbf{A}_K-\mathbf{B}_K\mathbf{B}_K^T))^-=\sum_{K=1}^{N}(\mathbf{A}_K-\mathbf{B}_K\mathbf{B}_K^T) \quad (3\text{-}24)$$

Therefore, Equation 3-23 reduces to

$$(-\sum_{K=1}^{N}(\mathbf{A}_K-\mathbf{B}_K\mathbf{B}_K^T)\mathbf{X}+N\mathbf{X}\mathbf{X}^T\mathbf{X})_{ij}\mathbf{X}_{ij}^2=0 \quad (3\text{-}25)$$

Equation 3-25 is equivalent to Equation 3-16, the respective Karush-Kuhn-Tucker condition, since both require either the first term $(-\sum_{K=1}^{N}(\mathbf{A}_K-\mathbf{B}_K\mathbf{B}_K^T)\mathbf{X}+N\mathbf{X}\mathbf{X}^T\mathbf{X})_{ij}$ to be zero or the second terms $\mathbf{X}_{ij}$, $\mathbf{X}_{ij}^2$ to be zero. ■

Theorem 3.2 indicates that if our algorithm converges, then its solution converges to a Karush-Kuhn-Tucker stationary point. In the next section, we will empirically show that under this set of alternating multiplicative rules, the reconstruction error converges at a fast rate. Therefore, in our implementation, we only need to set a maximum number of iterations T, after which our algorithm stops. In general, our algorithm produces relatively stable outputs, independent of network size and other properties, when T is greater than 20.

Algorithm 2 *learn_representation*$(\mathbf{A}_1,...,\mathbf{A}_N)$

Input: Adjacency matrices of a N-layers multiplex network $\mathbf{A}_1,...,\mathbf{A}_N$

Output: A common feature matrix $\mathbf{X}$ and layer-specific feature matrices $\mathbf{B}_1,...,\mathbf{B}_N$

1: $\mathbf{X}_{ij},\mathbf{B}_{1ij},...,\mathbf{B}_{Nij} \leftarrow$ *rand*$([0,1])$

2: **for** $t=1\rightarrow T$ **do**

3: $\quad\mathbf{X}_{ij}\leftarrow\mathbf{X}_{ij}\sqrt{\frac{((\sum_{K=1}^{N}(\mathbf{A}_K-\mathbf{B}_K\mathbf{B}_K^T))^+\mathbf{X})_{ij}}{((\sum_{K=1}^{N}(\mathbf{A}_K-\mathbf{B}_K\mathbf{B}_K^T))^-\mathbf{X})_{ij}+(N\mathbf{X}\mathbf{X}^T\mathbf{X})_{ij}}}$

4: $\quad$**for** $K=1\rightarrow N$ **do**

5: $\quad\quad\mathbf{B}_{Kij}\leftarrow\mathbf{B}_{Kij}\sqrt{\frac{((\mathbf{A}_K-\mathbf{X}\mathbf{X}^T)^+\mathbf{B}_K)_{ij}}{((\mathbf{A}_K-\mathbf{X}\mathbf{X}^T)^-\mathbf{B}_K)_{ij}+(\mathbf{B}_K\mathbf{B}_K^T\mathbf{B}_K)_{ij}}}$

6: $\quad$**end for**

7: **end for**

8: **return** $\mathbf{X}$, $\mathbf{B}_1,...,\mathbf{B}_N$

We summarize our representation learning algorithm in Algorithm 2. The complexity of Algorithm 2 can be analyzed in a quite straightforward manner. The update rule of $\mathbf{X}$ in Line 3 requires $O(Nn^2(c+\max(c_1,...,c_N)))$ operations by schoolbook matrix multiplication. For each $\mathbf{B}_K$, the update rule in Line 5 requires $O(n^2(c+c_K))$ operations.

Then, to update all the $\mathbf{B}_K, 1 \leqslant K \leqslant N$, $O(Nn^2(c+\max(c_1,...,c_N)))$ operations are required. Therefore, for a total of T iterations, Algorithm 2 has a computational cost of $O(NTn^2(c+\max(c_1,...,c_N)))$. For most real-world networks, $N, c, c_K \ll n$ and $T \sim 20$, and our representation learning algorithm runs in $O(n^2)$.

3.2.3 Clustering Representations

After the latent representations are learned, any standard clustering algorithm can handle the task of clustering representations, as they are in feature vector form. Therefore, in our implementation, we simply apply k-means algorithm for clustering.

The representation clustering algorithm is shown in Algorithm 3. Since it directly uses k-means, its complexity is $O(T'ndk_{com})$, where T' is the number of iterations until k-means convergence ($T' \sim 100$), $d = c + c'_K \ll n$ is the number of columns of the combined feature matrix $[\mathbf{X}\ \mathbf{B}_{K'}]$, $k_{com} \ll n$ is the number of communities to be found. Therefore, the representation clustering algorithm runs in $O(n)$.

Algorithm 3 *cluster_representation*$(\mathbf{X}, \mathbf{B}_K)$

Input: The common feature matrix $\mathbf{X}$ and the layer-specific feature matrix $\mathbf{B}_K$ of Kth layer.

Output: The community partition for Kth layer $\mathtt{C}_K$.

1: $\mathtt{C}_K \leftarrow$ *k-means*$([\mathbf{X}\ \mathbf{B}_K])$

2: **return** $\mathtt{C}_K$

3.3 Experiment

In this section, we will introduce our experiment setup for evaluating our algorithm and comparing it with monoplex methods.

3.3.1 Dataset

Unlike monoplex network, of which both synthetic benchmark datasets (e.g. Girvan-Newman [23] and Lancichinetti-Fortunato-Radicchi [76]) and large real-world dataset repositories (e.g. Stanford Network Analysis Project① and UC Irvine Network Data Repository②) are readily available to the community, only a few real-world multiplex networks

① `http://snap.stanford.edu/data/index.html`

② `https://networkdata.ics.uci.edu/`

are well recorded. We collect six small(~ 100 nodes) to medium ($\sim 10,000$ nodes) multiplex networks for our evaluation purposes as our algorithm ($O(n^2)$) can't deal with large networks ($\sim 1,000,000$ nodes) in a short time. Despite this, we also list some large multiplex networks in this subsection for other possible implementations of our framework, which may be better scalable ($\sim O(n)$).

