Heterogeneous Embedding via Aggregating Multiple Sources

Xiaoxiao Shi · Philip S. Yu

Received: 5 September 2012 / Revised: 25 December 2012 / Accepted: 20 April 2013 /
Published online: 10 April 2014
© Springer-Verlag Berlin Heidelberg 2014

Abstract Aggregating multiple correlated data sources may help improve the learning performance of a given task. For example, in recommendation problems, one can aggregate (1) user profile database (e.g., genders, age, etc.), (2) users' log data (e.g., clickthrough data, purchasing records, etc), and (3) users' social network (useful in social targeting) to build a recommendation model. All these data sources provide informative but heterogeneous features. For instance, user profile database usually has nominal features reflecting users' background, log data provides term-based features about users' historical behaviors, and social network database has graph relational features. Given multiple heterogeneous data sources, one important challenge is to find a unified embedding feature subspace that captures the knowledge from all sources. To this aim, we propose a principle of *collective component analysis* (CoCA), in order to find the optimal embedding across a mixture of vector-based features and graph relational features. The CoCA principle is to find a feature subspace with maximal variance under two constraints. First, there should be consensus among the projections from different feature spaces. Second, the similarity between connected data (in any of the network databases) should be maximized. The optimal solution is obtained by solving an eigenvalue problem. Moreover, we discuss how to use prior knowledge to distinguish informative data sources, and optimally weight them in CoCA. Since there is no previous model that can be directly applied to solve the problem, we devised a straightforward comparison method by performing dimensionality reduction on the concatenation of the data sources. Three sets of experiments show that CoCA substantially outperforms the comparison method.

X. Shi (✉) · P. S. Yu
Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA
e-mail: xiao.x.shi@gmail.com; xshi9@uic.edu

P. S. Yu
e-mail: psyu@uic.edu

Keywords Embedding · Heterogeneous features · Dimensionality reduction · PCA · Multiple data sources

1 Introduction

Multiple related data sources may be available for a given task. An example is user-oriented recommendation system. For this task, related data sources can be (1) user profile database (as shown in Fig. 1a), (2) users' log data (as shown in Fig. 1b), and (3) users' social connections with other users (as shown in Fig. 1c). Each single data source may not be informative enough to build an accurate model, but the combination of them may provide important knowledge. For instance, in Fig. 1a, given that we observe several users' opinions on the new iPhone, what is Lily's opinion about iPhone? Note that it is difficult to infer the answer just from any one of the databases (Fig. 1a, b or c). However, if we combine the knowledge from all three data sources, it can be inferred that Lily has a similar background/behavior to Bob; thus, she may like iPhone as Bob likes. These mixture types of data are rich in information, and it is desirable to build learning models by using the heterogeneous features collectively. We call this framework *heterogeneous learning*. More specifically, the task can have

Fig. 1 Heterogeneous feature spaces in computational advertising. It is difficult to analyze Lily's opinion on iPhone if we just consider only one of the data sources (a–c). However, if we combine the information from all three sources, we can infer Lily's opinion based on her similarity (background, behavior, social connection) to Bob. A unified embedding space generated from this example is given in (d)

(1) multiple vector-based data sources, and (2) multiple graph relational data sources. The aim is to devise learning algorithms to make good use of all types of features collectively.

Note that one essential challenge of heterogeneous learning is to conduct dimensionality reduction, in order to find a unified embedding feature space that captures the information from all sources. However, this is a challenging task. First, it is not clear how to find only one (unified) feature subspace, given the heterogeneous features from multiple sources. Second, it is not obvious how to find such embedding with multiple graph relational features (eg, Fig. 1c), and for an even more daunting task with a mixture of vector-based features and relational features. To solve the problem, we propose a principle **Collective Component Analysis (CoCA)**. The CoCA principle is to find a feature subspace to maximize the variance of the projected data with the following two constraints.

1. First, all vector-based feature spaces have to be projected onto the same reduced feature space consensually. For example, Fig. 1a, b provide two sets of heterogeneous vector-based features. They both have to agree on and to be projected to a common feature subspace as Fig. 1d, such that their original data structures (e.g., data similarity) are preserved.
2. Second, if there are relational features (as in Fig. 1c), the connected data are preferred to be similar in the reduced feature space. For example, Lily and Tom do not have connections in the graph Fig. 1c. Hence, as in the unified feature subspace Fig. 1d, Lily and Bob are more similar than Lily and Tom.

We first analyze the CoCA principle under two special cases. They are cases where (1) there are multiple vector-based data sources (as shown in Fig. 1a, b), and (2) there are multiple graph relational data sources (as shown in Fig. 1c). We then combine the two cases, and discuss the situation where we have a mixture of vector-based features and relational features. A corresponding optimization problem is devised, and the final solution is obtained via the eigenvectors of certain matrix. Furthermore, given multiple sources, we design a quadratic programming problem to identify sources that are more informative, and optimally weight them to learn a better projection. This is an important step since some of the data sources may contain substantial noise, and it is desirable to reduce their effect. Empirical studies include three sets of real-world datasets ranging from handwritten numbers recognition, document classification and terrorist attack detection. Since there was no previous method that can solve the same problem, we devised a straightforward comparison embedding model that performed dimension reduction on the concatenation of the data sources. In all experiments, CoCA substantially outperforms the comparison method by as much as 50 %. In summary, the proposed CoCA model has the following properties.

- Firstly, it finds a unified feature subspace from *multiple* vector-based data sources and *multiple* graph relational datasets.
- Secondly, it is a robust model since a quadratic programming framework is incorporated to reduce the effect of noise.
- Thirdly, it is an efficient model. It has the same algorithm complexity as traditional dimension reduction algorithms such as PCA, even though (1) heterogeneous data

Table 1 Symbol definition

Symbol	Definition
$\mathbf{X}^{(j)} = [\mathbf{x}_1^{(j)}, \mathbf{x}_2^{(j)}, \dots, \mathbf{x}_m^{(j)}] \in \mathbb{R}^{d_j \times m}$	The dataset in the j -th source where $\mathbf{x}_i^{(j)} \in \mathbb{R}^{d_j}$ is the i -th data
$\mathbf{M}^{(k)} \in \mathbb{R}^{m \times m}$	The adjacency matrix of the k -th relational graph
$\mathbf{u}^{(j)}$	Projection basis for the j -th source (the j -th feature space)
Φ	The embedding space

sources are considered, and (2) a quadratic programming framework is used. More details can be found at Sect. 3.5.

