Knowledge-Based Systems 174 (2019) 73–86

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Incremental multi-view spectral clustering

Peng Zhou [a], Yi-Dong Shen [b,*], Liang Du [c], Fan Ye [a], Xuejun Li [a]

[a] School of Computer Science and Technology, Anhui University, Hefei 230601, China
[b] The State Key Lab of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
[c] School of Computer and Information Technology, Shanxi University, Taiyuan 030006, China

ARTICLE INFO

Article history:
Received 13 November 2018
Received in revised form 8 February 2019
Accepted 28 February 2019
Available online 8 March 2019

Keywords:
Multi-view clustering
Spectral clustering
Incremental learning

ABSTRACT

Multi-view learning has attracted increasing attention in recent years, and the existing multi-view learning methods learn a consensus result by collecting all views. These methods have two obvious limitations. First, it is not scalable; with limited computational resources it would be difficult, if not impossible, to collect and process a large collection of views together. Second, in many applications views of data are available over time; it is infeasible to apply the existing multi-view learning methods to such streaming views. To address the two limitations, in this paper we propose a novel incremental multi-view spectral clustering (IMSC) method. In IMSC, instead of ensembling the collection of all views simultaneously, we integrate them one by one in an incremental way. We first learn an initial model from a small number of views; next when a new view is available, we need only use it to update the model and apply the updated model to learn a consensus result. This method is scalable and applicable to streaming views. To further reduce the time and space complexity, we apply low rank approximation by means of the well-known random Fourier features to construct the base kernels and do low rank SVD decompositions accordingly. The theoretical analysis and experimental results on benchmark data sets show that our incremental multi-view spectral clustering method is significantly faster in efficiency than the existing state-of-the-art non-incremental ones and is comparable or even better in clustering quality.

1. Introduction

Data on Internet often have multi-view representations, which are generated from different data sources or descriptors. For example, web data may contain different views such as texts, images and videos; even a single type of data like visual data can be represented by different descriptors, such as SIFT [1] and HOG [2].

Several learning methods have been proposed to handle multi-view data [3–9]. These methods usually take the collection of all views as inputs and learn a consensus result by extending single view learning methods, such as SVM, k-means, spectral clustering and so on, to the multi-view setting. For example, in supervised learning, Sun et al. [10] extended SVM to multi-view setting, and Chen et al. [11] introduced the multi-view Fisher discriminant analysis which is applicable for both binary and multi-class classification. In unsupervised learning, Kumar et al. [4] and Xia et al. [7] extended spectral clustering to multi-view setting; Cai et al. [5] presented a k-means based clustering method; Liu et al. [12] applied nonnegative matrix factorization method to learn a consensus clustering; Wang et al. [13] proposed a graph-based multi-view clustering method; Zhang et al. [14] combined multi-view clustering and multi-task learning in heterogeneous situations; Wang et al. [15] used multi-view clustering to detect coherent groups in crowd scenes.

The existing multi-view learning methods have two obvious limitations. First, it is not scalable; with limited computational resources it would be difficult to collect and process a large collection of views together. Second, in many applications views of data are available over time; it is infeasible to apply the existing methods to such streaming views. For example, in automatic detection of subkilometer craters in high-resolution planetary images [16,17], the instances are craters which are continually monitored by the latest available high-resolution images which are updated over time. For a crater, each image can be regarded as a view and the number of views increases with time. As another example, in dangerous gas detection system [18], a number of sensors have been deployed to detect the chemical gas and sample the field data once in a while. In this application, there are a number of gaseous substances which need to be classified and each gas substance is continuously monitored by several sensors. The sampled data at each time interval constitute a view, so the number of views increases over time. In this streaming

* Corresponding author.
E-mail addresses: zhoupeng@ahu.edu.cn (P. Zhou), ydshen@ios.ac.cn (Y.-D. Shen), duliang@sxu.edu.cn (L. Du), yfan@ahu.edu.cn (F. Ye), xjli@ahu.edu.cn (X. Li).

https://doi.org/10.1016/j.knosys.2019.02.036
0950-7051/

view setting, our task is to learn a new consensus result when a new view arrives. However, all of the aforementioned multi-view learning methods, which learn a consensus result by collecting all views, are infeasible for streaming views, because there would be an endless number of views available and it is too expensive, if not impossible, to store all historical views in a repository and process all of them.