- **AUCS** [71]. As introduced in Subsection 3.1.1, Aarhus Computer Science Department Network (AUCS) is an unweighted five-dimensional multiplex network. Each layer represents one type of interaction between the employees of the computer science department at the Aarhus University: lunch, Facebook, co-authors, leisure and work. It has 61 nodes and 620 edges. It is available at `http://multilayer.it.uu.se/datasets.html`.
- **Seventhgrade** [77]. Seventhgrade is an unweighted three-dimensional social network. It is derived from surveying a class of seventh grade students in a school in Victoria, Australia. The three layers record students' answer to the following questions: 1. *Who do you get on with in the class?* 2. *Who are your best friends in the class?* and 3. *Who would you prefer to work with?* It has 29 nodes and 740 edges. It is available at `http://deim.urv.cat/~manlio.dedomenico/data.php`.
- **Physician** [78]. Physician is an unweighted three-dimensional social network. It is concerned with the impact of network ties on the physicians' adoption of a new drug, tetracycline in Illinois, Peoria, Bloomington, Quincy and Galesburg, the United States. The three layers record physician' answers to the following questions: 1. *When you need information or advice about questions of therapy where do you usually turn?* 2. *Who are the three or four physicians with whom you most often find yourself discussing cases or therapy in the course of an ordinary week, last week for instance?* and 3. *Would you tell me the first names of your three friends whom you see most often socially?* It has 246 nodes and 1,551 edges. It is available at `http://deim.urv.cat/~manlio.dedomenico/data.php`.
- **Bos** [79, 80]. This unweighted four-dimensional biological network is derived from the Biological General Repository for Interaction Datasets (BioGRID[①]). It encodes the genetic and protein interactions of Bos Linnaeus. The four dimensions represent physical association, association, direct interaction and coloniza-

① `https://thebiogrid.org/`

tion, respectively. It has 321 nodes and 325 edges. In our experiment, we only keep the first and third layers as the other two layers are too sparse (containing 9 and 3 edges, respectively). This dataset is available at `http://deim.urv.cat/~manlio.dedomenico/data.php`.

- **Celegans** [81, 82]. Celegans is an unweighted three-dimensional biological network which represents neuronal network of the nematode *Caenorhabditis elegans*. The three layers corresponds to three different types of synaptic junction: electric junction, chemical monadic junction and chemical polyadic junction. It has 279 nodes and 5,863 edges. It is available at `http://deim.urv.cat/~manlio.dedomenico/data.php`
- **ff-tw-yt** [83]. ff-tw-yt is an unweighted three-dimensional online social network. It has been obtained starting from Friendfeed, a social media aggregator. Users of Friendfeed comment on other messages much like in Facebook, which forms the first layer of the network. The second layer and third layer represent Twitter and YouTube interaction of the same set of users with associated accounts respectively. It has 6,407 nodes and 74,862 edges. It is available at `http://multilayer.it.uu.se/datasets.html`.

We also list three large networks for evaluating other possible implementations of our framework.

- **Biogrid** [84]. This two-dimensional biological network is based on protein interactions in the BioGRID dataset. The two layers represent physical and genetic interactions between proteins, respectively. It has 54,549 nodes and 503,049 edges. It is available at `http://www.maths.qmul.ac.uk/~vnicosia/sw.html`.
- **ff-tw** [83]. Similar to ff-tw-yt, this network only records interaction of users who have associated accounts in Friendfeed and Twitter, which allows a much bigger size than ff-tw-yt. It has 155,804 nodes and 13,657,550 edges. It is available at `http://multilayer.it.uu.se/datasets.html`.
- **Aminer** [85]. Aminer itself is a collection of research papers, author information and co-authorship of the academic community. A two-dimensional academic network can be derived from the raw data. The nodes of the network are the scholars and the two types of connections represent co-authorship, where two scholars are connected if they co-author a paper, and citationship, where two scholars are

connected if one's paper cites the other's. This derived network has 1,712,433 nodes and 42,272,409 edges. The raw data is available at `https://aminer.org/aminernetwork`.

For all the datasets used in our experiments, we do the following pre-processing procedures:

1.Make its adjacency matrices symmetric by adding a returning edge for every existing edge. Any directed layer becomes undirected after this procedure.

2.Set the main diagonal entries of the adjacency matrices to zero. This removes all self-loops in the network.

3.3.2 Metric

Since our objective is to improve community detection performance in some layer with transferred knowledge from other layers, we only need evaluation metrics for the monoplex network, instead of multiplex network. For monoplex network, two types of quality functions are most widely accepted.

•**Normalized Mutual Information (NMI)** [86]. Normalized mutual information is a measure of similarity of partitions borrowed from information theory, which has proved to be reliable [87]. It evaluates the community detection results by calculating the correspondence from the obtained partition $\mathtt{C}$ to the reference partition $\mathtt{C}'$, which is often derived from ground truth. The NMI is defined as follows,

$$\mathrm{NMI}(\mathtt{C},\mathtt{C}') = \frac{2I(\mathtt{C},\mathtt{C}')}{H(\mathtt{C})+H(\mathtt{C}')} \tag{3-26}$$

where $I(\mathtt{C},\mathtt{C}') = H(\mathtt{C}) - H(\mathtt{C}|\mathtt{C}')$ is the mutual information between $\mathtt{C}$ and $\mathtt{C}'$, $H(\mathtt{C})$ and $H(\mathtt{C}')$ are the Shannon entropy of $\mathtt{C}$ and $\mathtt{C}'$ respectively, and $H(\mathtt{C}|\mathtt{C}')$ is the conditional entropy of $\mathtt{C}$ given $\mathtt{C}'$. NMI ranges in $[0,1]$, with $\mathrm{NMI}=1$ indicating that the reference partition and the obtained partition are identical. However, as an internal criterion, NMI requires that we know exactly how the underlying community partition is, which is available for many monoplex networks but very few multiplex networks. That is because even when some ground-truth community partitions are available in some multiplex networks (e.g. work group information is available for AUCS), we can't decide which layer these community structures correspond to.

Therefore, we must resort to external criteria for evaluating our results, the most widely accepted one of which is introduced below.

•**Modularity** [35]. As an external criterion to assess the community detection effectiveness, modularity can be computed from the obtained community partition along with network topological information, in the absence of ground truth. It measures how far the network interactions deviate from random connections, and is defined as follows,

$$Q = \frac{1}{2m} \sum_{C \in \mathtt{C}} \sum_{i \in C, j \in C} (A_{ij} - \frac{k_i k_j}{2m}) \tag{3-27}$$

where $\mathbf{A}_{ij}$ is the actual connection between a pair of nodes i, j in the same community C, $\frac{k_i k_j}{2m}$ is the expected number of edges between i and j if edges are placed randomly, the coefficient $\frac{1}{2m}$ is introduced to normalize the modularity value into $[-1, 1]$. For modularity, large value indicates more deviation from the random graph, and thus better community partition.

In our experiments, we use modularity to evaluate our community detection results.

3.3.3 Baseline

To illustrate the effect of the transferred knowledge that our algorithm exploits, we compare our results with other monoplex non-negative matrix factorization-based methods. Specifically, we consider two implementations of symmetric non-negative matrix factorization approaches.

•**Coordinate Descent (CD)** [88]. This algorithm directly solves the symmetric non-negative matrix factorization, a fourth-order nonconvex problem, by solving a series of fourth-order univariate subproblems exactly with efficient coordinate descent schemes. The algorithm is proved to converge and can be performed in $O(r \max(S, nr)T)$, where n is the number of columns of the original matrix, r is the number of columns of factorized matrix, S is the number of nonzero entries in the original matrix, and T is the number of iterations. Their implementation is available at `https://sites.google.com/site/nicolasgillis/code`.