2 Problem Formulation

In this section, we first formally define heterogeneous learning, and then introduce the CoCA principle for heterogeneous embedding. In heterogeneous learning, the data can be described in heterogeneous feature spaces from multiple sources. Note that in this paper, we mainly study two categories of heterogeneous features: vector-based features and graph relational features. For traditional vector-based features, we use column vector $\mathbf{x}_i^{(j)} \in \mathbb{R}^{d_j}$ to denote the i -th data in the j -th source (or the j -th feature space) whose dimension is d_j . In a matrix form, we denote $\mathbf{X}^{(j)} \in \mathbb{R}^{d_j \times m}$ as the set of data in the j -th feature space where m is the sample size. Different from vector-based features, graph relational features describe the relationships among the data. In other words, they are graphs representing similarity of the data, and the adjacency matrices of the graphs are considered as relational features. Specifically, we denote $\mathbf{M}^{(k)} \in \mathbb{R}^{m \times m}$ as the adjacency matrix of the k -th graph where m is the sample size. Note that the matrix $\mathbf{M}$ can be a “soft” adjacency matrix such that $\mathbf{M}^{(k)}(i, j)$ is the degree of similarity between the i -th data and the j -th data in the k -th graph. It is large (closed to 1) if the two data are strongly correlated; otherwise it is 0. We assume that the features from the same data source are from the same feature space, and hence each data source has a corresponding feature space. Thus, heterogeneous learning is a machine learning scenario where the data have heterogeneous feature spaces. For example, we have p data sources providing vector-based features $\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(p)}$ and q data sources providing relational features $\mathbf{M}^{(1)}, \dots, \mathbf{M}^{(q)}$. The aim is to derive learning models (classification or clustering) by collectively using the $p + q$ feature spaces (Table 1).

One innate challenge of heterogeneous learning is finding a low-dimensional embedding space. First, low-dimensional embedding is a necessary step to increase the learning accuracy and efficiency. Let us consider a straightforward strategy of utilizing the heterogeneous features by directly joining the $p + q$ sources together. In such a case, the dimension goes up to $\sum_i^p d_i + qm$, which is approximately $p + q$ times of the original dimension (if assuming that the data size is the same as the feature size). In a linear classification model, it means that the number of variables increases by $p + q$ times. Given a fixed number of labeled examples, it may easily encounter the under-fitting problem. In other words, the number of labeled examples is far from enough to determine the increasing number of model variables. Hence, it is necessary to reduce its overall dimension. Second, low-dimensional embedding is a necessary

step to reduce noise. This is because different sources may have different data qualities. Some may contain substantial noise that hurt the learning performance. Hence, low-dimensional embedding is also a necessary step to dig out informative features to reduce noise. However, it is a very challenging task. The first challenge is how to find a unified reduced feature subspace based on multiple feature spaces. The second challenge is how to use the graph relational features in finding the feature subspace.

We devise a CoCA principle for finding the optimal embedding for heterogeneous data, and further perform low-dimensional embedding. The idea is derived from PCA, which aims at maximizing the variance of the projected data to maximize the data separation. In addition, we introduce two more constraints. The first is to force all heterogeneous feature spaces to be projected onto the same subspace. The second is to maximize the similarity of the data with connections as reflected by the relational features. Similar to PCA, the data are normalized to have zero mean, and the aim is to find the orthogonal linear projection basis $\mathbf{u}^{(j)}$ for all heterogeneous feature spaces ($j = 1, 2, \dots$). The general objective function can be written as follows:

$$\max_{\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \dots} \gamma(\mathbf{u}) + c(\mathbf{u}) + \alpha \chi(\mathbf{u}) \quad (1)$$

where $\gamma(\mathbf{u})$ is the variance of the projected data, which depends on the projection basis $\mathbf{u} = [\mathbf{u}^{(1)T}, \mathbf{u}^{(2)T}, \dots]^T$.¹ The second term $c(\mathbf{u})$ is the similarity among connected data. In other words, if the i -th data and the j -th data are connected in any of the graph data sources, they should also be similar in the projected space.

The last term $\chi(\mathbf{u})$ forces the projections from different data sources to be similar. More specifically, if the projections of the i -th and j -th data sources are Φ_i and Φ_j respectively, then the difference between Φ_i and Φ_j should be very small. For example, suppose we have two objects and two data sources, denote the projections learned from the two data sources are $\hat{\Phi}_1$ and $\hat{\Phi}_2$ respectively. Further assume that $\hat{\Phi}_1 = [0, 1]^T$ and $\hat{\Phi}_2 = [1, 0]^T$. If we maximize the similarity of the two projected spaces via $\chi(\mathbf{u})$, we will get the final projections as $\Phi_1 = \hat{\Phi}_1$ and $\Phi_2 = \hat{\Phi}_2 \hat{I}$ where $\hat{I} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. As such, both feature spaces are projected into $[0, 1]^T$. It is equally likely to be both projected to $[1, 0]^T$, but the structure (separation) of the data is the same. Hence, the term $\chi(\mathbf{u})$ tries to find a consensus feature subspace across all data sources. Moreover, α is a parameter to control how strongly we want the data to be projected onto a unified subspace.

Note that by solving Eq. 1, we can obtain multiple projections $\mathbf{u}^{(j)}$ where $j = 1, 2, \dots$ (one for each feature space). According to the CoCA principle, all these projections are similar to each other. Hence, the final projection is the expected value of the feature subspace:

$$\Phi = \mathbb{E}(\mathbf{X}^T \mathbf{u}) = \frac{1}{n} \sum_{j=1}^n \mathbf{X}^{(j)T} \mathbf{u}^{(j)} \quad (2)$$

¹ The dependency of $\gamma()$ on the data is obvious, and we do not write it explicitly for simplicity.

where n is the number of data sources (feature spaces), and $\mathbf{X}^{(j)T} \mathbf{u}^{(j)}$ is the projected space of the j -th data source, and $\Phi \in \mathbb{R}^{m \times d}$ is the desired unified feature subspace with reduced dimension d .

3 Collective Component Analysis

In the last section, we introduce collective component analysis principle for unsupervised heterogeneous embedding, which is described in Eq. 1. In this section, for the sake of clarity, we first study the CoCA principle on two special cases where (1) all sources provide only vector-based features, and where (2) all sources provide only graph relational features. We then combine the two cases, and derive a general CoCA algorithm for vector-based features and relational features simultaneously. The most important challenges of the problem can be summarized as follows:

- How should we formulate the term $\chi(\mathbf{u})$ to force the similarity of the projected spaces?
- How should we solve the optimization in Eq. 1, especially given that it contains multiple objectives?
- How should we reduce the impact of the noisy data sources?

3.1 CoCA on Multiple Vector-Based Feature Spaces

We first consider the problem of finding a consensus feature subspace across multiple sources with only vector-based features. Without loss of generality, we first discuss the case where the desired dimension is one.