To address the two limitations, in this paper we propose a novel Incremental Multi-view Spectral Clustering (IMSC) method, i.e., instead of processing all views simultaneously, we integrate them one by one in an incremental way. Particularly, we always keep a model which represents a collection of previous views. This model consists of a few base kernels together with a spectral embedding. Initially, we construct the base kernels and learn a spectral embedding from the first few views; next, when one more view is available, we learn a consensus kernel from the base kernels and the new view and update the base kernels simultaneously; finally, we construct a Laplacian matrix from the consensus kernel and apply the Laplacian matrix to update the spectral embedding.

Kernel learning and spectral clustering often involves Singular Value Decomposition (SVD) operations on kernels whose time and space complexity is $O(n^3)$ and $O(n^2)$, respectively, where n is the number of instances. To improve the performance of IMSC, we present several speedup strategies. Particularly, instead of directly constructing a Gaussian kernel from a view, we learn and store a low rank approximation to the kernel by means of random Fourier features [19]; we then do low rank SVD decompositions accordingly. As a result, we reduce the complexity of time and space to $O(nr^2 + r^3)$ and $O(nr)$, respectively, where r is the rank of the kernel matrix and often $r \ll n$. Due to the low rank approximation, our method can easily handle the case that data contains a small quantity of new instances or missing instances which often happens in the streaming view setting.

To demonstrate the effectiveness and efficiency of our method, we conduct extensive experiments on benchmark data sets. The experimental results demonstrate that, while being comparable in clustering quality, our IMSC is significantly faster than the multi-view clustering methods.

The paper is organized as follows. Section 2 describes some related work. Section 3 gives some preliminaries. Section 4 presents in detail the main steps of IMSC. Section 5 shows the experimental results, and Section 6 concludes the paper. In order not to distract from the reading, proofs of the results are moved to Appendix.

2. Related work

In this section, we briefly review representative multi-view clustering methods reported in the literature.

Multi-view clustering methods learn a consensus clustering result from a collection of views by extending existing clustering methods for single view data, such as k-means or spectral clustering. Cai et al. [5] extended k-means for multi-view data, leading to a multi-view k-means clustering method. Liu et al. [12] extended the non-negative matrix factorization (NMF) to multi-view clustering, where a joint NMF process is formulated with the constraint that pushes clustering solution of each view towards a common consensus. Günnemann [20] presented a multi-view clustering method based on subspace learning, which provides multiple generalizations of the data by modeling individual mixture models, each representing a distinct view. Cao et al. [21] also presented a subspace learning based multi-view clustering method by making full use of the diversities in each view. Zhang et al. [22] provided a multi-view subspace clustering method which explores complementary information from multiple views and simultaneously seeks the underlying latent representation. Xie et al. [23] applied tensor multi-rank minimization to multi-view subspace learning. Ren et al. [24] presented a robust auto-weighted multi-view clustering method to handle noises on the multiple views.

Since kernel method is widely-used in machine learning tasks, it is natural to apply kernel methods to transfer multi-view learning into multiple kernel learning. So it is worth mentioning multiple kernel learning methods here. Liu et al. extended extreme learning machine and kernel k-means to multiple kernel learning in [25,26], respectively. Liu et al. [27] proposed a multiple kernel learning method to enhance the representability of the optimal kernel and strengthen the negotiation between kernel learning and clustering. Since kernel plays important role in some spectral method, some work focused on kernel spectral clustering. For example, Peluffo-Ordónez et al. [28] linearly combined multiple kernels and applied it to kernel spectral clustering for dynamic data clustering. Alazte et al. [29] proposed a weighted kernel PCA for spectral clustering which can handle out-of-sample problem.