•**Newton** [74]. This algorithm takes a Newton-like strategy, with several improvements compared to the original projected Newton method. The improved algorithm

is guaranteed to converge, and can be performed in $O(n^3rT)$, where n is the number of columns of the original matrix, r is the number of columns of factorized matrix, and T is the number of iterations. Their implementation is available at `https://github.com/dakuang/symnmf`.

Adjacency matrix of the target layer is factorized by these monoplex symmetric non-negative matrix factorization methods to obtain the latent feature matrix. Then, the representations learned are clustered by k-means, the same clustering algorithm our implementation uses, to obtain the community partition for the target layer.

3.3.4 Parameters

Our algorithm requires several parameters to work properly. We discuss our choices and present our explanations as follows.

- **Target layer** K'. Considering the effect of 'negative transfer', where the transferred knowledge contributed to reduced performance [56], not all layers can achieve improved performance through our algorithm. Basically, sparse layers witness more improved performance as some edges may not reveal themselves in the observed network data. Therefore, we generally choose the most sparse layer of the multiplex network as the target layer to receive the transferred knowledge. For AUCS, we choose the third layer (Facebook), for Seventhgrade, third layer (work with), for Physician, first layer (ask for advice), for Bos, second layer (direct interaction), for Celegans, second layer (chemical monadic junction), and for ff-tw-yt, first layer (YouTube).
- **Number of communities to be detected in the target layer** k_{com}. Since there is hardly any ground truth about the test datasets, we can only choose this parameter empirically. We run our algorithm with several different number of communities and keep those with largest modularity. For the test datasets, we choose 10 as the number of communities for target layer of AUCS, 3 for Seventhgrade, 4 for Physician, 10 for Bos, 10 for Celegans and 1000 for ff-tw-yt.
- **Latent feature dimensions** c, c_K. The number of dimensions of latent feature matrices are much less than that of the original utility matrix, which is usually quite sparse. Generally, we select smaller dimensions of layer-specific feature matrices for sparser layers since there is less information to encode, and the number of

dimensions for common feature matrix is usually similar to the minimum of those for layer-specific feature matrices. Common feature matrix dimensions and layer-specific feature matrix dimensions are 2 and $[1,1,1,1,1]$ for AUCS, 2 and $[2,2,2]$ for Seventhgrade, 2 and $[2,1,1]$ for Physician, 2 and $[2,1]$ for Bos, 1 and $[1,2,5]$ for Celegans and 1 and $[1,9,13]$ for ff-tw-yt.

- **Numbers of iterations** T, T'. Since both our learning algorithm and clustering algorithm converge fast and produce stable output regardless of input network data, we choose fixed numbers of iterations for both algorithms. For the representation learning algorithm, we choose $T = 20$, and for the representation clustering algorithm, we choose $T' = 100$.

For the baseline algorithms, we use $r = c + c_K$ for the number of columns in latent matrix of Kth layer, which guarantees same learning power as our algorithm. We use default options provided by their authors for choices of other parameters.

3.3.5 Environment

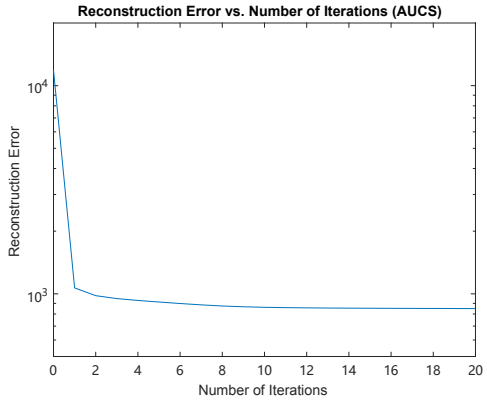
Our algorithm is implemented in MATLAB, as well as the two baseline methods. All experiments are conducted in MATLAB 9.0.0.341360 (R2016a), running on a laptop with Intel i7-4710MQ CPU and 8GB RAM.

3.4 Results

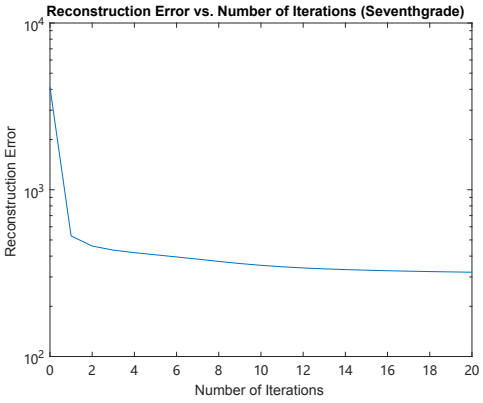
3.4.1 Convergence Analysis

We run our algorithm on the six test datasets. The decrease of reconstruction error (i.e. value of the objective function) as the number of iterations increase is shown in Figure 3-3. Note that the reconstruction error in all figures is drawn in logarithmic scale for better illustration.

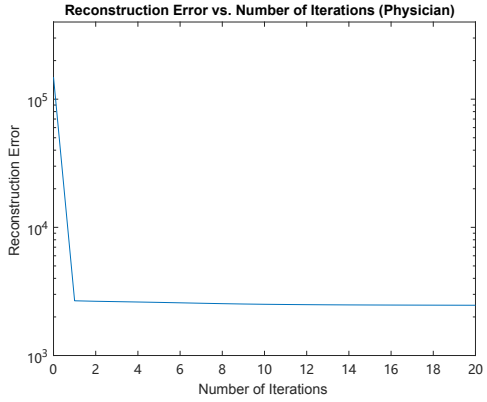
As is quite obvious in these figures, the reconstruction error drops sharply after the first iteration, because the latent feature matrices are randomly initialized. After the second iteration, however, the error decreases slowly but also steadily as the number of iterations increase, and reaches to a relative stable value before 20 iterations. As previous proved in Theorem 3.2, the solution converges to a Karush-Kuhn-Tucker stationary point



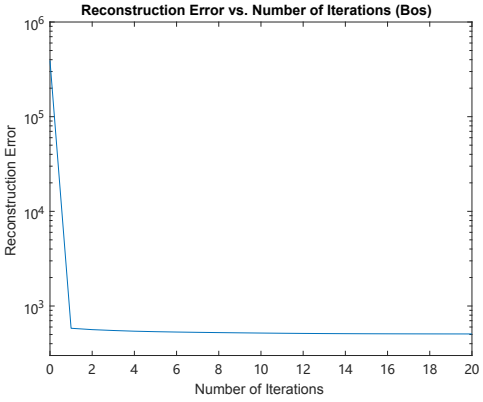
(a)



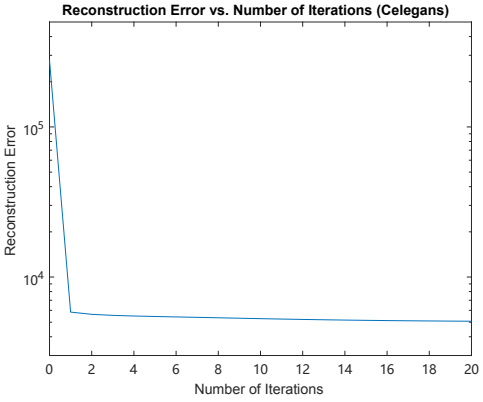
(b)



