Information Sciences 512 (2020) 18–30

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Auto-weighted multi-view co-clustering with bipartite graphs

Shudong Huang[a], Zenglin Xu[a,*], Ivor W. Tsang[b], Zhao Kang[a]

[a] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
[b] Centre for Artificial Intelligence, University of Technology Sydney, NSW 2007, Australia

ARTICLE INFO

Article history:
Received 9 May 2019
Revised 16 September 2019
Accepted 29 September 2019
Available online 30 September 2019

Keywords:
Multi-view learning
Co-clustering
Bipartite graph learning
Auto-weighted strategy

ABSTRACT

Co-clustering aims to explore coherent patterns by simultaneously clustering samples and features of data. Several co-clustering methods have been proposed in the past decades. However, in real-world applications, datasets are often with multiple modalities or composed of multiple representations (i.e., views), which provide different yet complementary information. Hence, it is essential to develop multi-view co-clustering models to solve the multi-view application problems. In this paper, a novel multi-view co-clustering method based on bipartite graphs is proposed. To make use of the duality between samples and features of multi-view data, a bipartite graph for each view is constructed such that the co-occurring structure of data can be extracted. The key point of utilizing the bipartite graphs to deal with the multi-view co-clustering task is to reasonably integrate these bipartite graphs and obtain an optimal consensus one. As for this point, the proposed method can learn an optimal weight for each bipartite graph automatically without introducing an additive parameter as previous methods do. Furthermore, an efficient algorithm is proposed to optimize this model with theoretically guaranteed convergence. Extensive experimental results on both toy data and several benchmark datasets have demonstrated the effectiveness of the proposed model.

1. Introduction

In many real-world data analysis applications, both data samples and features exhibit some co-occurrence properties [1,2]. For example, in movie recommendation, users share similar interests and habits tend to provide the same or similar rating scores to the same type of movies. Such data analysis settings naturally lead to a variational type of clustering method — co-clustering, which can simultaneously cluster the samples and features. Many co-clustering methods have been studied via different theories and methodologies [3,4]. Different from standard clustering methods which discover similar rows or columns [5–8], co-clustering allows for better interpretability by seeking blocks (i.e., co-clusters) of inter-correlated rows and columns. By utilizing the relationship between two entities, co-clustering also has the advantage in predicting missing values [9]. Thanks to these nice properties, co-clustering has been successfully applied to a number of applications such as text mining [1,10], chemometrics [11], bioinformatics [3], recommendation systems [12], etc.

Nowadays, more and more datasets are collected from different sources or represented by different views. In addition, different views usually capture different aspects of information and can be complementary to each other. For example, images can be described by different visual descriptors [13–15]; documents can be written in different languages [16,17];

* Corresponding author.
E-mail address: zlxu@uestc.edu.cn (Z. Xu).

https://doi.org/10.1016/j.ins.2019.09.079
0020-0255/© 2019 Published by Elsevier Inc.

and in web page categorization, web pages can be grouped by page content or citation links [18,19]. In these situations, data are often represented by attributes that can naturally be split into different subsets (also known as views), each of which suffices for mining knowledge from data. Thus it is necessary to efficiently explore the complementary information from these heterogeneous features. This necessity leads to the learning paradigm – multi-view learning. As an important multi-view learning task in the unsupervised setting, multi-view clustering aims to integrate compatible and complementary information from the multiple views [20,21]. It considers the diversity of different views and fuses these views to produce a more accurate and robust result [22–24].

However, most existing multi-view clustering methods focus on one-side clustering, i.e., these methods ignore the co-occurrence of both samples and features. Considering the advantages of co-clustering models as mentioned before, it is desirable to design a multi-view co-clustering model to overcome the limitations associated with traditional multi-view clustering methods. In general, multi-view co-clustering has advantages over traditional multi-view clustering in several perspectives: (1) Multi-view co-clustering provides more informative clustering to simultaneously cluster samples and features of multi-view data. It is able to provide compressed representations that are easily interpretable while preserving most of the information contained in the original multi-view data. (2) Multi-view co-clustering can often yield better quality clusters. In co-clustering, clustering on columns can be regarded as a "statistical regularization" to clustering on rows technique, and vise versa. The statistical regularization effect of multi-view co-clustering is extremely important when dealing with large, sparse data [1]. (3) Multi-view co-clustering can provide a much more compact representation for subsequent learning task, as simultaneous clustering along both rows and columns reduces dimensionality of both axes. Since the number of row and column clusters is usually much smaller than the original number of rows and columns, multi-view co-clustering can lead to substantial reduction in the running time [25].

Besides, the weight of each view in multi-view methods plays an important role to the clustering performance. In many methods, the weights are often learned by introducing an additive hyperparameter, which requires search in a large range to find the optimal value. Thus it is desirable to automatic determine the value. In this paper, a novel method for multi-view co-clustering is proposed via reformulating the spectral learning model. To make use of the duality between samples and features of multi-view data, a bipartite graph for each view is constructed such that the co-occurrence structure can be extracted. The key point of utilizing the bipartite graphs to deal with the multi-view co-clustering task is to integrate these bipartite graphs reasonably and obtain an optimal consensus one. As for this point, our method can automatically allocate ideal weight for each bipartite graph without additional hyperparameter as previous methods do. Furthermore, an efficient algorithm is proposed to optimize this model and its convergence is also theoretically guaranteed. Extensive experimental results on both toy data and several benchmark datasets have demonstrated the effectiveness of the proposed method.

It is worthwhile to highlight our contributions in this paper as follows:

- We propose a novel method for multi-view co-clustering via the reformulation of the spectral learning model to integrate heterogeneous representations of data. We also derive an efficient updating algorithm to solve the optimization problem.
- A bipartite graph for each view is constructed such that the co-occurrence structure can be extracted. The bipartite graphs are reasonably integrated and the optimal weight for each bipartite graph is automatically learned without introducing additive hyperparameter as previous methods do. An auto-weighted strategy is utilized in our model to avoid extra efforts in searching the additive hyperparameter while preserving the good performance.
- To the best of our knowledge, this is the first work on utilizing bipartite graphs to deal with the multi-view co-clustering task. Extensive experiments on benchmark datasets have demonstrated that our method outperforms other related multi-view learning methods, suggesting the effectiveness of the proposed method.

Notations. In this paper, matrices are written as boldface uppercase letters. For a matrix $\mathbf{A}$, boldface lowercase letter $\mathbf{a}_i$ and lowercase letter a_{ij} denote the ith column and ijth entity of $\mathbf{A}$, respectively. Tr($\mathbf{A}$) denotes the trace of $\mathbf{A}$ and the Frobenius norm of $\mathbf{A}$ is represented as $\|\mathbf{A}\|_F$. $\mathbf{A} \geq 0$ means that all elements of $\mathbf{A}$ are equal to or larger than zero.

2. Related work

In this section, we will briefly review the basic form of the classical spectral co-clustering.

2.1. Spectral co-clustering with bipartite graph

Given a dataset $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$ with n_1 data points and n_2 features. Thus the corresponding bipartite graph can be defined as $\mathcal{G} = \{\mathcal{V}_r, \mathcal{V}_c, \mathbf{E}\}$, where $\mathcal{V}_r$ denotes the sample vertex set, $\mathcal{V}_c$ denotes the feature vertex set and $\mathbf{E}$ is edge weight matrix with e_{ij} denotes the edge weight between the nodes $v_i \in \mathcal{V}_r$ and $v_j \in \mathcal{V}_c$. In order to simultaneously partition the rows and columns of $\mathbf{X}$, we first consider $\mathbf{X}$ as the weight matrix of $\mathcal{G}$ (i.e., $\mathbf{E} = \mathbf{X}$), where the nodes in $\mathcal{V}_r$ are the n_1 rows of $\mathbf{X}$, the nodes in $\mathcal{V}_c$ are the n_2 columns of $\mathbf{X}$, and the weight to connect the nodes $v_i \in \mathcal{V}_r$ and $v_j \in \mathcal{V}_c$ is x_{ij}. Accordingly, the affinity matrix $\mathbf{A}$ of the bipartite graph is

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{X} \\ \mathbf{X}^T & \mathbf{0} \end{bmatrix}. \tag{1}$$

Then the Laplacian matrix $\mathbf{L}$ [1] of $\mathcal{G}$ can be written as

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \tag{2}$$

where

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_c \end{bmatrix}, \tag{3}$$

and its diagonal element $d_{ii} = \sum_j a_{ij}$.