Definition 1 Given p data sources with different feature spaces $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \dots, \mathbf{X}^{(p)}$, the aim is to find a feature subspace that (1) maximizes the variance of the projected data, and that (2) maximizes the “agreement” of all projections:

$$\begin{aligned} \max_{\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(p)}} \quad & \sum_{j=1}^p \omega_j \sum_{i=1}^m \left(\mathbf{x}_i^{(j)T} \mathbf{u}^{(j)} \right)^2 \\ & - \alpha \sum_{j=1}^p \sum_{h=1}^p \sum_{i=1}^m \left\| \left(\mathbf{x}_i^{(j)T} \mathbf{u}^{(j)} \right) - \left(\mathbf{x}_i^{(h)T} \mathbf{u}^{(h)} \right) \right\|^2 \\ \text{s.t. } \quad & \|\mathbf{u}^{(j)}\| = 1 \text{ where } j = 1, \dots, p \end{aligned} \quad (3)$$

where $\mathbf{x}_i^{(j)T} \mathbf{u}^{(j)}$ is the length of the projection of $\mathbf{x}_i^{(j)}$ on $\mathbf{u}^{(j)}$, and $\|\cdot\|$ is the Euclidean norm to ensure the agreement of projections from different data sources. In other words, although the same object (e.g., $\mathbf{x}_i$) has different features from different feature spaces (e.g., $\mathbf{x}_i^{(j)}$ and $\mathbf{x}_i^{(h)}$), it has only one projected value in the final subspace (i.e., to force $\mathbf{x}_i^{(j)T} \mathbf{u}^{(j)} = \mathbf{x}_i^{(h)T} \mathbf{u}^{(h)}$). Furthermore, ω_j is the weight of importance of the j -th data source (or the j -th feature space). Ideally, we should give higher weights for informative sources. At the moment, we set all weights to be equal. In Sect. 3.4, a

separate method is introduced to obtain optimal weights to better use the informative sources. Note that although the optimization problem in Eq. 3 is straightforward, it is not easy to solve. We derive an equivalent optimization problem as follows:

Lemma 1 *The optimization problem in Eq. 3 is equivalent to the following optimization problem:*

$$\begin{aligned} \max_{\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(p)}} & \sum_{j=1}^p \omega_j \sum_{i=1}^m \left(\mathbf{x}_i^{(j)T} \mathbf{u}^{(j)} \right)^2 \\ & + \alpha \sum_{j=1}^p \sum_{h=1}^p \sum_{i=1}^m \left(\mathbf{x}_i^{(j)T} \mathbf{u}^{(j)} \right)^T \left(\mathbf{x}_i^{(h)T} \mathbf{u}^{(h)} \right) \\ \text{s.t. } & \|\mathbf{u}^{(j)}\| = 1 \quad \text{where } j = 1, \dots, p \end{aligned} \quad (4)$$

Proof The difference between Eqs. 3 and 4 is the last term in Eq. 3 and the last term in Eq. 4. We next prove that they are equivalent. Note that

$$\begin{aligned} & - \left\| \left(\mathbf{x}_i^{(j)T} \mathbf{u}^{(j)} \right) - \left(\mathbf{x}_i^{(h)T} \mathbf{u}^{(h)} \right) \right\|^2 \\ & = - \left(\mathbf{x}_i^{(j)T} \mathbf{u}^{(j)} \right)^T \left(\mathbf{x}_i^{(j)T} \mathbf{u}^{(j)} \right) - \left(\mathbf{x}_i^{(h)T} \mathbf{u}^{(h)} \right)^T \left(\mathbf{x}_i^{(h)T} \mathbf{u}^{(h)} \right) \\ & \quad + 2 \left(\mathbf{x}_i^{(j)T} \mathbf{u}^{(j)} \right)^T \left(\mathbf{x}_i^{(h)T} \mathbf{u}^{(h)} \right) \end{aligned} \quad (5)$$

Note that the first two terms can be incorporated into the first term in Eq. 3, and the last term is controlled by the parameter α . Thus, maximizing Eq. 3 is equivalent to maximize Eq. 4.

The optimization problem in Eq. 4 can be written in a matrix form as follows:

$$\begin{aligned} \max_{\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(p)}} & \sum_{j=1}^p \omega_j \mathbf{u}^{(j)T} \mathbf{X}^{(j)} \mathbf{X}^{(j)T} \mathbf{u}^{(j)} \\ & + \alpha \left(\sum_{j=1}^p \sum_{h=1}^p \mathbf{u}^{(j)T} \mathbf{X}^{(j)} \mathbf{X}^{(h)T} \mathbf{u}^{(h)} \right) \\ \text{s.t. } & \|\mathbf{u}^{(j)}\| = 1 \quad \text{where } j = 1, \dots, p \end{aligned} \quad (6)$$

Note that at the moment, we set the weight ω_j to be $\frac{1}{p}$. In Sect. 3.4, a separate method is introduced to obtain the optimal ω_j with some prior knowledge. Hence, the unknown variables in Eq. 6 are the projection bases $\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(p)}$. We further rewrite the optimization problem into a more compact form:

$$\begin{aligned} \max_{\mathbf{u}} \mathbf{u}^T \mathbf{X} \mathbf{u} \\ \text{s.t. } \|\mathbf{u}^{(j)}\| = 1 \quad \text{where } j = 1, \dots, p \end{aligned} \quad (7)$$

where $\mathbf{u} = [\mathbf{u}^{(1)T}, \dots, \mathbf{u}^{(p)T}]^T \in \mathbb{R}^{(d_1 + \dots + d_p) \times 1}$ and

$$\mathbf{X} = \begin{bmatrix} \omega_1 \mathbf{X}^{(1)} \mathbf{X}^{(1)T} & \alpha \mathbf{X}^{(1)} \mathbf{X}^{(2)T} & \dots & \alpha \mathbf{X}^{(1)} \mathbf{X}^{(p)T} \\ \alpha \mathbf{X}^{(2)} \mathbf{X}^{(1)T} & \omega_2 \mathbf{X}^{(2)} \mathbf{X}^{(2)T} & \dots & \alpha \mathbf{X}^{(2)} \mathbf{X}^{(p)T} \\ \dots & \dots & \dots & \dots \\ \alpha \mathbf{X}^{(p)} \mathbf{X}^{(1)T} & \alpha \mathbf{X}^{(p)} \mathbf{X}^{(2)T} & \dots & \omega_p \mathbf{X}^{(p)} \mathbf{X}^{(p)T} \end{bmatrix} \quad (8)$$

To solve Eq. 7, we first form its Lagrangian formula

$$\max_{\mathbf{u}} \mathbf{u}^T \mathbf{X} \mathbf{u} - \lambda \sum_{j=1}^p (\mathbf{u}^{(j)T} \mathbf{u}^{(j)} - 1) = \mathbf{u}^T \mathbf{X} \mathbf{u} - \lambda \mathbf{u}^T \mathbf{u} + \lambda p \quad (9)$$

where λ is the Lagrangian multiplier associated with the constraints. It is well known that the eigenvector corresponding to the largest eigenvalue of $\mathbf{X}$ maximizes Eq. 9 (e.g., in [6]). The final projected data is $\Phi = \frac{1}{p} \sum_{j=1}^p \mathbf{X}^{(j)T} \mathbf{u}^{(j)}$ as discussed in Eq. 2. Note that although we analyze the case where the desired dimension is one, it is not difficult to extend it to multiple dimensions by using the eigenvectors of the top d eigenvalues.