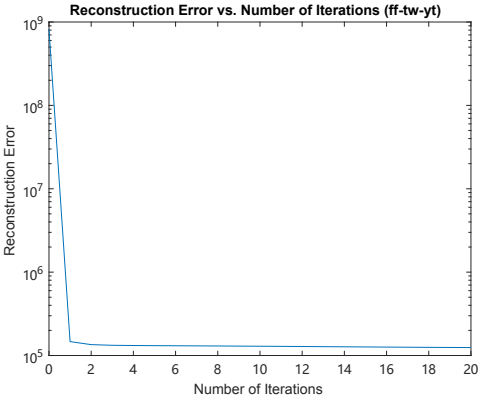
(c)



(d)



(e)



(f)

Figure 3-3 Relationship between reconstruction error and number of iterations for the test datasets.(a)AUCS;(b)Seventhgrade;(c)Physician;(d)Bos;(e)Celegans;(f)ff-tw-yt

if it converges. Combining it with our empirical results, we can say that the solution of our learning algorithm practically converges to a Karush-Kuhn-Tucker stationary point. Also note that the reconstruction error can't decrease to zero, which has also been proved in Theorem 3.1.

3.4.2 Comparative Analysis

We now compare the community detection performance of our algorithm (Multiplex) with other representation-based monoplex methods (Monoplex-CD and Monoplex-Newton). For ff-tw-yt dataset, we run each algorithm 100 times and compute the mean and the standard error of modularity of the result partition for comparison. For other datasets, we run each algorithm $10,000$ times. The results are summarized in Table 3-1 and illustrated in Figure 3-4.

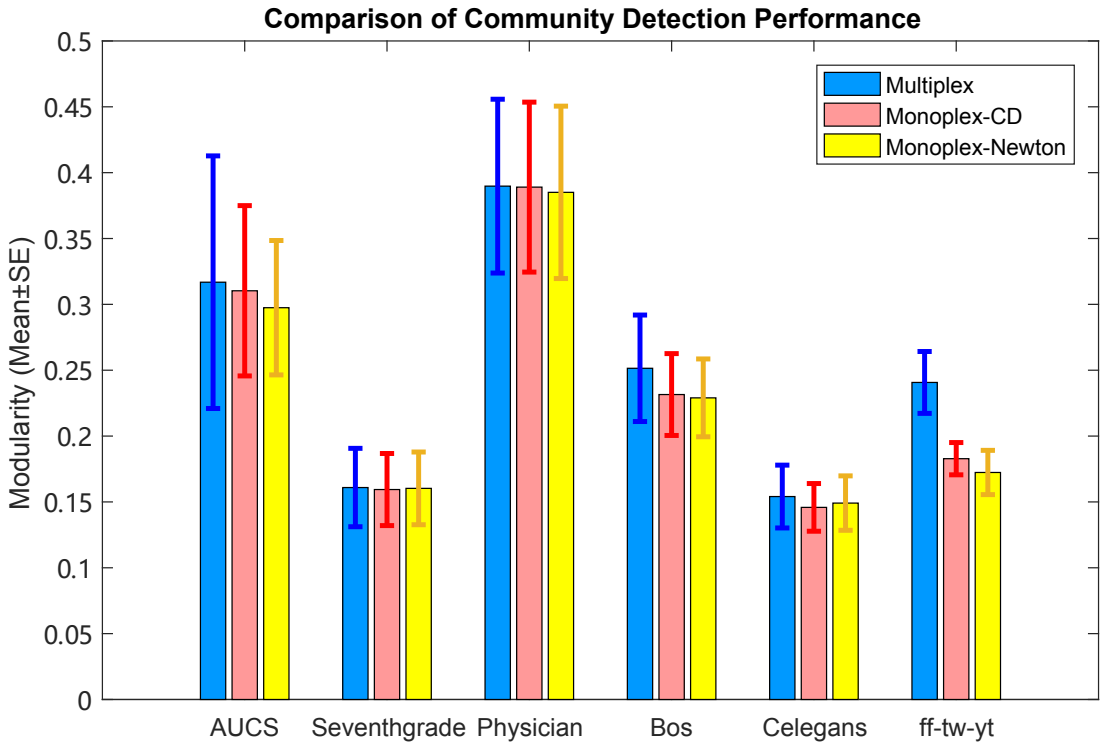


Figure 3-4 Comparison of community detection performance in terms of modualrity. Error bars depict standard errors.

For all the test datasets, we can see that our multiplex algorithm outperforms the monoplex algorithms. This is expected as our algorithm make use of transferred knowl-

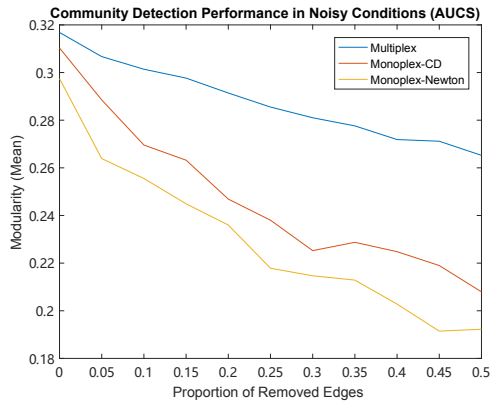
Table 3-1 Community detection performance of different algorithms

	AUCS	Seventhgrade	Physician	Bos	Celegans	ff-tw-yt
Multiplex	**0.317±0.096**	**0.161±0.030**	**0.390±0.066**	**0.251±0.040**	**0.154±0.024**	**0.241±0.023**
Monoplex-CD	0.310±0.065	0.159±0.027	0.389±0.065	0.232±0.031	0.146±0.018	0.183±0.012
Monoplex-Newton	0.297±0.051	0.160±0.028	0.385±0.065	0.229±0.030	0.149±0.021	0.172±0.017

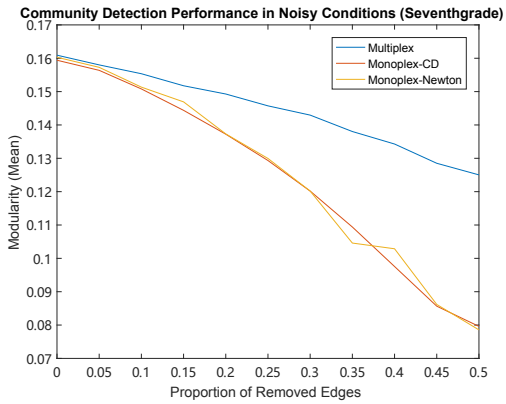
edge from other layers of the network through the common feature matrix, while the monoplex methods only have access to the topological features of that layer. The superiority of our algorithm is most prominent for the ff-tw-yt dataset (0.241 vs. 0.183 & 0.172), where the level of sparsity between layers is most significant (592 vs. 31921 & 42324). This means our algorithm is especially effective when the target layer is noisy (e.g. large number of missing links). We will further verify this property in the next subsection.