Let $\mathbf{P} \in \mathbb{R}^{(n_1+n_2)\times k}$ denote the partition indicator matrix, and we rewrite $\mathbf{P}$ as the following block matrix

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_r \\ \mathbf{P}_c \end{bmatrix}, \tag{4}$$

where k is the number of classes, $\mathbf{P}_r \in \mathbb{R}^{n_1\times k}$ is the partition on sample vertex set $\mathcal{V}_r$ and $\mathbf{P}_c \in \mathbb{R}^{n_2\times k}$ is the partition on feature vertex set $\mathcal{V}_c$.

The normalized cut [26] on the bipartite graph can be expressed as

$$\min_{\mathbf{P}^T\mathbf{P}=\mathbf{I}} \mathrm{Tr}(\mathbf{P}^T\mathbf{L}\mathbf{P}). \tag{5}$$

According to [26,27], Eq. (5) can be solved by eigenvalues calculating.

3. Problem formulation

In this section, first we summarize the multiple bipartite graphs learning methods into two general forms. By analyzing the deficiencies of them, we then propose a novel auto-weighted multi-view co-clustering method. Finally, we speed up the proposed model with an effective strategy which is based on normalized Laplacian function.

3.1. Multi-view co-clustering with bipartite graphs

The formulations mentioned above are presented for solving the single view co-clustering problems. Since many real world datasets are collected from different sources or represented by different views, it is essential to extend the single view models to solve the multi-view application problems. Given a multi-view data $\{\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(m)}\}$, where m is the number of views. $\mathbf{X}^{(v)} = [\mathbf{x}_1^{(v)}, \dots, \mathbf{x}_n^{(v)}]^T \in \mathbb{R}^{n\times d^{(v)}}$ denotes the v-th view with n data points and $d^{(v)}$ features. For convenience of discussion, we assume $d^{(1)} = d^{(2)} \dots = d^{(m)} = d$. But the proposed model can also handle the case when $d^{(1)} \neq d^{(2)} \dots \neq d^{(m)}$ with a simple and yet effective strategy, which will be discussed in a later section. For each view, an individual Laplacian matrix is constructed, thus we have $\mathbf{L}^{(1)}, \dots, \mathbf{L}^{(m)} \in \mathbb{R}^{(n+d)\times(n+d)}$. A straightforward way to utilize these bipartite graphs is to stack up to a new one, i.e., simply obtain a fusion bipartite graph by calculating $\bar{\mathbf{L}} = \frac{1}{m}\sum_{v=1}^m \mathbf{L}^{(v)}$. However, this strategy is equivalent to allocating the equal weight to each bipartite graph. Obviously, it ignores the importance of different bipartite graphs and may suffer when an unreliable one is considered. A more reasonable manner is to linearly integrate these bipartite graphs with suitable weights $w^{(v)} (v = 1, \dots, m)$, and keep the smooth of weights distribution by introducing an extra hyperparameter λ. If the indicator matrix $\mathbf{P}$ is enforced to be unified one across all the views, this thought can be formulated as

$$\begin{aligned} &\min_{\mathbf{P}^T\mathbf{P}=\mathbf{I}, w^{(v)}} \sum_{v=1}^m \left(w^{(v)}\right)^{\lambda} \mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right) \\ &\qquad \text{s.t.} \sum_{v=1}^m w^{(v)} = 1, w^{(v)} \geq 0, \end{aligned} \tag{6}$$

where λ is the non-negative scalar to control the weights distribution, it could be regularization parameter in another model:

$$\begin{aligned} &\min_{\mathbf{P}^T\mathbf{P}=\mathbf{I}, w^{(v)}} \sum_{v=1}^m w^{(v)} \mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right) + \lambda\|\mathbf{w}\|_2^2 \\ &\qquad \text{s.t.} \sum_{v=1}^m w^{(v)} = 1, w^{(v)} \geq 0, \end{aligned} \tag{7}$$

where $\mathbf{w} = [w^{(1)}, \dots, w^{(m)}]$. It is noteworthy that, similar to Eq. (5), our model described in Eq. (6) (or Eq. (7)) is essentially a kind of discriminative model which aims to cluster multi-view data by directly optimizing an objective function.

3.2. Optimization algorithm for Eq. (6)

In the following, an efficient iterative updating algorithm is proposed to solve the optimization problem in Eq. (6). Specifically, we will optimize the objective with respect to one variable while fixing another variable. This procedure repeats until convergence.

(1) Updating $\mathbf{P}$ with fixed $w^{(v)}$: Optimizing Eq. (6) w.r.t. $\mathbf{P}$ is equivalent to optimizing

$$\min_{\mathbf{P}^T\mathbf{P}=\mathbf{I}} \mathrm{Tr}\left(\mathbf{P}^T\widehat{\mathbf{L}}\mathbf{P}\right), \tag{8}$$

where $\widehat{\mathbf{L}} = \sum_{v=1}^m \left(w^{(v)}\right)^\lambda \mathbf{L}^{(v)}$. As mentioned in Eq. (5), the optimal solution $\mathbf{P}$ is obtained by the k eigenvectors of $\widehat{\mathbf{L}}$ corresponding to the k smallest eigenvalues.

(2) Updating $w^{(v)}$ with fixed $\mathbf{P}$: Optimizing Eq. (6) w.r.t. $w^{(v)}$ is equivalent to optimizing

$$\begin{aligned} &\min_{w^{(v)}} \sum_{v=1}^m \left(w^{(v)}\right)^\lambda \mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right) \\ &\qquad \text{s.t.} \sum_{v=1}^m w^{(v)} = 1, w^{(v)} \geq 0, \end{aligned} \tag{9}$$

We can define the Lagrange function of Eq. (9) as

$$J = \sum_{v=1}^m \left(w^{(v)}\right)^\lambda \mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right) - \zeta\left(\sum_{v=1}^m w^{(v)} - 1\right), \tag{10}$$

where ζ is the Lagrange multiplier. Setting $\frac{\partial J}{\partial w^{(v)}} = 0$, we obtain

$$w^{(v)} = \left(\frac{\zeta}{\lambda \mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right)}\right)^{\frac{1}{\lambda-1}}. \tag{11}$$

Considering the constraint $\sum_{v=1}^m w^{(v)} = 1$, we have

$$w^{(v)} = \frac{\left(\lambda \mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right)\right)^{\frac{1}{1-\lambda}}}{\sum_{v=1}^m \left(\lambda \mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right)\right)^{\frac{1}{1-\lambda}}}. \tag{12}$$

The detailed algorithm to solve the objective in Eq. (6) is summarized in Algorithm 1.

Algorithm 1 The Algorithm for Solving Eq. (13).

```
Input: Data with $m$ views $\{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \dots, \mathbf{X}^{(m)}\}$, hyperparameter $\lambda$ and the number of classes $k$;
    Initialize the weight factor $w^{(v)} = \frac{1}{m}$ for each view;
    Construct the Laplacian matrix $\mathbf{L}^{(v)}$ for each view;
Output: Cluster label of each data point.
1: repeat
2:     Calculate $\widehat{\mathbf{L}} = \sum_{v=1}^m \left(w^{(v)}\right)^\lambda \mathbf{L}^{(v)}$
3:     Compute $\mathbf{P}$, which is formed by the $k$ eigenvectors of $\widehat{\mathbf{L}}$ corresponding to the $k$ smallest eigenvalues
4:     Update $w^{(v)}$ according to Eq. (12)
5: until converge
6: Treat each row of $\mathbf{P}$ as a new representation of each datapoint and compute the clustering labels by using $K$-meansalgorithm.
```

For unsupervised learning models, the fewer parameters to be tuned, the more robustness they possess. Obviously, γ can be searched in a large range. Since the choice of γ is crucial to the final clustering performance and its ideal value varies for different datasets, it is really elusive to pursue good performance while rely less on parameter searching. In the next section, we will propose an auto-weighted strategy to alleviate such a challenging problem. Our ultimate goal is to remove the additive hyperparameter while preserving the good performance.