3.2 CoCA on Multiple Relational Feature Spaces

For relational features, the aim is to find a subspace that maximizes the similarity among the data with connections. In other words, we maximize the second term in Eq. 1. Note that since there are no vector-based features, we do not have to find the linear projection basis $\mathbf{u}$ as in Sect. 3.1. Instead, we can find the projected data Φ directly. The principle is: if data a and data b are connected, they should have similar projected values $\Phi(a)$ and $\Phi(b)$. Also note that since the projected data Φ are normalized to have zero mean as in PCA, maximizing the similarity of $\Phi(a)$ and $\Phi(b)$ is approximately equivalent to maximizing their product $\Phi(a) \cdot (\Phi(b))^T$, in order to ensure that they have the same sign. Hence, the optimization problem is as follows:

$$\begin{aligned} \max \sum_{j=1}^q \tilde{\omega}_j \sum_{a=1}^m \sum_{b=1}^m \mathbf{M}^{(j)}(a, b) \cdot \Phi(a) \cdot (\Phi(b))^T \\ \text{s.t. } \|\Phi\| = 1 \end{aligned} \quad (10)$$

where $\mathbf{M}^{(j)}(a, b)$ is large if data a and data b have strong similarity in the j -th relational graph (e.g., equals to 1 if connected, 0 otherwise), and $\Phi(a)$, $\Phi(b)$ are the projected features of a and b respectively. Moreover, $\tilde{\omega}_j$ is the weight of importance of the j -th source. In Sect. 3.4, a separate method is introduced to learn the best $\tilde{\omega}_j$ from the dataset. We rewrite Eq. 10 in a matrix form as follows:

$$\begin{aligned} \max \quad & \sum_{j=1}^q \tilde{\omega}_j \Phi \mathbf{M}^{(j)} \Phi^T = \Phi \left(\sum_{j=1}^q \tilde{\omega}_j \mathbf{M}^{(j)} \right) \Phi^T \\ \text{s.t.} \quad & \|\Phi\| = 1 \end{aligned} \quad (11)$$

Hence, the optimal Φ is the set of eigenvectors of the top d eigenvalues of the matrix sum $\sum_{j=1}^q \tilde{\omega}_j \mathbf{M}^{(j)}$.

3.3 A General CoCA Algorithm

Sections 3.1 and 3.2 introduce two special cases of the CoCA principle. In this section, we consider the general scenario where we have a mixture of vector-based features and relational features. The objective function contains three components as discussed in Eq. 1. The first term maximizes the variance of the projected data. The second term is to ensure a common feature subspace as in Sect. 3.1. The last term is to maximize the similarity of the data with connections as in Sect. 3.2. Hence, the objective function can be written as follows:

$$\begin{aligned} \max_{\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(p)}} \quad & \sum_{j=1}^p \omega_j \sum_{i=1}^m \left(\mathbf{x}_i^{(j)T} \mathbf{u}^{(j)} \right)^2 \\ & + \alpha \sum_{j=1}^p \sum_{h=1}^p \sum_{i=1}^m \left(\mathbf{x}_i^{(j)T} \mathbf{u}^{(j)} \right) \left(\mathbf{x}_i^{(h)T} \mathbf{u}^{(h)} \right) \\ & + \sum_{j=1}^p \sum_a \sum_b \left(\sum_{t=1}^q \tilde{\omega}_t \mathbf{M}^{(t)}(a, b) \cdot \left(\mathbf{x}_a^{(j)T} \mathbf{u}^{(j)} \right)^T \cdot \mathbf{x}_b^{(j)T} \mathbf{u}^{(j)} \right) \\ \text{s.t.} \quad & \|\mathbf{u}^{(j)}\| = 1 \quad \text{where } j = 1, \dots, p \end{aligned} \quad (12)$$

where the first term is to maximize the variance, and the third term is to maximize the similarity of connected data, and the second term is to ensure the multiple feature spaces to be projected to the same feature subspace. In a matrix form, it can be written as:

$$\begin{aligned} \max_{\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(p)}} \quad & \mathbf{u}^T \mathbf{X} \mathbf{u} + \mathbf{u}^T \tilde{\mathbf{X}} \mathbf{u} \\ \text{s.t.} \quad & \|\mathbf{u}^{(j)}\| = 1 \quad \text{where } j = 1, \dots, p \end{aligned} \quad (13)$$

where $\mathbf{X}$ is as defined in Eq. 8 of Sect. 3.1 and

$$\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{Z}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{Z}_2 & \dots & \mathbf{0} \\ \dots & \dots & \dots & \dots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{Z}_p \end{bmatrix} \quad (14)$$

where

$$\mathbf{Z}_i = \mathbf{X}^{(i)} \left(\sum_t^q \tilde{\omega}_t \mathbf{M}^{(t)} \right) \mathbf{X}^{(i)T}$$

Hence, the optimal solution is given by the eigenvectors of the top d eigenvalues of the matrix $\mathbf{X} + \hat{\mathbf{X}}$. The output is the expected value of the projections $\Phi = \frac{1}{p} \sum_{j=1}^p (\mathbf{X}^{(j)} + \hat{\mathbf{X}}^{(j)})^T \mathbf{u}^{(j)}$.

It is important to note that Eq. 12 is a general solution, and it includes Eq. 7 in Sect. 3.1 and Eq. 10 in Sect. 3.2 as special cases. In the experiment, we use Eq. 12 to generate the reduced feature spaces since it can handle all three cases: (1) learning with only vector-based features, (2) learning with only relational features, and (3) learning with both vector-based features and relational features.