3.4.3 Noisy Condition Analysis

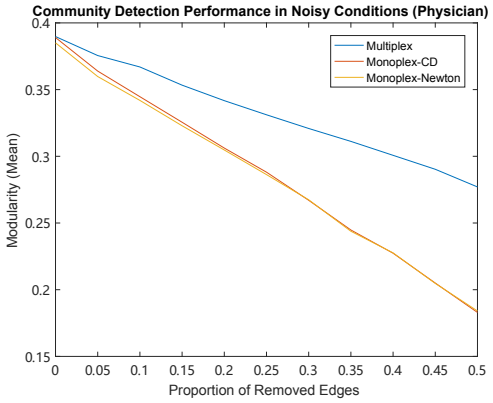
To further investigate the performance of our algorithm under noisy conditions, we randomly remove a set of edges from the target layer. The proportion of removed edges ranges from $[0, 0.5]$, with 0.05 as the step length. For each step, we randomly generate 100 adjacency matrices that have the indicated proportion of edges removed. For each adjacency matrix, we run the algorithms 100 times and compute the mean and the standard error of modularity. Note that when computing modularity, we use the original network information instead of that with edges removed, because the original topological features are more consistent with the underlying community structure, given no prior knowledge. We run the algorithms on the AUCS, Seventhgrade, Physician and Bos datasets and the results with comparison to the monoplex methods are shown in Figure 3-5. We also compute the proportion of performance retained when 50% edges are removed for the three algorithms in Table 3-2, Table 3-3 and Table 3-4, respectively.



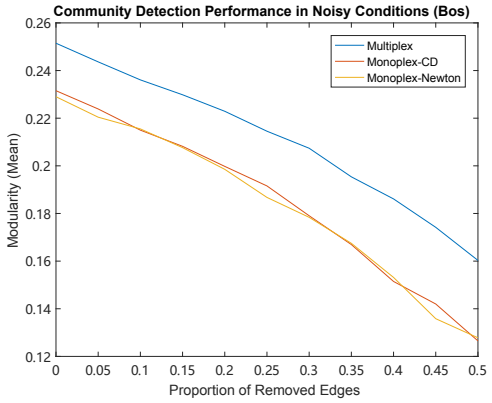
(a)



(b)



(c)



(d)

Figure 3-5 Community detection performance in terms of modularity in noisy conditions.(a)AUCS;(b)Seventhgrade;(c)Physician;(d)Bos

For all the datasets tested, our algorithm outperforms the other two monoplex methods in all noisy conditions. For the AUCS, Seventhgrade and Physician datasets, the performance of our algorithm decreases at a much slower rate than monoplex methods. For the Bos dataset, while our performance decrease at a similar rate to the monoplex methods, a constant performance gap is witnessed. Both cases indicate that our algorithm is far more robust in noisy conditions, as topological information can be retrieved from other layers through the common feature matrix. Specifically, when the noise level is high, e.g. when 50% edges are removed, our algorithm can still retain an remarkable level of performance, as high as 83.7% (in AUCS) and at least 63.7% (in Bos) of its original performance. By comparison, monoplex methods can only retain approximately half

of its performance, which is expected as half of the edges are removed.

Table 3-2 Retained Community Detection Performance in Noisy Condition (Multiplex)

	AUCS	Seventhgrade	Physician	Bos
No Noise	0.317±0.096	0.161±0.030	0.390±0.066	0.251±0.040
50% Edges Removed	0.265±0.097	0.125±0.044	0.277±0.064	0.160±0.040
Retained Performance	83.7%	77.7%	71.1%	63.7%

Table 3-3 Retained Community Detection Performance in Noisy Condition (Monoplex-CD)

	AUCS	Seventhgrade	Physician	Bos
No Noise	0.310±0.065	0.159±0.027	0.389±0.065	0.232±0.031
50% Edges Removed	0.208±0.072	0.080±0.040	0.183±0.047	0.127±0.029
Retained Performance	67.0%	49.9%	47.0%	54.6%

Table 3-4 Retained Community Detection Performance in Noisy Condition (Monoplex-Newton)

	AUCS	Seventhgrade	Physician	Bos
No Noise	0.297±0.051	0.160±0.028	0.385±0.065	0.229±0.030
50% Edges Removed	0.265±0.079	0.079±0.040	0.184±0.046	0.128±0.026
Retained Performance	64.6%	49.0%	47.8%	55.8%

Chapter 4 Conclusion

4.1 Summary

In this thesis, we define a novel objective for the multiplex community detection problem. The goal is to refine community detection results in some layer of a multiplex network with transferred knowledge from other layers. We achieve this goal with an extended representation-based community detection framework, which first learns a shared common feature representation and layer-specific feature representations simultaneously and then cluster the combined representations to obtain the community partition. An implementation of our framework, which includes an extended symmetric non-negative matrix factorization approach for learning representations and k-means for clustering representations, is provided and compared with other representation-based monoplex community detection algorithms on several real-world multiplex network datasets. Results show that our implementation outperforms other methods in terms of modularity, especially when the target layer that receives transferred knowledge is noisy (e.g. much sparser than other layers). We further evaluate our algorithm in high-level noisy conditions by randomly removing a set of edges in the target layer. It is shown that our algorithm is very robust against noise, retaining as high as 83.7% of its original performance even when half of the edges are removed, while other methods can only retain about 50%.

4.2 Future Work

Future research can consider other implementations of the learning and clustering algorithms for more efficiency and accuracy, as well as ability to handle other types of network (e.g. directed and weighted network, temporal network, heterogeneous network, etc). Another interesting direction is to deal with 'negative transfer' in the community detection context.

Acknowledgements

Four years of undergraduate study at University of Electronic Science and Technology of China is an invaluable asset in my life. I would like to take this opportunity to express my deep and sincere gratitude to all the people who supported, motivated and guided me along the way.

First and foremost, I would like to thank my thesis advisor, Prof. Sinno Jialin Pan, for offering me this precious opportunity to intern at his lab at Nanyang Technological University and supporting me with his expert supervision, constructive advice and generous recommendation for my application to graduate schools. The same goes for my supervisor at UESTC, Prof. Junming Shao, who kindly accepted me into his lab when I was a sophomore knowing nothing about research. He guided me to become a qualified scientific researcher step by step, with enlightenment and encouragement. I would cherish my memories of working in their labs.

I would also like to give my warmest thanks to my teachers at school. In particular, I want to thank Prof. Baohua Teng, whose insightful and humorous instruction in class, and helpfulness and care out of class really inspired me to become a teacher in the future, just like him. Without him backing me as my referee, it would not have been possible for me to receive the most outstanding students award of UESTC. I am also indebted to Prof. Shan Gao, who taught me the professionalism of academic writing, and influenced me with her optimism and passion. She was always there in my hardest times, supporting me and encouraging me to go it through.

I am also very grateful to my friends, fellow classmates and seniors for their company and help. It was really a great time to be with them.

Finally, I would like to express my deepest thanks to my family for their unconditional love, understanding and support. This thesis is dedicated to them.