3.3. Auto-weighted multi-view co-clustering with bipartite graphs

In this paper, a novel auto-weighted multi-view co-clustering with bipartite graphs is proposed. We define the objective function of our model as follows

$$\min_{\mathbf{P}^T\mathbf{P}=\mathbf{I}} \sum_{v=1}^m w^{(v)} \mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right). \tag{13}$$

It can be observed that no weight hyperparameter is defined in Eq. (13). And $w^{(v)}$ can be calculated automatically according to the following theorem.

Theorem 1. *The weights* $w^{(v)}$ *can be determined as* $w^{(v)} = \frac{1}{2\sqrt{\mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right)}}$.

Proof. See Appendix A. □

Since $w^{(v)}$ are dependent on the target variable $\mathbf{P}$, Eq. (A.3) cannot be solved directly. But if we set $w^{(v)}$ to be stationary, then Eq. (A.3) can be treated as the solution to Eq. (13). In detail, based on the assumption that the weights $w^{(v)}$ are stationary, the Lagrange function of Eq. (A.1) can also be applied to Eq. (13). If we calculate $\mathbf{P}$ from Eq. (13), the values of $w^{(v)}$ can be further updated correspondingly. This inspires us to optimize Eq. (A.1) using an iterative way. Furthermore, if the iterative strategy converges (it will be discussed in a later section), the converged value of $\mathbf{P}$ is the local optimal solution to Eq. (A.1) according to Eqs. (A.3) and (A.4). Similarly, $w^{(v)}$ can be tuned to the optimal values correspondingly, and they are exactly the learned weights for all views. Note that in Eq. (13), it is exactly the linear combination of different views using the learned weights $w^{(v)}$. Compared with Eqs. (6) and (7), Eq. (13) is the real problem we want to solve and this is the core why we chose to solve such a problem like Eq. (A.1).

According to Eq. (A.4), if the vth view is good, then $\mathrm{Tr}(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P})$ should be small, thus the learned weight $w^{(v)}$ is large. Accordingly, a small weight will be assigned to a weak view. That is to say, the proposed auto-weighted strategy for our multi-view co-clustering model is reasonable and meaningful. In fact, the re-weighted optimization strategy described above can be considered as a special case of the *Iteratively Re-weighted* (IR) technique [28]. It is interesting that the effectiveness of the proposed method will reflect the practical significance of the weight formula in IR.

In the following, an efficient iterative updating algorithm is proposed to solve the optimization problem in Eq. (13). Specifically, we will optimize the objective with respect to one variable while fixing another variable. This procedure repeats until convergence.

(1) Updating $\mathbf{P}$ with fixed $w^{(v)}$: Optimizing Eq. (13) w.r.t. $\mathbf{P}$ is equivalent to optimizing

$$\min_{\mathbf{P}^T\mathbf{P}=\mathbf{I}} \mathrm{Tr}\left(\mathbf{P}^T\widetilde{\mathbf{L}}\mathbf{P}\right), \tag{14}$$

where $\widetilde{\mathbf{L}} = \sum_{v=1}^{m} w^{(v)}\mathbf{L}^{(v)}$. Similar to Eq. (8), the optimal solution $\mathbf{P}$ is obtained by the k eigenvectors of $\widetilde{\mathbf{L}}$ corresponding to the k smallest eigenvalues.

(2) Updating $w^{(v)}$ with fixed $\mathbf{P}$: $w^{(v)}$ can be updated by using Eq. (A.4).

The detailed algorithm to solve the objective in Eq. (13) is summarized in Algorithm 2.

Algorithm 2 The Algorithm for Solving Eq. (13).

```
Input: Data with $m$ views $\{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(m)}\}$ and the number of classes $k$;
   Initialize the weight factor $w^{(v)} = \frac{1}{m}$ for each view;
   Construct the Laplacian matrix $\mathbf{L}^{(v)}$ for each view;
Output: Cluster label of each data point.
1: repeat
2:    Calculate $\widetilde{\mathbf{L}} = \sum_{v=1}^{m} w^{(v)}\mathbf{L}^{(v)}$
3:    Compute $\mathbf{P}$, which is formed by the $k$ eigenvectors of $\widetilde{\mathbf{L}}$ corresponding to the $k$ smallest eigenvalues
4:    Update $w^{(v)}$ according to Eq. (A.4)
5: until converge
6: Treat each row of $\mathbf{P}$ as a new representation of each datapoint and compute the clustering labels by using $K$-
   meansalgorithm.
```

3.4. Convergence analysis

In this section, we prove the convergence of the Algorithm 2. We will show that our algorithm can find a local optimal solution. Before we prove its convergence, first we introduce an important lemma as follows [29]:

Lemma 1. *For any positive real number q and t, the following inequality holds:*

$$\sqrt{q} - \frac{q}{2\sqrt{t}} \le \sqrt{t} - \frac{t}{2\sqrt{t}}. \tag{15}$$

Proof. It is obvious that inequality $(\sqrt{q} - \sqrt{t})^2 \ge 0$, thus we have

$(\sqrt{q} - \sqrt{t})^2 \ge 0 \Rightarrow q - 2\sqrt{qt} + t \ge 0 \Rightarrow \sqrt{q} - \frac{q}{2\sqrt{t}} \le \frac{\sqrt{t}}{2} \Rightarrow \sqrt{q} - \frac{q}{2\sqrt{t}} \le \sqrt{t} - \frac{t}{2\sqrt{t}}$ which completes the proof. □

Theorem 2. *In Algorithm 2, updated* $\mathbf{P}$ *will monotonically decrease the objective in Eq. (13), which generally makes the solution converge to the local optimum of Eq. (13).*

Proof. See Appendix B. □

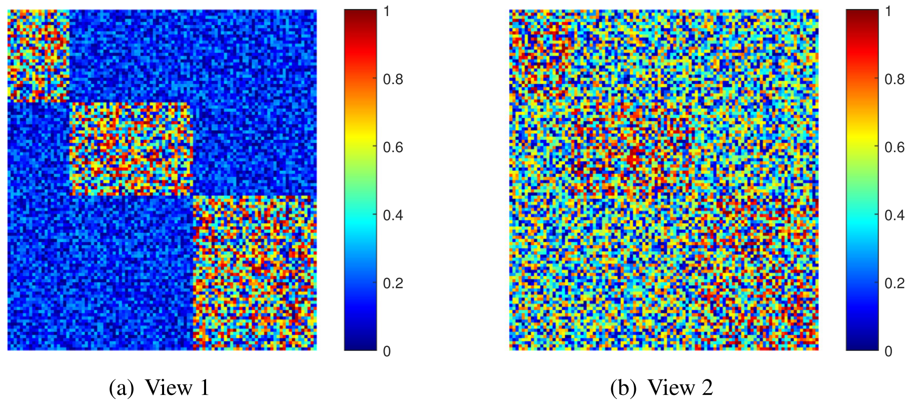

Fig. 1. Illustrations on the toy dataset with two views.