The most closely related to our work are multi-view spectral clustering methods of [4,7,30–34]. Kumar et al. [30] presented a co-training approach for multi-view spectral clustering by bootstrapping the clusterings of different views. Kumar et al. [4] also proposed two coregularization based approaches for multi-view spectral clustering by enforcing the clustering hypotheses on different views to agree with each other; the approaches construct an objective function consisting of the graph Laplacians from all views and make regularizations on the eigenvectors of the Laplacians such that the resulting cluster structures are consistent. Xia et al. [7] proposed a robust multi-view spectral clustering method by building a Markov chain based on a low rank and sparse decomposition method. In the process of building the Markov chain, this method decomposes the transition matrix of each view into a low rank consensus transition matrix and a sparse noise matrix. Then it applies an Alternating Direction Method of Multipliers (ADMM) algorithm [35] to learn the consensus transition matrix and obtain the final clustering result from this consensus transition matrix. Li et al. [33] constructed a bipartite graph for large-scale multi-view spectral clustering. Nie et al. [31,32] presented a parameter-free multi-view spectral clustering method, which learns an optimal weight for each view automatically without introducing an additive parameter. Recently, Nie et al. [34] learned spectral embedding from each view and combined them via adaptively weighted procrustes.

As we discussed in Introduction, all of the aforementioned multi-view learning methods have two limitations: they are not scalable and cannot be applied for streaming view data. In addition, the spectral based multi-view learning methods need to save all kernel or Laplacian matrices. When the number of views grows, it is very space consuming.

It is worthy to note that, there have existed some researches about multi-view learning on the streaming data [28,29,36–38]. For example, [36,37] focus on multi-view video processing, which contains multiple video stream shotted from different viewpoints. [28,29,38] are multiple kernel learning methods which handle streaming data or out-of-sample data. These works are different from our streaming views setting. The multi-view video is a collection of multiple videos capturing the same 3D scene at different viewpoints. So the number of views is the number of different viewpoints and is fixed. Similarly, those multiple kernel learning methods handle the data whose instances arrive in a stream, while the number of kernels is fixed. However, in our setting, the number of views or kernels is increasing with time.

3. Preliminaries

In this section, we briefly review spectral clustering and random Fourier features method. Throughout the paper, we use boldface uppercase and lowercase letters to denote matrices and vectors, respectively. The (i, j)th element of a matrix $\mathbf{M}$ is denoted by $M_{i,j}$.

3.1. Spectral clustering

Spectral clustering [39] is a widely used clustering method. There are many spectral clustering method, such as normalized cut (NCut) [40], random walk [41], kernel spectral clustering [29] and so on. In this paper, we use the NCut method. Given a data set containing n data points/instances $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, we define a sparse similarity matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$, where $W_{i,j} \geq 0$ is the similarity of $\mathbf{x}_i$ and $\mathbf{x}_j$. There are many ways to construct $\mathbf{W}$. In this paper, we construct it from Gaussian kernels because Gaussian kernel is one of the most widely-used kernel function in machine learning methods. In more details, we compute the kernel value $K(\mathbf{x}_i, \mathbf{x}_j)$ of two instances $\mathbf{x}_i$ and $\mathbf{x}_j$ as $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|_2^2}{2\sigma^2}}$, where σ is the bandwidth parameter. Then we construct $\mathbf{W}$ by setting $W_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$ if $\mathbf{x}_j$ is one of the top k nearest instance to $\mathbf{x}_i$ and $W_{i,j} = 0$ otherwise. After obtaining $\mathbf{W}$, we construct a Laplacian matrix $\mathbf{L} \in \mathbb{R}^{n\times n}$ with $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{I}$ is an identity matrix and $\mathbf{D} \in \mathbb{R}^{n\times n}$ is a diagonal matrix with the (i, i)th element $D_{i,i} = \sum_{j=1}^{n} W_{i,j}$. Spectral clustering aims to learn a spectral embedding $\mathbf{Y} \in \mathbb{R}^{n\times c}$ (c is the dimension of the embedding space and is often set to the number of clusters) by optimizing the following function

$$\begin{aligned} &\min_{\mathbf{Y}} \quad tr(\mathbf{Y}^T\mathbf{L}\mathbf{Y}), \\ &s.t. \quad \mathbf{Y}^T\mathbf{Y} = \mathbf{I}. \end{aligned} \tag{1}$$

where $tr(\cdot)$ is the trace function.