References

[1] S. Fortunato. Community detection in graphs[J]. Physics reports, 2010, 486(3):75–174

[2] M. Newman. Networks: an introduction[M]. Oxford university press, 2010

[3] S. N. Dorogovtsev, J. F. Mendes. Evolution of networks: From biological nets to the Internet and WWW[M]. OUP Oxford, 2013

[4] E. Estrada. The structure of complex networks: theory and applications[M]. Oxford University Press, 2012

[5] S. Li, C. M. Armstrong, N. Bertin, et al. A map of the interactome network of the metazoan C. elegans[J]. Science, 2004, 303(5657):540–543

[6] P. Holme, M. Huss, H. Jeong. Subnetwork hierarchies of biochemical pathways[J]. Bioinformatics, 2003, 19(4):532–538

[7] A. Nagurney. Supply chain network economics: dynamics of prices, flows and profits[M]. Edward Elgar Publishing, 2006

[8] S. R. Proulx, D. E. Promislow, P. C. Phillips. Network thinking in ecology and evolution[J]. Trends in ecology & evolution, 2005, 20(6):345–353

[9] S. Gupta, R. M. Anderson, R. M. May. Networks of sexual contacts: implications for the pattern of spread of HIV.[J]. AIDS (London, England), 1989, 3(12):807–817

[10] D. Knoke. Political networks: the structural perspective[M]. Cambridge University Press, 1994

[11] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, et al. Wireless sensor networks: a survey[J]. Computer networks, 2002, 38(4):393–422

[12] F. Lorrain, H. C. White. Structural equivalence of individuals in social networks[M]. Elsevier, 1977, 67–98

[13] S. Fortunato, D. Hric. Community detection in networks: A user guide[J]. Physics Reports, 2016, 659:1–44

[14] D. Lusseau. The emergent properties of a dolphin social network[J]. Proceedings of the Royal Society of London B: Biological Sciences, 2003, 270(Suppl 2):S186–S188

[15] A. Arenas, A. Fernandez, S. Gomez. Analysis of the structure of complex networks at different resolution levels[J]. New journal of physics, 2008, 10(5):053039

[16] L. A. Adamic, B. A. Huberman. Power-law distribution of the world wide web[J]. science, 2000, 287(5461):2115–2115

[17] A.-L. Barabási, E. Bonabeau. Scale-free networks[J]. Scientific american, 2003, 288(5):60–69

[18] K.-M. Lee, B. Min, K.-I. Goh. Towards real-world complexity: an introduction to multiplex networks[J]. arXiv preprint arXiv:1502.03909, 2015

[19] M. Berlingerio, M. Coscia, F. Giannotti, et al. Multidimensional networks: foundations of structural analysis[J]. World Wide Web, 2013, 16(5-6):567–593

[20] S. Boccaletti, G. Bianconi, R. Criado, et al. The structure and dynamics of multilayer networks[J]. Physics Reports, 2014, 544(1):1–122

[21] F. Battiston, V. Nicosia, V. Latora. Structural measures for multiplex networks[J]. Physical Review E, 2014, 89(3):032804

[22] X. Wang, J. Liu. A layer reduction based community detection algorithm on multiplex networks[J]. Physica A: Statistical Mechanics and its Applications, 2017, 471:244–252

[23] M. Girvan, M. E. Newman. Community structure in social and biological networks[J]. Proceedings of the national academy of sciences, 2002, 99(12):7821–7826

[24] G. W. Flake, S. Lawrence, C. L. Giles, et al. Self-organization and identification of web communities[J]. Computer, 2002, 35(3):66–70

[25] Y. Dourisboure, F. Geraci, M. Pellegrini. Extraction and classification of dense communities in the web[M]. 2007, 461–470

[26] A. W. Rives, T. Galitski. Modular organization of cellular networks[J]. Proceedings of the National Academy of Sciences, 2003, 100(3):1128–1133

[27] V. Spirin, L. A. Mirny. Protein complexes and functional modules in molecular networks[J]. Proceedings of the National Academy of Sciences, 2003, 100(21):12123–12128

[28] R. Guimera, L. A. N. Amaral. Functional cartography of complex metabolic networks[J]. nature, 2005, 433(7028):895

[29] G. Palla, I. Derényi, I. Farkas, et al. Uncovering the overlapping community structure of complex networks in nature and society[J]. nature, 2005, 435(7043):814

[30] P. K. Reddy, M. Kitsuregawa, P. Sreekanth, et al. A graph based approach to extract a neighborhood customer community for collaborative filtering[M]. 2002, 188–200

[31] R. S. Burt. Positions in networks[J]. Social forces, 1976, 55(1):93–122

[32] R. Agrawal, H. Jagadish. Algorithms for searching massive graphs[J]. IEEE Transactions on Knowledge and Data Engineering, 1994, 6(2):225–238

[33] B. Krishnamurthy, J. Wang. On network-aware clustering of web clients[J]. ACM SIGCOMM Computer Communication Review, 2000, 30(4):97–110

[34] J. Shi, J. Malik. Normalized cuts and image segmentation[J]. IEEE Transactions on pattern analysis and machine intelligence, 2000, 22(8):888–905

[35] M. E. Newman. Modularity and community structure in networks[J]. Proceedings of the national academy of sciences, 2006, 103(23):8577–8582

[36] B. W. Kernighan, S. Lin. An efficient heuristic procedure for partitioning graphs[J]. The Bell system technical journal, 1970, 49(2):291–307

[37] J. Friedman, T. Hastie, R. Tibshirani. The elements of statistical learning[M]. Springer series in statistics New York, 2001

[38] J. MacQueen, et al. Some methods for classification and analysis of multivariate observations[M]. 1967, 281–297

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

[40] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, et al. Fast unfolding of communities in large networks[J]. Journal of statistical mechanics: theory and experiment, 2008, 2008(10):P10008

[41] M. Rosvall, C. T. Bergstrom. Maps of random walks on complex networks reveal community structure[J]. Proceedings of the National Academy of Sciences, 2008, 105(4):1118–1123

[42] C. W. Loe, H. J. Jensen. Comparison of communities detection algorithms for multiplex[J]. Physica A: Statistical Mechanics and its Applications, 2015, 431:29–45

[43] O. Boutemine, M. Bouguessa. Mining community structures in multidimensional networks[J]. ACM Transactions on Knowledge Discovery from Data (TKDD), 2017, 11(4):51

[44] M. Berlingerio, M. Coscia, F. Giannotti. Finding and characterizing communities in multidimensional networks[M]. 2011, 490–494

[45] L. Tang, X. Wang, H. Liu. Community detection via heterogeneous interaction analysis[J]. Data mining and knowledge discovery, 2012, 25(1):1–33

[46] M. De Domenico, A. Lancichinetti, A. Arenas, et al. Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems[J]. Physical Review X, 2015, 5(1):011027

[47] B. Boden, S. Günnemann, H. Hoffmann, et al. Mining coherent subgraphs in multi-layer graphs with edge labels[M]. 2012, 1258–1266

[48] M. Hmimida, R. Kanawati. Community detection in multiplex networks: A seed-centric approach.[J]. NHM, 2015, 10(1):71–85

[49] L. Gauvin, A. Panisson, C. Cattuto. Detecting the community structure and activity patterns of temporal networks: a non-negative tensor factorization approach[J]. PloS one, 2014, 9(1):e86028