3.5. Speeding up our model

Similar to the definition of Eq. (1), we denote the affinity matrix of the vth view as $\mathbf{A}^{(v)}$ and

$$\mathbf{A}^{(v)} = \begin{bmatrix} \mathbf{0} & \mathbf{X}^{(v)} \\ \mathbf{X}^{(v)T} & \mathbf{0} \end{bmatrix}. \tag{16}$$

Thus the normalized Laplacian matrix of $\mathbf{A}^{(v)}$ can be defined as

$$\widetilde{\mathbf{L}}^{(v)} = \mathbf{I} - \mathbf{D}^{(v)^{-\frac{1}{2}}}\mathbf{A}^{(v)}\mathbf{D}^{(v)^{-\frac{1}{2}}}, \tag{17}$$

where $\mathbf{D}^{(v)}$ has the same definition as mentioned in Eq. (3) and $\mathbf{D}^{(v)}$ can be written as the following block matrix

$$\mathbf{D}^{(v)} = \begin{bmatrix} \mathbf{D}_r^{(v)} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_c^{(v)} \end{bmatrix}. \tag{18}$$

Based on Eqs. (17) and (13) can be rewritten as

$$\max_{\mathbf{P}^T\mathbf{P}=\mathbf{I}} \sum_v w^{(v)}\text{Tr}\left(\mathbf{P}^T\mathbf{D}^{(v)^{-\frac{1}{2}}}\mathbf{A}^{(v)}\mathbf{D}^{(v)^{-\frac{1}{2}}}\mathbf{P}\right). \tag{19}$$

Then according to the definition of $\mathbf{A}^{(v)}$, $\mathbf{D}^{(v)}$ and $\mathbf{P}$ in Eqs. (16), (18) and (4), respectively, Eq. (19) can be further rewritten as

$$\max_{\mathbf{P}_r^T\mathbf{P}_r+\mathbf{P}_c^T\mathbf{P}_c=\mathbf{I}} \sum_{v=1}^{m} w^{(v)}\text{Tr}\left(\mathbf{P}_r^T\mathbf{D}_r^{(v)^{-\frac{1}{2}}}\mathbf{X}^{(v)}\mathbf{D}_c^{(v)^{-\frac{1}{2}}}\mathbf{P}_c\right). \tag{20}$$

Note that in addition to the constraint $\mathbf{P}_r^T\mathbf{P}_r + \mathbf{P}_c^T\mathbf{P}_c = \mathbf{I}$, $\mathbf{P}_r$ and $\mathbf{P}_c$ should be constrained to be discrete values according to the definitions of $\mathbf{P}_r$ and $\mathbf{P}_c$. This discrete constraint makes the problem very difficult to solve. To address it, we first remove the discrete constraint to make the Eq. (20) solvable with Lemma 3, and then run K-means on $\mathbf{P}_r$ and $\mathbf{P}_c$ to get the discrete solution.

Theorem 3. *Given* $\mathbf{H} \in \mathbb{R}^{n\times d}$, $\mathbf{G} \in \mathbb{R}^{n\times k}$ *and* $\mathbf{F} \in \mathbb{R}^{d\times k}$*. The optimal solutions to the problem*

$$\max_{\mathbf{G}^T\mathbf{G}+\mathbf{F}^T\mathbf{F}=\mathbf{I}} \text{Tr}\left(\mathbf{G}^T\mathbf{H}\mathbf{F}\right) \tag{21}$$

are $\mathbf{G} = \frac{\sqrt{2}}{2}\mathbf{U}_1$, $\mathbf{F} = \frac{\sqrt{2}}{2}\mathbf{V}_1$, *where* $\mathbf{U}_1$, $\mathbf{V}_1$ *are the leading* k *left and right singular vectors of* $\mathbf{H}$*, respectively.*

Proof. Denote the Lagrangian function of Eq. (21) as $\mathcal{L}(\mathbf{G}, \mathbf{F}, \mathbf{\Lambda}) = \text{Tr}(\mathbf{G}^T\mathbf{B}\mathbf{F}) - \text{Tr}(\mathbf{\Lambda}(\mathbf{G}^T\mathbf{G} + \mathbf{F}^T\mathbf{F} - \mathbf{I}))$. Setting the derivative of $\mathcal{L}(\mathbf{G}, \mathbf{F}, \mathbf{\Lambda})$ w.r.t. $\mathbf{G}$ to zero, then we have $\mathbf{B}\mathbf{F} = \mathbf{G}\mathbf{\Lambda}$. Setting the derivative of $\mathcal{L}(\mathbf{G}, \mathbf{F}, \mathbf{\Lambda})$ w.r.t. $\mathbf{F}$ to zero, then we have $\mathbf{B}^T\mathbf{G} = \mathbf{F}\mathbf{\Lambda}$. Thus $\mathbf{B}\mathbf{B}^T\mathbf{G} = \mathbf{B}\mathbf{F}\mathbf{\Lambda} = \mathbf{G}\mathbf{\Lambda}^2$. Therefore, the optimal solution $\mathbf{G}$ should be the eigenvectors of $\mathbf{B}\mathbf{B}^T$, i.e, the left singular vectors of $\mathbf{H}$. Similarly, the optimal solution $\mathbf{F}$ should be the right singular vectors of $\mathbf{H}$. Since it is a maximization problem, the optimal solution **G, F** should be the leading k left and right singular vectors of $\mathbf{H}$, respectively. □

According to Lemma 3, if the discrete constraint on $\mathbf{P}_r$ and $\mathbf{P}_c$ is not considered, the optimal solution $\mathbf{P}_r$ and $\mathbf{P}_c$ to Eq. (20) are the leading k left and right singular vectors of $\widetilde{\mathbf{B}}$ and

$$\widetilde{\mathbf{B}} = \sum_{v=1}^{m} w^{(v)}\left(\left(\mathbf{D}_r^{(v)}\right)^{-\frac{1}{2}}\mathbf{X}^{(v)}\left(\mathbf{D}_c^{(v)}\right)^{-\frac{1}{2}}\right). \tag{22}$$

Table 1
Co-clustering performance on the toy data (%).

	Method	ACC	Purity	NMI
Clustering	SCC(1)	95.45(0.27)	95.45(0.0.27)	83.84(0.80)
Accuracy on	SCC(2)	67.09(0.38)	67.09(0.38)	34.68(0.31)
Rows	BiMVCC	100(0.00)	100(0.00)	100(0.00)
Clustering	SCC(1)	94.00(0.20)	94.00(0.20)	76.16(0.73)
Accuracy on	SCC(2)	65.30(0.82)	65.30(0.82)	26.22(0.61)
Columns	BiMVCC	100(0.00)	100(0.00)	100(0.00)

Since the solution $\mathbf{P}_r$ and $\mathbf{P}_c$ are not discrete values, we need to run the K-means on the rows of $\mathbf{P}$ to obtain the final clustering results.

The detailed algorithm to solve the objective in Eq. (20) is summarized in Algorithm 3.

Algorithm 3 The Algorithm for Solving Eq. (20).

```
Input: Data with $m$ views $\{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(m)}\}$ and the number of classes $k$;
   Initialize the weight factor $w^{(v)} = \frac{1}{m}$ for each view;
Output: Cluster label of each data point.
1: repeat
2:    Calculate $\widetilde{\mathbf{B}}$ as defined in Eq. (22)
3:    Compute $\mathbf{P} = \begin{bmatrix} \mathbf{P}_r \\ \mathbf{P}_c \end{bmatrix}$, where $\mathbf{P}_r$ and $\mathbf{P}_c$ are the leading $k$ left and right singular vectors of $\widetilde{\mathbf{B}}$
4:    Update $w^{(v)}$ according to Eq. (A.4)
5: until converge
6: Treat each row of $\mathbf{P}$ as a new representation of each datapoint and compute the clustering labels by using $K$-meansalgorithm.
```

4. Experiments

In this section, we conduct multiple experiments to evaluate our model. We will introduce the experimental settings throughout the section and present evaluation results on both toy data and benchmark datasets.

Since our method in Algorithm 3 (denoted by BiMVCC) is kind of multi-view learning model, we will perform the proposed method on several benchmark datasets, compared with other related state-of-the-art multi-view clustering methods. In detail, we compare the proposed BiMVCC with following methods:

- Co-trained multi-view spectral clustering (Co-train) [30].
- Co-regularized multi-view spectral clustering (Co-reg) [22].
- Multi-view kernel clustering (MVKKM) [31].
- Multi-view subspace clustering (MVSC) [32].
- Auto-weighted multiple graph learning (AMGL) [33].
- Multi-view clustering with adaptive neighbours (MLAN) [34].
- Multi-view clustering with capped-norm K-means (CaMVC) [23].
- Self-weighted multi-view clustering with soft capped norm (SCMVC) [35].