3.4 Weight Learning

In Eq. 12, we set the weights $\omega_j = \frac{1}{p}$ and $\tilde{\omega}_t = \frac{1}{q}$ where $1 \leq j \leq p$ and $1 \leq t \leq q$. In other words, all sources providing vector-based features have the same weight; all sources with relational features have the equal weight. However, in some cases, some of the sources may be more informative than the others. In this situation, a better strategy is to increase the weights of the informative sources, in order to obtain a more informative feature subspace. In the following, we study how to obtain the optimal weights with the “must-link” and “cannot-link” constraints as in semi-supervised learning [16]. We first introduce how to learn optimal weights for data sources with vector-based features. The general idea is to first use PCA to obtain projections $\Phi^{(k)} = \mathbf{X}^{(k)T} \mathbf{u}^{(k)}$ individually for each source where $\mathbf{u}^{(k)}$ is given by PCA, and then find which feature subspaces $\Phi^{(k)}$ can better satisfy the constraints. To do so, we first denote a $m \times m$ constraint matrix $\mathbf{C}$ as:

$$\mathbf{C}(i, j) = \begin{cases} 1 & \text{if the } i\text{-th and } j\text{-th data should have} \\ & \text{similar projected values} \\ 0 & \text{no preference} \\ -1 & \text{if the } i\text{-th and } j\text{-th data should have} \\ & \text{dissimilar projected values} \end{cases} \quad (15)$$

In other words, $\mathbf{C}(i, j) = 1$ if it is a “must-link” constraint; $\mathbf{C}(i, j) = -1$ if it is a “cannot-link” constraint; otherwise it is 0. We further define a *truncated similarity matrix* $\mathbf{E}^{(k)}$ for the k -th source such that

$$\mathbf{E}^{(k)}(i, j) = |\mathbf{C}(i, j)| \times \left(\Phi^{(k)}(i) \Phi^{(k)}(j)^T \right) \quad (16)$$

where $|*|$ takes the absolute value, and hence $|\mathbf{C}(i, j)| = 1$ iff the i -th data and the j -th data have certain constraint, regardless it is a “must-link” or a “cannot-link”. Furthermore, $\Phi^{(k)} = \mathbf{X}^{(k)T} \mathbf{u}^{(k)}$ is the reduced subspace from the k -th source given by

PCA. Hence, if the i -th data and the j -th data do not have any constraint, $\mathbf{E}^{(k)}(i, j) = 0$; otherwise

$$\mathbf{E}^{(k)}(i, j) = (\Phi^{(k)}(i))(\Phi^{(k)}(j))^T \quad (17)$$

We call $\mathbf{E}^{(k)}$ “truncated similarity matrix” since it only shows the similarity of the data with constraints. Moreover, since $\Phi^{(k)}$ has zero mean as in PCA, $\mathbf{E}^{(k)}(i, j) > 0$ iff the i -th and the j -th data have similar projected values of the same sign, and $\mathbf{E}^{(k)}(i, j) < 0$ iff they are with opposite signs. The ideal truncated similarity matrix should have the same values as the constraint matrix. Hence, given the truncated similarity matrices $\mathbf{E}^{(k)}$ from all sources where $k = 1, 2, \dots$, the objective function is to find a linear combination strategy that best satisfies the constraint matrix $\mathbf{C}$:

$$\begin{aligned} \min_{\omega_1, \omega_2, \dots} \quad & \left\| \left(\sum_{k=1}^p \omega_k \mathbf{E}^{(k)} \right) - \mathbf{C} \right\|_F^2 \\ \text{s.t.} \quad & \sum_{k=1}^p \omega_k = 1, \omega_k \geq 0 \text{ for } k = 1, \dots, p \end{aligned} \quad (18)$$

where $\| * \|_F^2$ is the Frobenius norm. The sources with better truncated similarity matrices are learned to have higher weights, and we use these weights in the general CoCA algorithm Eq. 12. We next solve Eq. 18.

Theorem 1 *The optimization in Eq. 18 is equivalent to the following quadratic programming problem:*

$$\begin{aligned} \min_{\omega} \quad & \omega^T (\mathbf{K}) \omega - 2\mathbf{v}^T \omega \\ \text{s.t.} \quad & \sum_{k=1}^p \omega_k = 1, \omega_k \geq 0 \text{ for } k = 1, \dots, p \end{aligned} \quad (19)$$

where $\mathbf{K}$ is a $p \times p$ matrix, and $\mathbf{v}$ is a $p \times 1$ vector such that

$$\begin{aligned} \mathbf{K}(i, j) &= \text{tr} \left((\mathbf{E}^{(i)})^T \mathbf{E}^{(j)} \right) \\ \mathbf{v}(i) &= \text{tr}(\mathbf{C} \mathbf{E}^{(i)}) \end{aligned}$$

Proof The optimization function in Eq. 18 can be written as

$$\begin{aligned} & \left\| \left(\sum_{k=1}^p \omega_k \mathbf{E}^{(k)} \right) - \mathbf{C} \right\|_F^2 \\ &= \text{tr} \left(\left(\sum_{k=1}^p \omega_k \mathbf{E}^{(k)} \right)^T \left(\sum_{k=1}^p \omega_k \mathbf{E}^{(k)} \right) \right) \end{aligned}$$

Algorithm 1: A general algorithm with the CoCA principle

Input: p sources with vector-based features: $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \dots, \mathbf{X}^{(p)}$, q sources with relational features: $\mathbf{M}^{(1)}, \mathbf{M}^{(2)}, \dots, \mathbf{M}^{(q)}$ where either p or q can be zero, the parameter α and the desired dimension d , and the constraint matrix $\mathbf{C}$ (optional).

Output: The dataset with reduced dimension Φ .

```

1 if the constraint matrix  $\mathbf{C}$  is available then
2   | Learn  $\omega$ 's and  $\tilde{\omega}$ 's as in Eq. 19;
3 else
4   | if  $p = 0$  then
5     |   Set  $\omega$ 's to be 0 and  $\tilde{\omega}$ 's to be  $\frac{1}{q}$ ;
6   | else
7     |   if  $q = 0$  then
8       |     Set  $\omega$ 's to be  $\frac{1}{p}$  and  $\tilde{\omega}$ 's to be 0;
9     |   else
10      |     Set  $\omega$ 's to be  $\frac{1}{p}$  and  $\tilde{\omega}$ 's to be  $\frac{1}{q}$ ;
11      |   end
12   | end
13 end
14 Construct the matrix  $\mathbf{X}$  as in Eq. 8;
15 Construct the matrix  $\tilde{\mathbf{X}}$  as in Eq. 14;
16 Find the eigenvectors  $\mathbf{u}$  of the top  $d$  eigenvalues of the matrix  $\mathbf{X} + \tilde{\mathbf{X}}$ ;
17 Construct  $\Phi$  with  $\mathbf{u}$  as in Eq. 2.