[50] G. P. C. Fung, J. X. Yu, H. Lu, et al. Text classification without negative examples revisit[J]. IEEE transactions on Knowledge and Data Engineering, 2006, 18(1):6–20

[51] J. Blitzer, M. Dredze, F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification[M]. 2007, 440–447

[52] P. Wu, T. G. Dietterich. Improving SVM accuracy by training on auxiliary data sources[M]. 2004, 110

[53] S. J. Pan, V. W. Zheng, Q. Yang, et al. Transfer learning for wifi-based indoor localization[M]. 2008, 6

[54] V. C. Raykar, B. Krishnapuram, J. Bi, et al. Bayesian multiple instance learning: automatic feature selection and inductive transfer[M]. 2008, 808–815

[55] A. Arnold, R. Nallapati, W. W. Cohen. A comparative study of methods for transductive transfer learning[M]. 2007, 77–82

[56] S. J. Pan, Q. Yang. A survey on transfer learning[J]. IEEE Transactions on knowledge and data engineering, 2010, 22(10):1345–1359

[57] W. Dai, Q. Yang, G.-R. Xue, et al. Boosting for transfer learning[M]. 2007, 193–200

[58] N. D. Lawrence, J. C. Platt. Learning to learn with the informative vector machine[M]. 2004, 65

[59] L. Mihalkova, T. Huynh, R. J. Mooney. Mapping and revising Markov logic networks for transfer learning[M]. 2007, 608–614

[60] J. Huang, A. Gretton, K. M. Borgwardt, et al. Correcting sample selection bias by unlabeled data[M]. 2007, 601–608

[61] J. Blitzer, R. McDonald, F. Pereira. Domain adaptation with structural correspondence learning[M]. 2006, 120–128

[62] W. Dai, Q. Yang, G.-R. Xue, et al. Self-taught clustering[M]. 2008, 200–207

[63] Z. Wang, Y. Song, C. Zhang. Transferred dimensionality reduction[M]. 2008, 550–565

[64] I. Psorakis, S. Roberts, M. Ebden, et al. Overlapping community detection using bayesian non-negative matrix factorization[J]. Physical Review E, 2011, 83(6):066114

[65] I. H. Witten, E. Frank, M. A. Hall, et al. Data Mining: Practical machine learning tools and techniques[M]. Morgan Kaufmann, 2016

[66] S. Wasserman, K. Faust. Social network analysis: Methods and applications[M]. Cambridge university press, 1994

[67] I. Borg, P. J. Groenen. Modern multidimensional scaling: Theory and applications[M]. Springer Science & Business Media, 2005

[68] A. Y. Ng, M. I. Jordan, Y. Weiss. On spectral clustering: Analysis and an algorithm[M]. 2002, 849–856

[69] M. E. Newman. Finding community structure in networks using the eigenvectors of matrices[J]. Physical review E, 2006, 74(3):036104

[70] L. Yang, X. Cao, D. He, et al. Modularity Based Community Detection with Deep Learning.[M]. 2016, 2252–2258

[71] L. Rossi, M. Magnani. Towards effective visual analytics on multiplex and multilayer networks[J]. Chaos, Solitons & Fractals, 2015, 72:68–76

[72] Z. He, S. Xie, R. Zdunek, et al. Symmetric nonnegative matrix factorization: Algorithms and applications to probabilistic clustering[J]. IEEE Transactions on Neural Networks, 2011, 22(12):2117–2131

[73] S. Lu, M. Hong, Z. Wang. A nonconvex splitting method for symmetric nonnegative matrix factorization: Convergence analysis and optimality[M]. 2017, 2572–2576

[74] D. Kuang, S. Yun, H. Park. SymNMF: nonnegative low-rank approximation of a similarity matrix for graph clustering[J]. Journal of Global Optimization, 2015, 62(3):545–574

[75] C. H. Ding, T. Li, M. I. Jordan. Convex and semi-nonnegative matrix factorizations[J]. IEEE transactions on pattern analysis and machine intelligence, 2010, 32(1):45–55

[76] A. Lancichinetti, S. Fortunato, F. Radicchi. Benchmark graphs for testing community detection algorithms[J]. Physical review E, 2008, 78(4):046110

[77] M. Vickers, S. Chan. Representing classroom social structure[J]. Victoria Institute of Secondary Education, Melbourne, 1981

[78] J. Coleman, E. Katz, H. Menzel. The diffusion of an innovation among physicians[J]. Sociometry, 1957, 20(4):253–270

[79] C. Stark, B.-J. Breitkreutz, T. Reguly, et al. BioGRID: a general repository for interaction datasets[J]. Nucleic acids research, 2006, 34(suppl_1):D535–D539

[80] M. De Domenico, V. Nicosia, A. Arenas, et al. Structural reducibility of multilayer networks[J]. Nature communications, 2015, 6:ncomms7864

[81] B. L. Chen, D. H. Hall, D. B. Chklovskii. Wiring optimization can relate neuronal structure and function[J]. Proceedings of the National Academy of Sciences of the United States of America, 2006, 103(12):4723–4728

[82] M. De Domenico, M. A. Porter, A. Arenas. MuxViz: a tool for multilayer analysis and visualization of networks[J]. Journal of Complex Networks, 2015, 3(2):159–176

[83] M. Magnani, L. Rossi. The ml-model for multi-layer social networks[M]. 2011, 5–12

[84] V. Nicosia, V. Latora. Measuring and modeling correlations in multiplex networks[J]. Physical Review E, 2015, 92(3):032805

[85] J. Tang, J. Zhang, L. Yao, et al. Arnetminer: extraction and mining of academic social networks[M]. 2008, 990–998

[86] A. Strehl, J. Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions[J]. Journal of machine learning research, 2002, 3(Dec):583–617

[87] L. Danon, A. Diaz-Guilera, J. Duch, et al. Comparing community structure identification[J]. Journal of Statistical Mechanics: Theory and Experiment, 2005, 2005(09):P09008

[88] A. Vandaele, N. Gillis, Q. Lei, et al. Efficient and non-convex coordinate descent for symmetric nonnegative matrix factorization[J]. IEEE Transactions on Signal Processing, 2016, 64(21):5571–5584

Finding community structure in networks using the eigenvectors of matrices

M. E. J. Newman
Department of Physics and Center for the Study of Complex Systems, University of Michigan, Ann Arbor, Michigan 48109, USA
(Received 19 May 2006; published 11 September 2006)

We consider the problem of detecting communities or modules in networks, groups of vertices with a higher-than-average density of edges connecting them. Previous work indicates that a robust approach to this problem is the maximization of the benefit function known as "modularity" over possible divisions of a network. Here we show that this maximization process can be written in terms of the eigenspectrum of a matrix we call the modularity matrix, which plays a role in community detection similar to that played by the graph Laplacian in graph partitioning calculations. This result leads us to a number of possible algorithms for detecting community structure, as well as several other results, including a spectral measure of bipartite structure in networks and a centrality measure that identifies vertices that occupy central positions within the communities to which they belong. The algorithms and measures proposed are illustrated with applications to a variety of real-world complex networks.