The classic single view graph-based co-clustering method, spectral co-clustering (SCC) [1], is also included as baseline method. We apply SCC on every dataset using each view of features (e.g., SCC(1) means performing SCC on the 1st view).

To do a comprehensive evaluation, three evaluation metrics are used to evaluate the clustering results, namely, accuracy (ACC), normalized mutual information (NMI), and Purity [35]. Note that these measures with a higher value means a better performance.

4.1. Interpretability of multi-view co-clustering

In this section, we use a randomly generated toy data to verify the effectiveness of our method. We construct a two-view toy dataset where each view is a 110×100 matrix with 30×20, 30×40 and 50×40 block matrices diagonally arranged. The elements within each block matrix denote the edge weight, while the elements outside all blocks denote noise. In detail, elements in all blocks are randomly generated in the range of 0 and 1, while the noise elements in the first view are randomly generated in the range of 0 and 0.5, and noise elements in the second view are in the range of 0 and 0.8, as shown in Fig. 1.

The co-clustering results in terms of ACC, Purity and NMI are recorded in Table 1. It can be verified that SCC can directly recover the desired block diagonal matrix from view 1 but fails on view 2. For the proposed BiMVCC, it perfectly learns the clean block diagonal matrix. By exploring the duality relationship between samples and features, BiMVCC allows for a better interpretability by seeking "block" structures exist in multi-view data space. That is to say, BiMVCC is able to provide compressed representations that are easily interpretable while preserving the co-occurring structure hidden in data. Furthermore, the normalized weights of the two views learned by BiMVCC are around 0.67/0.33. It is clear that the elements within diagonally blocks always contributes to clustering, which shows that the proposed BiMVCC can arrive at the optimal solution in a different style. Note that the first view contains less noise than the second view, according to Eq. (A.4), we can see this phenomenon agrees with our prior inference in the last section. That is, a large weight will be assigned to a good view, while a small weight will be assigned to a weak view. In this way, the proposed auto-weighted strategy achieves the goal of removing the additive hyperparameter while preserving the good performance.

4.2. Experiments on real-world data

To fully assess the performance of our proposed method, the experiments are also performed on several real-world benchmark datasets.

Cornell[1] is the WebKB dataset and has been widely used in multi-view learning [16]. It contains 195 webpages collected from Cornell university. The webpages are distributed over five classes: student, project, course, staff and faculty and described by two views: the content view and the citation view. Each webpage is denoted by 1703 words in the content view, and 195 citation links between other pages in the citation view.

Sonar is selected from UCI machine learning repository.[2] It contains 111 patterns obtained by bouncing sonar signals off a metal cylinder at various angles and under various conditions, and 97 patterns obtained from rocks under similar conditions. The feature of this data is the energy within a particular frequency band, integrated over a certain period of time. Following [36], we formulate the multiple view data by splitting the feature equally and orderly into three views. Thus each view is comprised of 20 features.

BBC is derived from the bbc news corpora which have been previously used in document clustering tasks [37]. The original bbc corpus contains a total of 2225 documents with 5 annotated topic labels. From this corpus we constructed new synthetic datasets with 2-4 views. In our experiments, we construct a subset of BBC with 5470, 5549 and 5583 words in each view, respectively.

Caltech101[3] is an object recognition dataset containing 101 categories of images [38]. Each image is represented by six type of features: Gabor, Wavelet Moments, Centrist, HOG, GIST and LBP. Following [39], we select a subset which contains 2386 images of 20 classes (**Caltech20**): Brain, Binocular, Camera, Car-Side, Dolla-Bill, Face, Ferry, Hedgehog, Garfield, Motorbikes, Leopards, Pagoda, Rhino, Snoopy, Stapler, Stop-Sign, Water-Lilly, Wrench, Windsor Chair and Yin-yang.

4.3. Experimental settings

For the proposed algorithms, Algorithm 3 only needs to conduct SVD on an $n \times d$ matrix in each iteration, while Algorithm 2 needs to perform eigendecomposition on $(n+d) \times (n+d)$ matrix. In some cases, $\min(n, d) \ll (n+d)$, thus Algorithm 3 is much more efficient than Algorithm 2. Therefore, in this section, we use Algorithm 3 to conduct the experiments. Furthermore, inspired by the weighted clustering strategy, the classical unsupervised feature selection method [40] is utilized in our algorithm. It has two primary purposes. The first one is to improve the multi-view clustering performance. The second purpose is to enforce the feature number of all views to be the same, i.e., we select the same number of features for each view, and thus the bipartite graph dimension of each view can be unified. For simplicity, we set $d = \min\{d^{(1)}, d^{(2)}, \dots, d^{(m)}\}$. Note that the selected features are also used as the input of other compared methods for fair comparison. In order to verify the effectiveness of the auto-weighted strategy, the experimental results of Algorithm 1 (denoted by PMVCC) are also recorded. In PMVCC, we search the logarithm of hyperparameter λ, i.e., we vary the value of $\log_{10}\lambda$ from 0.1 to 2 in steps of 0.2 to get the best parameter λ^*.

For other compared methods, we download the source code from the authors' website and follow their experimental setting for fair comparison. We repeat clustering 10 times, and the mean results are recorded. We perform our experiment with Matlab R2016a on a machine with Core 12 Duad 2.6 GHz and 64 GB memory.

4.4. Clustering results

The clustering results of different methods on all the datasets are shown in Tables 2–5. The mean and standard deviation of the performance are reported and the best results are highlighted in boldface.

It can be observed that the multi-view clustering methods, including ours, are generally better than baseline method on each individual view. This indicates considering the integrated information of all the views can improve the clustering performance. Note that the clustering results of our method are better than the other compared multi-view clustering methods, which demonstrates the effectiveness of the proposed method. By making use of the duality between features and samples,

[1] http://lig-membres.imag.fr/grimal/data.html.

[2] http://archive.ics.uci.edu/ml/datasets/.

[3] http://www.vision.caltech.edu/Image_Datasets/Caltech101/.

Table 2
Clustering performance on Cornell (%).

Method	ACC	Purity	NMI
SCC(1)	43.59(0.00)	44.62(0.00)	15.15(0.00)
SCC(2)	42.56(0.00)	42.56(0.00)	10.00(0.00)
Co-train	38.56(0.84)	49.74(0.63)	10.25(0.85)
Co-reg	33.03(2.31)	43.90(0.46)	5.47(1.56)
MVKKM	41.54(3.84)	44.82(1.23)	7.32(1.67)
MVSC	40.61(3.56)	44.92(1.69)	8.57(2.22)
AMGL	42.56(0.52)	43.79(0.28)	3.63(0.48)
MLAN	41.54(0.00)	46.67(0.00)	24.70(0.00)
CaMVC	46.67(2.35)	49.74(2.85)	20.32(1.67)
SCMVC	45.21(1.23)	50.03(3.21)	18.98(1.74)
PMVCC	**48.62(1.60)**	**58.56(0.84)**	26.66(2.56)
BiMVCC	46.15(0.36)	57.18(1.45)	**28.25(0.71)**

Table 3
Clustering performance on Sonar (%).

Method	ACC	Purity	NMI
SCC(1)	62.50(0.00)	62.50(0.00)	15.01(0.00)
SCC(2)	58.17(0.00)	58.17(0.00)	11.70(0.00)
SCC(3)	61.06(0.00)	61.06(0.00)	13.53(0.00)
Co-train	64.13(0.26)	64.13(0.26)	16.61(0.28)
Co-reg	59.13(0.00)	59.13(0.00)	12.68(0.03)
MVKKM	53.37(0.00)	53.37(0.00)	10.20(0.00)
MVSC	57.69(0.00)	57.69(0.00)	12.01(0.00)
AMGL	50.48(0.00)	53.37(0.00)	14.50(0.00)
MLAN	55.77(0.00)	55.77(0.00)	**20.19(0.00)**
CaMVC	54.97(1.21)	54.97(1.21)	10.89(0.38)
SCMVC	51.92(0.34)	54.33(0.01)	10.30(0.06)
PMVCC	64.81(1.29)	64.81(1.29)	17.16(0.88
BiMVCC	**66.12(1.70)**	**66.12(1.70)**	17.86(1.05)

Table 4
Clustering performance on BBC (%).