```

$$\begin{aligned}
 & -2\mathbf{C}^T \left(\sum_{k=1}^p \omega_k \mathbf{E}^{(k)} \right) + \mathbf{C}^T \mathbf{C} \\
 & = \omega^T (\mathbf{K}) \omega - 2\mathbf{v}^T \omega + \text{tr}(\mathbf{C}^T \mathbf{C})
 \end{aligned}$$

Note that $\text{tr}(\mathbf{C}^T \mathbf{C})$ is a constant. Hence, the optimization in Eq. 18 is equivalent to that in Eq. 19.

The quadratic programming problem in Eq. 19 can be conveniently solved with standard solvers such as MATLAB. Note that if the given heterogeneous features are relational features $\mathbf{M}^{(1)}, \mathbf{M}^{(2)}, \dots$, we can replace $\Phi^{(k)} \Phi^{(k)T}$ by $\mathbf{M}^{(k)}$ in Eq. 17 with some modifications on $\mathbf{M}^{(i)}$. We have to set $\mathbf{M}^{(k)}(i, j) = -1$ if the i -th and the j -th data is disconnected. This modification is to reflect the “cannot-link” constraints when forming $\mathbf{E}^{(k)}$. All the formulas and properties still hold for relational features.

3.5 Algorithm Flow and Complexity Analysis

The CoCA approach is described in Algorithm 1. It takes the data from various sources as input, and its aim is to construct an embedding with reduced dimensions. The algorithm has an optional input, which is the constraint matrix $\mathbf{C}$. If $\mathbf{C}$ is available, the CoCA algorithm can take advantage of $\mathbf{C}$ to learn the optimal weights (ω 's) as in Eq. 19; otherwise, CoCA assigns equal weights to the sources to learn the target feature subspace. It is important to mention that although the CoCA algorithm applies quadratic programming to obtain the optimal weights, this step can actually be finished

in constant time. This is because the complexity of the quadratic programming solver (e.g., MATLAB) is $O(u^3)$ where u is the number of unknown parameters, which is equal to the number of sources. In practice, the number of sources is usually not large, and it can be viewed as a constant. Hence, the quadratic programming step can take constant time to finish. The most expensive step of the algorithm is to get the eigenvectors. The complexity of the whole algorithm is $O(dIN)$ where d is the number of eigenvectors desired, I is the number of Lanczos iteration steps, and N is the non-zero entries of the matrix. In the worst case, $N = (\sum_j d_j)^2$ as $\mathbf{X} + \hat{\mathbf{X}}$ in Eq. 12 where $\sum_j d_j$ is the total dimension of all feature spaces.

4 Experiments

In this section, we analyze the proposed model Collective Component Analysis (CoCA) on three sets of datasets with heterogeneous features.

4.1 Comparison Approaches

Since there was no previous model that can be directly used to handle the same problem, we compared CoCA with a straightforward strategy. The comparison strategy is to directly join all features together. In other words, given the sources with vector-based features $\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(p)}$ and sources with relational features $\mathbf{M}^{(1)}, \dots, \mathbf{M}^{(q)}$, the joined features can be represented as follows:

$$\mathbf{X} = [\mathbf{X}^{(1)T}, \dots, \mathbf{X}^{(p)T}, \mathbf{M}^{(1)T}, \dots, \mathbf{M}^{(q)T}]^T \quad (20)$$

Hence, traditional dimensionality reduction methods can be applied on the joined features to obtain a reduced space. Note that CoCA is a general principle to handle heterogeneous features. The principle can be incorporated by most of the unsupervised dimensionality reduction models by adding the projection consensus and graph constraints. In this paper, we take PCA as an example. In other words, the proposed CoCA model adopts PCA's strategy to maximize the variance of the data but with the CoCA principle, and PCA was used as baseline that run on the joined dataset (Eq. 20).

4.2 Evaluation Strategy

Note that the output of CoCA is the dataset with a reduced feature space. In order to evaluate the quality of the reduced feature space, we performed clustering on the output, and the clustering result was evaluated by normalized mutual information (NMI). Note that NMI equals to zero when clustering is random, and it is close to one when the clustering result is good. Furthermore, k-means was chosen as the based clustering algorithm since it is efficient and widely used in practice. Note that k-means is sensitive to initial seed selection. Hence, we run k-means ten times on each parameter setting, and report the summarized NMI with mean value and standard deviation. It is also important to note that we propose two versions of CoCA, which are different in how to obtain the weights of the sources. In other words, we have two versions

Fig. 2 Handwritten numbers recognition

of CoCA depending on whether the optional constraint matrix $\mathbf{C}$ in Algorithm 1 is available or not. If the constraint matrix $\mathbf{C}$ is not available, we directly assign equal weights to the sources; otherwise we learn the optimal weights as in Sect. 3.4. In the experiment, we denote the CoCA without the constraint matrix $\mathbf{C}$ as “CoCA (equal weights)” and the one with $\mathbf{C}$ as “CoCA (unequal weights)”. To construct constraint matrix $\mathbf{C}$, we sampled 30 % data and generated “must-links” among the data with the same class label, and “cannot-links” among the data with different labels. We report the results with increasing number of desired dimensions.

4.3 Handwritten Dutch Numbers Recognition

The first dataset contains 2,000 handwritten numerals (“0”–“9”) extracted from a collection of Dutch utility maps [7]. The handwritten numbers are scanned and digitized as binary images. They are represented in terms of the following six feature spaces (six sources) with different vector-based features: (1) 76 Fourier coefficients of the character shapes, (2) 216 profile correlations, (3) 64 Karhunen-Love coefficients, (4) 240 pixel averages in 2×3 windows, (5) 47 Zernike moments, and (6) 6 morphological features. All these features are conventional vector-based features but in different feature spaces. The aim is to find the optimal embedding with reduced dimensions collectively with the six different sources. As mentioned before, we use two versions of CoCA to find the reduced feature spaces. One is to assign equal weights to the sources, and the other is to learn the optimal weights from prior knowledge. The results are presented in Fig. 2. As it can be observed, CoCA outperforms the comparison method substantially. For example, when the dimension is around 8, the NMI obtained from the two versions of CoCA are around 0.65 while that of the comparison method is only around 0.45. This shows that CoCA performs better than the intuitive strategy which directly uses the joined features. Between the two versions of CoCA, the unequal-weight solution is noticeably better, especially when the dimension is low. This is because some prior knowledge (“must-links” and “cannot-links”) is used to emphasize those informative sources and reduce noise.