DOI: 10.1103/PhysRevE.74.036104 PACS number(s): 89.75.Hc, 05.10.−a, 02.10.Ud, 87.23.Ge

I. INTRODUCTION

Networks have attracted considerable recent attention in physics and other fields as a foundation for the mathematical representation of a variety of complex systems, including biological and social systems, the Internet, the worldwide web, and many others [1–4]. A common feature of many networks is "community structure," the tendency for vertices to divide into groups, with dense connections within groups and only sparser connections between them [5,6]. Social networks [5], biochemical networks [7–9], and information networks such as the web [10] have all been shown to possess strong community structure, a finding that has substantial practical implications for our understanding of the systems these networks represent. Communities are of interest because they often correspond to functional units such as cycles or pathways in metabolic networks [8,9,11] or collections of pages on a single topic on the web [10], but their influence reaches further than this. A number of recent results suggest that networks can have properties at the community level that are quite different from their properties at the level of the entire network, so that analyses that focus on whole networks and ignore community structure may miss many interesting features.

For instance, in some social networks one finds individuals with different mean numbers of contacts in different groups; the individuals in one group might be gregarious, having many contacts with others, while the individuals in another group might be more reticent. An example of this behavior is seen in networks of sexual contacts, where separate communities of high- and low-activity individuals have been observed [12,13]. If one were to characterize such a network by quoting only a single figure for the average number of contacts an individual has, one would be missing features of the network directly relevant to questions of scientific interest such as epidemiological dynamics [14].

It has also been shown that vertices' positions within communities can affect the role or function they assume. In social networks, for example, it has long been accepted that individuals who lie on the boundaries of communities, bridging gaps between otherwise unconnected people, enjoy an unusual level of influence as the gatekeepers of information flow between groups [15–17]. A surprisingly similar result is found in metabolic networks, where metabolites that straddle the boundaries between modules show particular persistence across species [8]. This finding might indicate that modules in metabolic networks possess some degree of functional independence within the cell, allowing vertices central to a module to change or disappear with relatively little effect on the rest of the network, while vertices on the borders of modules are less able to change without affecting other aspects of the cellular machinery.

One can also consider the communities in a network themselves to form a higher-level metanetwork, a coarse-grained representation of the full network. Such coarse-grained representations have been used in the past as tools for visualization and analysis [18] but more recently have also been investigated as topologically interesting entities in their own right. In particular, networks of modules appear to have degree distributions with interesting similarities to but also some differences from the degree distributions of other networks [9] and may also display so-called preferential attachment in their formation [19], indicating the possibility of distinct dynamical processes taking place at the level of the modules.

For all of these reasons and others besides there has been a concerted effort in recent years within the physics community and elsewhere to develop mathematical tools and computer algorithms to detect and quantify community structure in networks. A huge variety of community detection techniques have been developed, based variously on centrality measures, flow models, random walks, resistor networks, optimization, and many other approaches [5,8,9,18,20–35]. For reviews see Refs. [6,36].

In this paper we focus on one approach to community detection that has proven particularly effective, the optimization of the benefit function known as "modularity" over the possible divisions of a network. Methods based on this approach have been found to produce excellent results in standardized tests [36,37]. Unfortunately, exhaustive optimization of the modularity demands an impractically

利用矩阵特征值发现网络中的社团结构

M. E. J. Newman

物理系、复杂系统研究中心，密歇根大学，安娜堡，密歇根 48109, 美国

（2006年5月19日收稿，2006年9月11日发表）

本文考虑网络中的社团或者模块检测问题，即发现若干组连接他们之间边的密度高于平均值的点。之前的工作表明解决这一问题的一个稳健的方法是在可能的网络划分结果上最大化收益函数“模块度”。本文证明了这种最大化过程可以以我们称为模块度矩阵的特征谱给出，这种矩阵在社团检测中起的作用与图拉普拉斯矩阵在图划分计算中起的作用类似。这一结果引出了若干社团检测的可能算法，以及包括一种网络中二分结构的谱度量和说明点在它所属社团中的中心位置程度的中心度量在内的其他结果。本文提出的算法以及度量方法通过在多种真实场景的复杂网络的应用进行展示。

DOI: 10.1103/PhysRevE.74.036104　　PACS号: 89.75.Hc, 05.10.−a, 02.10.Ud, 87.23.Ge

I. 引言

网络作为包括生物系统、社会系统、互联网、万维网等多种复杂系统的数学表征的基础，近期吸引了物理界和其他领域的许多关注[1–4]. 对于很多网络而言，一种常见的特性就是“社团结构”，即网络中的点有着分成组的趋势，并且组内有着较强的连接性和组间反之[5,6]。社交网络[5]，生化网络[7–9]，以及诸如万维网的信息网络[10]都被证明拥有很强的社团结构，这一发现对理解这些网络所代表的系统有着巨大的实用意义。人们之所以对社团感兴趣，是因为其在代谢网络中对应于诸如环和路径这样的功能单元[8,9,11]，以及万维网中关于同一个话题的网页的集合[10]，当然社团的影响远不止这些。多个近期的研究结果表明，网络在社团级别上有着许多与整个网络级别上不同的性质，因此只注重于整个网络而忽视社团结构的分析可能会错过许多有趣的特性。

比如说，在一些社交网络中，人们发现拥有不同平均联系人数目的个体在不同的团体里面；某个团体里的个体可能爱社交，因此有着比其他人多的联系人，而另一个团体的人可能比较含蓄。这种现象的一个例子是性交网络，在这种网络中人们发现活动频繁与活动不频繁的个体位于不同的社团中[12,13]。如果只通过引用个体的平均联系人数目这一个数据来刻画这种网络，与诸如流行病动态等具有科学意义的问题直接相关的网络特征就会被忽视[14]。

许多研究还表明，点在社团中的位置影响他们的作用或者功能。比如，在社交网络中，一种广为接受的观点是那些在网络边缘，连接那些没有他们就没法互相连接的人的个体，作为团体间信息流的看门人有着超常的影响力[15–17]。一个令人震惊般类似的结果也在代谢网络中被发现：跨越多个模块边界的代谢物存在于许多不同物种中[8]。这一结果可能意味着在代谢网络中，模块在细胞内有着一定程度上的功能独立性，这让处于模块中心的点的改变与消失较少地影响网络的其他部分，而在模块边界上的点则不能在不影响细胞机器的其他方面的情况下进行改变。

人们也可认为，网络中的社团自身也组成了一个更高一层的元网络，即一种对整个网络的粗粒度表征。这种粗粒度表征在过去被用作可视化和分析的工具[18]，但在最近也作为他们自身被看做拓扑上有意义的实体而研究。特别地是，模块组成的网络似乎与其他网络在度分布上有着有趣的类似点但也有不同之处[9]，并且在他们的形成过程中还展现出被称为优先连接的特点[19]，这意味着在模块级别发生着与众不同的动力学过程。

因为上述的所有以及一些其他的原因，近年来在物理界和其他学术领域，学者们十分努力地开发数学工具和计算机算法来发现和量化网络中的社团结构。各种各样的社团检测方法被开发出来：基于中心度量、流模型、随机行走、电阻网络、最优化等[5,8,9,18,20–35]。可以参考[6,36]给出的综述。