Method	ACC	Purity	NMI
SCC(1)	26.64(0.04)	26.77(0.09)	11.30(0.05)
SCC(2)	25.98(0.12)	26.58(0.16)	11.29(0.24)
SCC(3)	25.69(0.17)	26.17(0.12)	11.24(0.20)
Co-train	30.50(0.22)	33.47(0.31)	19.54(0.42)
Co-reg	23.93(0.89)	26.01(0.07)	13.20(0.12)
MVKKM	26.53(0.78)	27.22(0.78)	11.76(0.71)
MVSC	26.42(0.01)	27.05(0.33)	12.49(1.56)
AMGL	25.95(0.00)	26.18(0.00)	10.61(0.00)
MLAN	47.59(0.00)	47.59(0.00)	28.86(0.00)
CaMVC	26.05(0.04)	26.18(0.08)	10.85(0.27)
SCMVC	26.78(0.00)	26.78(0.00)	11.45(0.03)
PMVCC	**59.09(1.76)**	**59.53(1.99)**	31.28(0.50)
BiMVCC	56.87(0.39)	59.07(0.35)	**35.13(0.54)**

the co-occurring structure can be better extracted in our model. The superiority of our method also demonstrates the advantage of the proposed auto-weighted strategy, which is able to achieve the goal of removing the additive hyperparameter while preserving the good performance.

Note that the running time of MVSC is even larger than 24 h (86,400 s) when performing the clustering task on Caltech20, as shown in Table 5. This is because MVSC needs to perform the eigendecomposition for each view in each iteration, thus the time complexity of MVSC is extremely large.

4.5. Parameter tuning

For the proposed BiMVCC, as described in Algorithm 3, it has no parameter needs to be tuned. For PMVCC, as described in Algorithm 1, it has an essential regularization parameter λ. Fig. 2 shows the clustering results with respect to different value of λ.

It can be seen that PMVCC achieves the best result at different λ on each data set. For Cornell and BBC, it seems a smaller λ is more suitable, while for Sonar and Caltech20, a larger λ is better. When the value of λ changes, the performance

Table 5
Clustering performance on Caltech20 (%). "–" means that the running time is larger than 24 h (86400 s).

Method	ACC	Purity	NMI
SCC(1)	31.02(1.04)	53.48(0.39)	32.90(0.16)
SCC(2)	32.91(1.59)	60.22(1.31)	36.40(0.75)
SCC(3)	34.58(0.98)	57.94(0.21)	34.55(0.54)
SCC(4)	38.65(0.70)	68.99(0.49)	51.38(0.27)
SCC(5)	32.54(1.35)	65.17(1.04)	42.71(0.94)
SCC(6)	33.85(0.68)	64.02(0.72)	43.19(0.82)
Co-train	37.80(1.61)	70.84(0.86)	54.40(0.99)
Co-reg	39.18(2.39)	69.74(1.13)	55.54(1.01)
MVKKM	46.23(1.76)	72.84(0.58)	54.51(0.78)
MVSC	–	–	–
AMGL	54.70(1.49)	68.24(2.40)	56.80(3.42)
MLAN	56.49(0.00)	68.85(0.00)	54.98(0.00)
CaMVC	44.45(1.12)	70.42(1.17)	54.91(2.06)
SCMVC	46.10(2.00)	70.19(0.11)	53.62(0.45)
PMVCC	52.50(2.36)	71.69(1.79)	56.17(1.08)
BiMVCC	**59.87(0.39)**	**72.97(0.35)**	**57.55(0.54)**

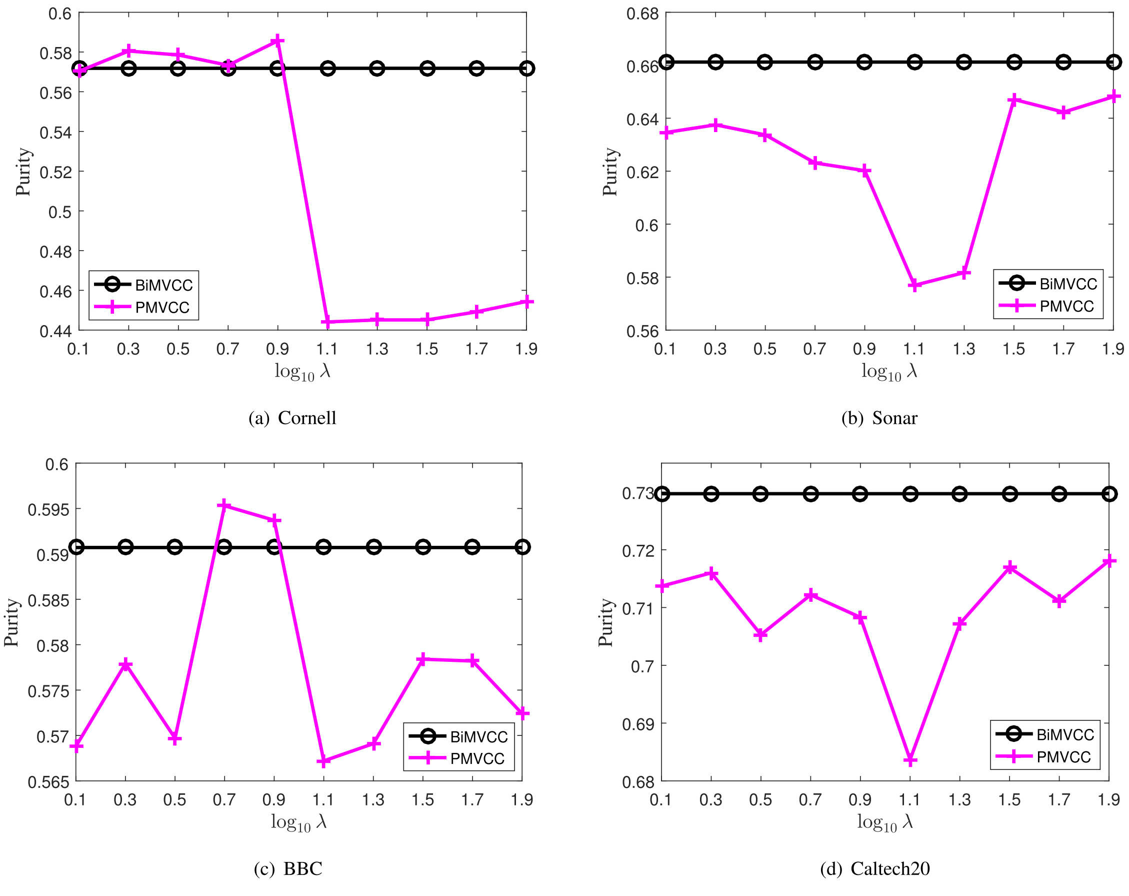

Fig. 2. Clustering results comparison between PMVCC and BiMVCC on different datasets.

of PMVCC drops down dramatically. Thus it is unlikely to apply a fixed λ throughout all the applications. Therefore, the proposed BiMVCC is preferred. This new multiview co-clustering method does not need the extra hyperparameter to learn the weights for each view and without much accuracy loss, is interesting and can be acceptable.