4.4 Citeseer Document Dataset

In this experiment, we studied high-dimensional text data. The dataset [3] is originally from Citeseer with 3312 scientific papers under six different topics: Agents, AI, DB,

Fig. 3 Citeseer dataset

IR, ML, and HCI. It includes two types of features. One is the vector-based features that describe the papers with “bag of words” representation. In other words, each feature represents one word and the feature value reflects the presence or absence of the word in the document. The relational features come from the citations that link papers together if they have citation relations, which are converted into an undirected graph. Note that in order to examine the weight learning algorithm described in Sect. 3.4, we split the vector-based “bag of words” representation into five different subspaces. This was done by first calculating the mutual information of each feature (word) with respect to the class labels (topics), and then sorting them in descending order. We then split the sorted features into five groups where the first group contains the top 20 % of features with the largest mutual information, and the second group is the following 20 % of features, and so on. Furthermore, we randomly generated additional sets of vector-based features and relational features by generating matrices with binary entry values (0 or 1) randomly. As a result, we have eight sources which contain six sets of vector-based features and two sets of relational features. Note that some of the sources are highly correlated with the class label (informative), while others are very noisy. The weight learning algorithm discussed in Sect. 3.4 is supposed to be able to identify the informative sources.

The results summarized in ten runs are presented in Fig. 3 with the changing value of dimensions. Since we were dealing with a document dataset that has over 3,000 features, we thus extended the maximal dimension up to 100. As it can be observed, the CoCA with unequal weights has the best performance while the CoCA with equal weights ranks the second. Both approaches beat the comparison modified PCA substantially and significantly. It shows that the naïve strategy is far from enough to build an accurate learner. Instead, CoCA can collectively take different sources into consideration, and obtain a better reduced feature space. Furthermore, some prior knowledge can help CoCA get better weights of the sources, and the results are further improved.

4.5 Terrorist Attack Detection

The third dataset is about terrorist attack [3] that consists of 1,293 different attacks in one of the six labels indicating the type of the attack: arson, bombing, kidnapping, NBCR attack, weapon attack and other attack. Each attack is described by a binary value vector of attributes whose entries indicate the absence or presence of a feature. There is a total of 106 distinct vector-based features, along with two sets of relational

Fig. 4 Terrorist attack detection dataset

features. One set connects the attacks from the same location, and the other connects the attacks if they are planned by the same organization. As in the previous experiment, we split the vector-based features into five different groups, and randomly generated one additional set of vector-based features and one additional set of relational features. In summary, this dataset contains features from nine sources including six sets of vector-based features and three sets of relational features.

The results with the changing value of dimensions are presented in Fig. 4. It is clear that the equal-weight CoCA beats the modified PCA by as much as 50 %, and the weighted CoCA can beat the comparison model by as much as three times in NMI. Furthermore, when the number of features reaches 15, the proposed model gets the best performance. Note that selecting the optimal dimension is a nontrivial model selection problem, but it is beyond the scope of the present paper. In practice, one can apply cross-validation or validation set to obtain the best dimension.

4.6 Discussion

In this section, we aim at analyzing CoCA more in detail in order to answer the following three questions:

1. How does the embedding look like geometrically?
2. Can the embedding data with reduced dimensions work well on other learners (such as different classifiers in classification problem)?
3. How does the parameter affect the algorithm?

Figure 5 plots a subset of the hand-written number data in certain feature spaces. The black squares represent the data belonging to one class, while the red crosses are the data of another class. Specifically, Fig. 5a–f are the feature spaces from the six sources, and the shown dimensions are the ones that are most correlated with the class labels. In other words, in each source, the two best features (evaluated by mutual information) were chosen to show in Fig. 5. It can be clearly observed that some feature spaces are relatively informative (e.g., Fig. 5a, c help separate part of the data), while some are quite noisy (e.g., Fig. 5e). However, the final reduced feature space in Fig. 5g best captures the characteristics of the dataset by taking all sources into consideration.

Note that in the experiment section, we evaluated the embedding with reduced dimensions via clustering algorithm k-means. It is also interesting to see how the low-

Fig. 5 Samples of the projected data: squares are the data of the same class, while *multiplication signs* are the data of another class

Fig. 6 Classification capability and parameter sensitivity analysis

dimensional data performs on classification problems. Hence, we used the 10-classes hand-written number recognition dataset as an example to analyze the classification capability of the low-dimensional data. To isolate the effect of prior knowledge, we considered the equal weight CoCA. The modified PCA was again used as a comparison method. Several classifiers were trained on the dataset with reduced dimensions and we compared their average accuracy with 10-fold cross validation. The results are shown in Fig. 6a with six different classifiers SVM, C4.5, Naive Bayes, Nearest Neighbor, Logistic Regression and the semi-supervised SVM. All six experiments show that CoCA better captures the characteristics of the data and obtains a better classification accuracy. For example, when the accuracy given by the modified PCA is only 55 % with SVM, the accuracy of CoCA reaches 65 %, which is 18 % better.

In the proposed algorithm, we have one parameter α that controls how much we want the different feature spaces to be projected onto the same subspace. We use the terrorist dataset as an example and plot the results with different values of α as shown in Fig. 6b. It can be observed that when α is small, the result does not look good because the heterogeneous feature spaces are not bounded to find a consensus subspace. When α is around 60, the result tends to be stable, and we choose α to be 60 in the experiment. However, when α is larger than 75, the performance drops a little because the learning model encounters overfitting by forcing all feature spaces (including the noisy ones) “vote” for the consensus subspace. In practice, one can use cross-validation or directly use the root mean square error (the objective function of k-means) as a criterion to choose α . In the experiment, we chose $\alpha = 60$.

5 Related Work

There are several areas of related works that we built upon. First, multi-view learning (e.g., [2, 8, 11, 12]) is proposed to learn from instances which have multiple views in different feature spaces. For example, in [11], a framework is proposed to reconcile the clustering results from different views. In [5], a term called consensus learning is proposed. The general idea is to perform learning on each heterogeneous feature space independently and then ensemble the results. Furthermore, [1] proposes an approach to combine different data sources for recommendation systems. There are mainly two differences between our work with these previous approaches. First, so far as we know,

most of the previous works do not consider vector-based features and relational features simultaneously. Second, most of these models aim at tackling classification or clustering problem. In this paper, we instead aim at solving low-dimensional embedding especially for heterogeneous learning, to compress the features from multiple sources.

Another area of related work is collective classification (e.g., [13]) that aims at predicting the class label from a network. Its key idea is to combine the supervision knowledge from traditional vector-based feature vectors, as well as the linkage information from the network. It has been applied to various applications such as part-of-speech tagging [10], classification of hypertext documents using hyperlinks [14], etc. Most of these works study the case when there is only one vector-based feature space and only one relational feature space, and the focus is how to combine the two. Different from traditional collective classification framework, we consider multiple vector-based features and multiple relational features simultaneously. Furthermore, we solve embedding problem instead of classification problem since “the curse of dimension” is more likely to happen in heterogeneous learning (increasing number of features from multiple sources).