Eq. (1) can be solved by eigenvalue decomposition of $\mathbf{L}$, i.e., $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_c]$, where $\mathbf{y}_1, \ldots, \mathbf{y}_c$ are the eigenvectors corresponding to the smallest c eigenvalues of $\mathbf{L}$.

Once the spectral embedding $\mathbf{Y}$ is obtained, we apply spectral rotation [42] to discretize $\mathbf{Y}$, thus yielding the final clustering result $\mathbf{C} \in \{0, 1\}^{n\times c}$ (i.e., if the ith data instance belongs to the jth cluster, then $C_{i,j} = 1$ and $C_{i,1}, \ldots, C_{i,j-1}, C_{i,j+1}, \ldots, C_{i,c}$ are all zeros). Spectral rotation solves the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{C},\mathbf{R}^*} \quad \|\mathbf{C} - \mathbf{Y}\mathbf{R}^*\|_F^2, \\ &s.t. \quad \mathbf{C} \in \{0, 1\}^{n\times c}, \quad \mathbf{C}\mathbf{1}_c = \mathbf{1}_n, \quad \mathbf{R}^{*T}\mathbf{R}^* = \mathbf{I}. \end{aligned} \tag{2}$$

where $\mathbf{R}^* \in \mathbb{R}^{c\times c}$ is a rotation matrix, $\mathbf{1}_c$ (resp. $\mathbf{1}_n$) is a vector of length c (resp. n) whose elements are all 1, and $\|\cdot\|_F$ is the Frobenius norm. Yu et al. [42] presented an SVD based method to solve Eq. (2) with the time complexity $O(nc^2 + c^3)$.

3.2. Random Fourier features

Random Fourier features method [19] is proposed by Rahimi et al., and used to learn a low rank approximation of a kernel matrix, i.e., given a kernel matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$, it aims to learn a low rank matrix $\mathbf{H} \in \mathbb{R}^{r\times n}$ with $r \ll n$, such that $\mathbf{H}^T\mathbf{H} \approx \mathbf{K}$.

The random Fourier features method is based on the following Bochner's theorem:

Theorem 1 ([43]). *A continuous kernel function $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y})$ on $\mathbb{R}^d$ is positive definite if and only if its Fourier transform is a non-negative measure.*

If the kernel function $k(\cdot)$ is properly scaled, Bochner's theorem guarantees that its Fourier transform $p(\boldsymbol{\omega})$ is a proper probability distribution, i.e.,

$$\begin{aligned} k(\mathbf{x} - \mathbf{y}) &= \int_{\mathbb{R}^d} p(\boldsymbol{\omega}) e^{j\boldsymbol{\omega}^T(\mathbf{x}-\mathbf{y})}\, d\boldsymbol{\omega} \\ &= \int_{\mathbb{R}^d} p(\boldsymbol{\omega}) \cos(\boldsymbol{\omega}^T(\mathbf{x} - \mathbf{y}))\, d\boldsymbol{\omega} \\ &\quad + j \int_{\mathbb{R}^d} p(\boldsymbol{\omega}) \sin(\boldsymbol{\omega}^T(\mathbf{x} - \mathbf{y}))\, d\boldsymbol{\omega} \\ &= \int_{\mathbb{R}^d} p(\boldsymbol{\omega}) \cos(\boldsymbol{\omega}^T(\mathbf{x} - \mathbf{y}))\, d\boldsymbol{\omega} \end{aligned} \tag{3}$$

where j is the imaginary unit; the second equality is by Euler's formula and the third one is due to that the imaginary part must be zero since $k(\cdot)$ is a real-valued function. Since $p(\boldsymbol{\omega})$ is a probability distribution, the above integral can be viewed as the expectation $\mathbb{E}_\omega$ of $\cos(\boldsymbol{\omega}^T(\mathbf{x} - \mathbf{y}))$ w.r.t. ω. Since $\cos(\alpha - \beta) = \cos\alpha\cos\beta + \sin\alpha\sin\beta$, Eq. (3) can be rewritten as:

$$k(\mathbf{x} - \mathbf{y}) = \mathbb{E}_\omega[f(\boldsymbol{\omega}, \mathbf{x})^T f(\boldsymbol{\omega}, \mathbf{y})] \tag{4}$$

where $f(\boldsymbol{\omega}, \mathbf{x}) = [\cos(\boldsymbol{\omega}^T\mathbf{x}), \sin(\boldsymbol{\omega}^T\mathbf{x})]^T$.