4.6. Convergence study

Although theoretical analysis about the convergence of our algorithm has been provided, in this section, we empirically show how fast our method will converge. The convergence curves of objective in Eq. (13) on all datasets are plotted in

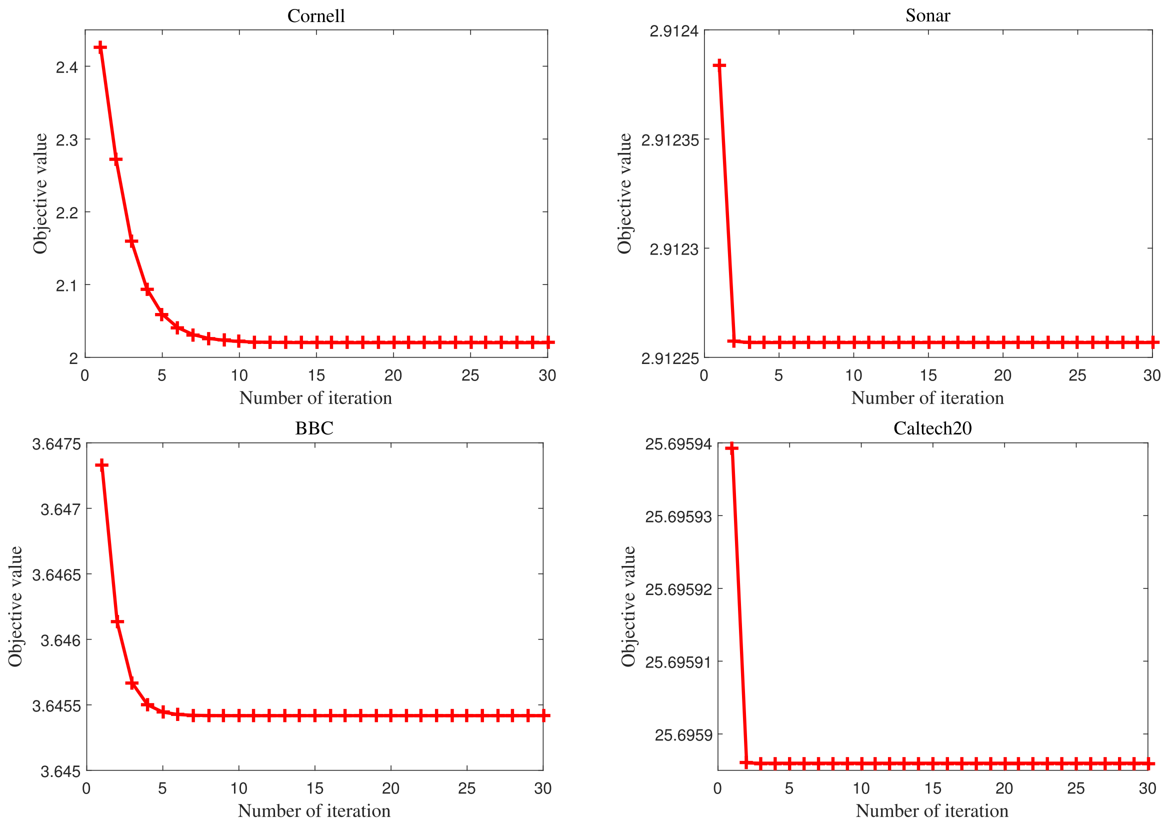

Fig. 3. Objective function value convergence curve *vs* number of iterations of BiMVCC on all datasets.

Fig. 3, where the x-axis denotes the iteration number and the y-axis denotes the value of objective function. It can be observed that the iterative update rules for BiMVCC converge very fast, usually within 10 iterations, which demonstrates the efficiency of our algorithm. In addition, with the effective feature selection strategy utilized in our model, the redundancy of the feature space of each view is small. This is the possible reason why the objective value varied a little with the increasing number of iterations.

5. Conclusion

In this paper, we propose a novel framework for multi-view co-clustering via the reformulation of the spectral learning model. To make use of the duality between features and samples of the multi-view data, a bipartite graph for each view is constructed. The key point of utilizing the multiple bipartite graphs to deal with the multi-view co-clustering task is to reasonably integrate these bipartite graphs and obtain an optimal consensus one. As for this point, the proposed methods can learn an optimal weight for each bipartite graph automatically without introducing extra hyperparameter. An efficient algorithm is proposed to optimize this model and its convergence is also theoretically guaranteed. Extensive experimental results on several benchmark datasets demonstrate that the proposed model outperforms other state-of-the-art multi-view clustering algorithms. In our future work, we are interested in extending the proposed model to other machine learning areas such as kernel learning and semi-supervised learning.

Declaration of Competing Interest

None

Acknowledgment

This work was partially supported by NSF China (No. 61572111), Research Fund for the Central Universities of China (Nos. ZYGX2016Z003, ZYGX2017KYQD177), ARC Future Fellowship FT130100746, ARC grant LP150100671 and DP180100106.

Appendix A. Proof to Theorem 1

Proof. We define an auxiliary function as follows

$$\min_{\mathbf{P}^T\mathbf{P}=\mathbf{I}} \sum_{v=1}^{m} \sqrt{\mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right)}. \tag{A.1}$$

Let Λ be the Lagrange multiplier of Eq. (A.1), thus the Lagrange function of Eq. (A.1) can be written as

$$\sum_{v=1}^{m} \sqrt{\mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right)} + \Gamma(\Lambda, \mathbf{P}), \tag{A.2}$$

where $\Gamma(\Lambda, \mathbf{P})$ denotes the formalized term derived from constraints. Taking the derivative of Eq. (A.2) w.r.t $\mathbf{P}$ and setting the derivative to zero, we obtain

$$\sum_{v=1}^{m} \widehat{w}^{(v)} \frac{\partial \mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right)}{\partial \mathbf{P}} + \frac{\partial \Gamma(\Lambda, \mathbf{P})}{\partial \mathbf{P}} = 0, \tag{A.3}$$

where $\widehat{w}^{(v)}$ is given as the following form [4]

$$\widehat{w}^{(v)} = 1 \Big/ \left(2\sqrt{\mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right)}\right). \tag{A.4}$$

If we perform the same operations on Eq. (13), we can get the same result as shown in Eq. (A.3). Thus, the solution to weights are $w^{(v)} = \frac{1}{2\sqrt{\mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right)}}$. □

Appendix B. Proof to Theorem 2

Proof. Suppose the alternatively updated $\mathbf{P}$ is $\widetilde{\mathbf{P}}$ in each iteration. According to the first two steps of loop in Algorithm 2, we have

$$\widetilde{\mathbf{P}} = \arg\min_{\mathbf{P}^T\mathbf{P}=\mathbf{I}} \sum_{v=1}^{m} w^{(v)} \mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right). \tag{B.1}$$

Combining with Eq. (A.4), i.e., $w^{(v)} = \frac{1}{2\sqrt{\mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right)}}$, we have

$$\sum_{v=1}^{m} \frac{\mathrm{Tr}\left(\widetilde{\mathbf{P}}^T\mathbf{L}^{(v)}\widetilde{\mathbf{P}}\right)}{2\sqrt{\mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right)}} \le \sum_{v=1}^{m} \frac{\mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right)}{2\sqrt{\mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right)}}. \tag{B.2}$$

According to Lemma 1, we have

$$\begin{aligned} &\sum_{v=1}^{m} \sqrt{\mathrm{Tr}\left(\widetilde{\mathbf{P}}^T\mathbf{L}^{(v)}\widetilde{\mathbf{P}}\right)} - \sum_{v=1}^{m} \frac{\mathrm{Tr}\left(\widetilde{\mathbf{P}}^T\mathbf{L}^{(v)}\widetilde{\mathbf{P}}\right)}{2\sqrt{\mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right)}} \\ &\quad \le \sum_{v=1}^{m} \sqrt{\mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right)} - \sum_{v=1}^{m} \frac{\mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right)}{2\sqrt{\mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right)}}. \end{aligned} \tag{B.3}$$

By summing over Eqs. (B.2) and (B.3) in the two sides, we obtain

$$\sum_{v=1}^{m} \sqrt{\mathrm{Tr}\left(\widetilde{\mathbf{P}}^T\mathbf{L}^{(v)}\widetilde{\mathbf{P}}\right)} \le \sum_{v=1}^{m} \sqrt{\mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right)}, \tag{B.4}$$

which completes the prove. That is, the iterative optimization will monotonically decease the objective of Eq. (13) in each iteration until it converges. When the convergence reaches, the equality in Eq. (B.3) holds, thus $\widetilde{\mathbf{P}}$ will satisfy Eq. (A.3), the KKT condition of Eq. (A.1). Therefore, the Algorithm 2 will at least converge to a local optimal solution of Eq. (13). □

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ins.2019.09.079.