The proposed model is also related to dimensionality reduction and kernel learning (e.g., [15]). One of the surveys can be found at [4]. There are two differences of the current work with previous works. First, most of the previous works do not consider vector-based features and graph relational features simultaneously. Second, most of approaches treat all the data sources the same, while the proposed CoCA has a schema to identify informative data source from the noisy ones. Robustness is a very important property of CoCA since the multiple data sources may contain substantial unobserved noise.

Specifically, the work in [9] also attacks dimensionality reduction on “heterogeneous” data. However, it is totally different from the present work. The objective of [9] is to design an implementation for PCA in the distributed computing environment and the term “heterogeneous data” refers to the data stored in different computers. The present work instead aims at developing an algorithm on a new learning schema called heterogeneous learning where “heterogeneous” refers to the different feature spaces of the objects, and they can have different physical meanings.

6 Conclusion

In this paper, we study the problem of finding the optimal embedding given multiple sets of vector-based features and multiple sets of graph relational features. We propose a CoCA principle to solve the problem by finding a reduced feature subspace that maximizes the variance of the data, with two constraints. First, all the heterogeneous feature spaces have to be consensually projected on the same subspace. Second, if relational features are available, the data with connections are preferred to be similar in the projected space. An optimization problem is derived from the CoCA principle, and the solution is obtained by solving an eigenvalue problem. Furthermore, we extend the method by using prior knowledge to identify better data sources and construct informative feature subspaces. Three sets of experiments were performed to evaluate

CoCA. It can be clearly observed that the proposed CoCA model outperforms the comparison algorithm by as much as 50 %, and the weighted CoCA beats the comparison model even more (by as much as three times in NMI). We also analyze the performance of CoCA on classification problems in the discussion section, where we can observe an 18 % gain over comparison model.

Acknowledgments This work is supported in part by NSF through grants IIS-0905215, CNS-1115234, IIS-0914934, DBI-0960443, and OISE-1129076, US Department of Army through grant W911NF-12-1-0066, Google Mobile 2014 Program and KAU grant.

References

1. Agarwal D, Chen B, Long B (2011) Localized factor models for multi-context recommendation. KDD, p 609–617
2. Blum A, Mitchell TM (1998) Combining labeled and unlabeled data with co-training. COLT, p 92–100
3. Collective-Classification-Dataset. <http://www.cs.umd.edu/projects/linqs/projects/lbc/index.html>. Accessed 26 Jan 2012
4. Dash M, Liu H (2008) Dimensionality reduction. Wiley encyclopedia of computer science and engineering. Wiley, Hoboken
5. Gao J, Fan W, Sun Y, Han J (2009) Heterogeneous source consensus learning via decision propagation and negotiation. KDD, p 339–348
6. Golub G, Loan CV (1996) Matrix computation. The Johns Hopkins University Press, Baltimore
7. Handwritten-Numerals-Dataset. <http://archive.ics.uci.edu/ml/datasets/multiple+features>. Accessed 26 Jan 2012
8. Hardoon David R, Szedmák Sándor, Shawe-Taylor John (2004) Canonical correlation analysis: an overview with application to learning methods. Neural Comput 16(12):2639–2664
9. Kargupta H, Huang W, Sivakumar K, Park B, Wang S (2000) Collective principal component analysis from distributed, heterogeneous data. PKDD, p 452–457
10. Lafferty JD, McCallum A, Pereira FCN (2001) Conditional random fields: probabilistic models for segmenting and labeling sequence data. Proceedings of the international conference on machine learning
11. Long B, Yu PS, Zhang Z (2008) A general model for multiple view unsupervised learning. SDM, p 822–833
12. Nigam K, Ghani R (2000) Analyzing the effectiveness and applicability of co-training. CIKM, p 86–93
13. Sen P, Namata G, Bilgic M, Getoor L, Gallagher B, Eliassi-Rad T (2008) Collective classification in network data. AI Mag 29(3):93–106
14. Taskar B, Abbeel P, Koller D (2002) Discriminative probabilistic models for relational data. Proceedings of the annual conference on uncertainty in artificial intelligence
15. Zhao Z, Liu H (2007) Spectral feature selection for supervised and unsupervised learning. ICML, p 1151–1157
16. Zhu X (2005) Semi-supervised learning with graphs. Technical report, Carnegie Mellon University

Xiaoxiao Shi is a quantitative researcher and portfolio manager in the financial industry. Dr. Shi earned his Ph.D. degree in Computer Science at the University of Illinois at Chicago, and his Master degree in Applied Math concurrently. Before joining the research team led by Prof. Philip Yu, Dr. Shi received the BEng degree from Computer Science Department of Sun Yat-sen University, China. His research interest is on big data, robust prediction and time series analysis. More details about his research and background can be found at <http://www.cs.uic.edu/xiaoxiao>.

Philip S. Yu is a Distinguished Professor in Computer Science at the University of Illinois at Chicago and also holds the Wexler Chair in Information Technology. Dr. Yu spent most of his career at IBM, where he was manager of the Software Tools and Techniques group at the Watson Research Center. His research interest is on big data, including data mining, data stream, database and privacy. He has published more than 800 papers in refereed journals and conferences. He holds or has applied for more than 300 US patents. Dr. Yu is a Fellow of the ACM and the IEEE. He is the Editor-in-Chief of ACM Transactions on Knowledge Discovery from Data. He is on the steering committee of the IEEE Conference on Data Mining and ACM Conference on Information and Knowledge Management and was a member of the IEEE Data Engineering steering committee. He was the Editor-in-Chief of IEEE Transactions on Knowledge and Data Engineering (2001–2004). He received the IEEE Computer Society 2013 Technical Achievement Award for “pioneering and fundamentally innovative

contributions to the scalable indexing, querying, searching, mining and anonymization of big data”, the ICDM 2013 10-year Highest-Impact Paper Award, the EDBT Test of Time Award (2014), and the Research Contributions Award from IEEE Intl. Conference on Data Mining (2003). He had received several IBM honors including 2 IBM Outstanding Innovation Awards, an Outstanding Technical Achievement Award, 2 Research Division Awards and the 94th plateau of Invention Achievement Awards. He was an IBM Master Inventor. Dr. Yu received the B.S. Degree in E.E. from National Taiwan University, the M.S. and Ph.D. degrees in E.E. from Stanford University, and the M.B.A. degree from New York University.