Then, the idea of random Fourier features is to use the empirical mean over $r/2$ Fourier components $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_{r/2}$, sampled from the distribution $p(\boldsymbol{\omega})$, to approximate the expectation Eq. (4) and thus obtain a rank-r approximation $\mathbf{H} \in \mathbb{R}^{r\times n}$ to the kernel $\mathbf{K}$ whose kernel function is $k(\mathbf{x} - \mathbf{y})$ (i.e., $\mathbf{K} \approx \mathbf{H}^T\mathbf{H}$). More formally, let the data set $\mathcal{X}$ contains instances $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, then

$$\mathbf{H} = [\mathbf{h}(\mathbf{x}_1)^T, \ldots, \mathbf{h}(\mathbf{x}_n)^T] \tag{5}$$

where

$$\mathbf{h}(\mathbf{x}) = \frac{\sqrt{2}}{\sqrt{r}}[\cos(\boldsymbol{\omega}_1^T\mathbf{x}), \ldots, \cos(\boldsymbol{\omega}_{r/2}^T\mathbf{x}), \sin(\boldsymbol{\omega}_1^T\mathbf{x}), \ldots, \sin(\boldsymbol{\omega}_{r/2}^T\mathbf{x})] \tag{6}$$

Intuitively, $\mathbf{H}^T\mathbf{H}$ is closer to $\mathbf{K}$ when r gets larger.

4. Incremental multi-view spectral clustering

In this paper, We use $\mathcal{X} = \{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(t)}, \ldots, \mathbf{X}^{(v)}\}$ to denote a multi-view data set, where $\mathbf{X}^{(t)} \in \mathbb{R}^{n\times d_t}$ is the tth view of $\mathcal{X}$ with n instances and d_t features. We also use $\mathbf{C}^{(t)} \in \{0, 1\}^{n\times c}$ to denote the clustering result of the first t views of $\mathcal{X}$, where c is the number of clusters. If the ith instance belongs to the jth cluster, $C_{i,j}^{(t)} = 1$ and $C_{i,1}^{(t)}, \ldots, C_{i,j-1}^{(t)}, C_{i,j+1}^{(t)}, \ldots, C_{i,c}^{(t)}$ are all zeros. The task of multi-view spectral clustering is to learn $\mathbf{C}^{(t)}$ ($t \geq 1$) from $\mathcal{X}$ by means of spectral embedding.

The idea of incremental multi-view learning is to integrate a collection of views one by one in an incremental way; i.e., we keep a model representing the previous views and when the next view is available, we update the model with the new view and then learn a consensus result by applying the updated model.

In our incremental multi-view spectral clustering (IMSC), the model consists of a few base kernels as well as a spectral embedding. Initially, we construct m Gaussian kernels as base kernels $\mathbf{K}_b^{(1)}$, $\mathbf{K}_b^{(2)}$, ..., $\mathbf{K}_b^{(m)}$ from the first m views $\mathbf{X}^{(1)}$, $\mathbf{X}^{(2)}$, ..., $\mathbf{X}^{(m)}$, respectively (in our experiments we choose $m = 2$), and learn an initial consensus spectral embedding $\mathbf{Y}^{(m)}$ from the m initial base kernels. Next, we combine the base kernels and the new kernel built from the $(m + 1)$th view $\mathbf{X}^{(m+1)}$ to learn a consensus kernel $\mathbf{K}^{(m+1)}$ while jointly updating the base kernels. Finally, we construct a consensus Laplacian matrix $\mathbf{L}^{(m+1)}$ from $\mathbf{K}^{(m+1)}$ and apply it and $\mathbf{Y}^{(m)}$ to learn a spectral embedding $\mathbf{Y}^{(m+1)}$. The process is repeated until no more view is available. Apparently, the most difficult part of IMSC is how to learn a consensus kernel $\mathbf{K}^{(t)}$ ($t > m$) and update the base kernels.