[4] To avoid dividing by zero, in practice we can use the following formula $w^{(v)} = \frac{1}{2\sqrt{\mathrm{Tr}\left(\mathbf{P}^T\mathbf{L}^{(v)}\mathbf{P}\right)+\delta}}$, where δ is a very small value, like 0.0001.

References

[1] I.S. Dhillon, Co-clustering documents and words using bipartite spectral graph partitioning, in: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001, pp. 269–274.
[2] F. Nie, X. Wang, C. Deng, H. Huang, Learning a structured optimal bipartite graph for co-clustering, in: Proceedings of the Advances in Neural Information Processing Systems, 2017, pp. 4132–4141.
[3] B. Pontes, R. Giráldez, J.S. Aguilar-Ruiz, Biclustering on expression data: a review, J. Biomed. Inf. 57 (2015) 163–180.
[4] M.-G. Martínez-Peñaloza, E. Mezura-Montes, N. Cruz-Ramírez, H.-G. Acosta-Mesa, H.-V. Ríos-Figueroa, Improved multi-objective clustering with automatic determination of the number of clusters, Neural Comput. Appl. 28 (8) (2017) 2255–2275.
[5] J.d.J. Rubio, D. Ricardo Cruz, I. Elias, G. Ochoa, R. Balcazarand, A. Aguilar, Anfis system for classification of brain signals, J. Intell. Fuzzy Syst. (2019) 1–9.
[6] J. de Jesús Rubio, Usnfis: uniform stable neuro fuzzy inference system, Neurocomputing 262 (2017) 57–66.
[7] X. Li, H. Li, B. Sun, F. Wang, Assessing information security risk for an evolving smart city based on fuzzy and grey fmea, J. Intell. Fuzzy Syst. 34 (4) (2018) 2491–2501.
[8] J. de Jesús Rubio, Sofmls: online self-organizing fuzzy modified least-squares network, IEEE Trans. Fuzzy Syst. 17 (6) (2009) 1296–1309.
[9] D. Agarwal, S. Merugu, Predictive discrete latent factor models for large scale dyadic data, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2007, pp. 26–35.
[10] S. Huang, Z. Xu, J. Lv, Adaptive local structure learning for document co-clustering, Knowl. Based Syst. 148 (2018) 74–84.
[11] R. Bro, E.E. Papalexakis, E. Acar, N.D. Sidiropoulos, Coclustering—a useful tool for chemometrics, J. Chemomet. 26 (6) (2012) 256–263.
[12] A.L.V. Pereira, E.R. Hruschka, Simultaneous co-clustering and learning to address the cold start problem in recommender systems, Knowl. Based Syst. 82 (2015) 11–19.
[13] A. Oliva, A. Torralba, Modeling the shape of the scene: a holistic representation of the spatial envelope, Int. J. Comput. Vis. 42 (3) (2001) 145–175.
[14] H. Yu, M. Li, H.-J. Zhang, J. Feng, Color texture moments for content-based image retrieval, in: Proceedings of the International Conference on Image Processing, 3, 2002, pp. 929–932.
[15] T.-X. Wu, X.-C. Lian, B.-L. Lu, Multi-view gender classification using symmetry of facial images, Neural Comput. Appl. 21 (4) (2012) 661–669.
[16] G. Bisson, C. Grimal, Co-clustering of multi-view datasets: a parallelizable approach, in: Proceedings of the IEEE 12th International Conference on Data Mining, 2012, pp. 828–833.
[17] S. Sun, A survey of multi-view machine learning, Neural Comput. Appl. 23 (7-8) (2013) 2031–2038.
[18] Y. Cui, X.Z. Fern, J.G. Dy, Non-redundant multi-view clustering via orthogonalization, in: Proceedings of the IEEE International Conference on Data Mining, 2007, pp. 133–142.
[19] Z. Xu, I. King, M.R. Lyu, Web page classification with heterogeneous data fusion, in: Proceedings of the 16th International Conference on World Wide Web, WWW, Banff, Alberta, Canada, May 8-12, 2007, 2007, pp. 1171–1172.
[20] T. Li, J.M. Corchado, S. Sun, J. Bajo, Clustering for filtering: multi-object detection and estimation using multiple/massive sensors, Inf. Sci. 388–389 (2017) 172–190.
[21] X. Zhang, H. Gao, G. Li, J. Zhao, J. Huo, J. Yin, Y. Liu, L. Zheng, Multi-view clustering based on graph-regularized nonnegative matrix factorization for object recognition, Inf. Sci. 432 (2018) 463–478.
[22] A. Kumar, P. Rai, H. Daume, Co-regularized multi-view spectral clustering, in: Proceedings of the Advances in Neural Information Processing Systems, 2012, pp. 1413–1421.
[23] S. Huang, Y. Ren, Z. Xu, Robust multi-view data clustering with multi-view capped-norm k-means, Neurocomputing 311 (2018) 197–208.
[24] X. Zhang, H. Sun, Z. Liu, Z. Ren, Q. Cui, Y. Li, Robust low-rank kernel multi-view subspace clustering based on the schatten p-norm and correntropy, Inf. Sci. 477 (2019) 430–447.
[25] I.S. Dhillon, S. Mallela, D.S. Modha, Information-theoretic co-clustering, in: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003, pp. 89–98.
[26] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.
[27] S. Huang, H. Wang, D. Li, Y. Yang, T. Li, Spectral co-clustering ensemble, Knowl. Based Syst. 84 (2015) 46–55.
[28] I. Daubechies, R. DeVore, M. Fornasier, C.S. Güntürk, Iteratively reweighted least squares minimization for sparse recovery, Commun. Pure Appl. Math. A J. Issued Courant Inst. Math. Sci. 63 (1) (2010) 1–38.
[29] F. Nie, H. Huang, X. Cai, C.H. Ding, Efficient and robust feature selection via joint $\ell_{2,1}$-norms minimization, in: Proceedings of the NIPS, 2010, pp. 1813–1821.
[30] A. Kumar, H. Daumé, A co-training approach for multi-view spectral clustering, in: Proceedings of the 28th International Conference on Machine Learning, 2011, pp. 393–400.
[31] G. Tzortzis, A. Likas, Kernel-based weighted multi-view clustering, in: Proceedings of the 12th IEEE International Conference on Data Mining, 2012, pp. 675–684.
[32] H. Gao, F. Nie, X. Li, H. Huang, Multi-view subspace clustering, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 4238–4246.
[33] F. Nie, J. Li, X. Li, Parameter-free auto-weighted multiple graph learning: a framework for multiview clustering and semi-supervised classification., in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, 2016, pp. 1881–1887.
[34] G.C. Feiping Nie, X. Li, Multi-view clustering and semi-supervised classification with adaptive neighbours, in: Proceedings of the 31st AAAI Conference on Artificial Intelligence, 2017, pp. 2408–2414.
[35] S. Huang, Z. Kang, Z. Xu, Self-weighted multi-view clustering with soft capped norm, Knowl. Based Syst. 158 (2018) 1–8.
[36] C. Hou, C. Zhang, Y. Wu, F. Nie, Multiple view semi-supervised dimensionality reduction, Pattern Recogn. 43 (3) (2010) 720–730.
[37] D. Greene, P. Cunningham, A matrix factorization approach for integrating multiple data views, in: Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2009, pp. 423–438.
[38] L. Fei-Fei, R. Fergus, P. Perona, Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories, Comput. Vis. Image Understand. 106 (1) (2007) 59–70.
[39] Y. Li, F. Nie, H. Huang, J. Huang, Large-scale multi-view spectral clustering via bipartite graph, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2015, pp. 2750–2756.
[40] D. Cai, C. Zhang, X. He, Unsupervised feature selection for multi-cluster data, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010, pp. 333–342.