4.1. Learning a consensus kernel $\mathbf{K}^{(t)}$ and update base kernels

Given the base kernels $\mathbf{K}_b^{(1)}, \mathbf{K}_b^{(2)}, \ldots, \mathbf{K}_b^{(m)}$ and a new view $\mathbf{X}^{(t)}$ ($t > m$), we want to learn a consensus kernel $\mathbf{K}^{(t)}$ and jointly update these base kernels. To achieve this, we first construct a Gaussian kernel $\mathbf{K}_c^{(t)}$ from the current new view $\mathbf{X}^{(t)}$. By abuse of notation, in the sequel we use $\hat{\mathbf{K}}_b^{(1)}, \hat{\mathbf{K}}_b^{(2)}, \ldots, \hat{\mathbf{K}}_b^{(m)}$ to denote the current base kernels and use $\mathbf{K}_b^{(1)}, \mathbf{K}_b^{(2)}, \ldots, \mathbf{K}_b^{(m)}$ to denote the updated base kernels by applying the new view $\mathbf{X}^{(t)}$. Let $rank(\mathbf{K})$ denote the rank of a kernel matrix $\mathbf{K}$. Then we aim for an objective function with the following properties:

1. The consensus kernel $\mathbf{K}^{(t)}$ is expected to be as close as possible to each $\mathbf{K}_b^{(i)}$ of the updated base kernels; i.e., the difference $\sum_{i=1}^{m} \|\mathbf{K}^{(t)} - \mathbf{K}_b^{(i)}\|_F^2$ should be minimized.
2. $\mathbf{K}^{(t)}$ is expected to be as close as possible to the kernel $\mathbf{K}_c^{(t)}$ of the new view $\mathbf{X}^{(t)}$; i.e., the difference $\|\mathbf{K}^{(t)} - \mathbf{K}_c^{(t)}\|_F^2$ should be minimized.
3. The base kernels should be stable, which means these kernels before and after being updated by the new view are expected to be as close as possible. Therefore, the difference $\sum_{i=1}^{m} \|\mathbf{K}_b^{(i)} - \hat{\mathbf{K}}_b^{(i)}\|_F^2$ should be minimized.
4. Furthermore, in order to have a clear cluster structure [44, 45], the consensus kernel and the updated base kernels are supposed to be low rank; i.e., $rank(\mathbf{K}^{(t)}) + \sum_{i=1}^{m} rank(\mathbf{K}_b^{(i)})$ should be minimized.

Formally we present the following objective function, which learns the consensus kernel $\mathbf{K}^{(t)}$ and updates the base kernels $\mathbf{K}_b^{(i)}$ ($1 \le i \le m$) jointly:

$$\begin{aligned} \min_{\mathbf{K}_b^{(1)},\ldots,\mathbf{K}_b^{(m)},\mathbf{K}^{(t)}} & \sum_{i=1}^{m} \|\mathbf{K}^{(t)} - \mathbf{K}_b^{(i)}\|_F^2 + \|\mathbf{K}^{(t)} - \mathbf{K}_c^{(t)}\|_F^2 \\ & + \lambda_1 \sum_{i=1}^{m} \|\mathbf{K}_b^{(i)} - \hat{\mathbf{K}}_b^{(i)}\|_F^2 \\ & + \lambda_2 \left(rank(\mathbf{K}^{(t)}) + \sum_{i=1}^{m} rank(\mathbf{K}_b^{(i)}) \right), \\ s.t. \quad & \mathbf{K}^{(t)T} = \mathbf{K}^{(t)}, \quad \mathbf{K}^{(t)} \succeq 0, \\ & \mathbf{K}_b^{(i)T} = \mathbf{K}_b^{(i)}, \quad \mathbf{K}_b^{(i)} \succeq 0, \forall 1 \le i \le m. \end{aligned} \tag{7}$$

where λ_1 and λ_2 are two balancing parameters. Note that the constraints in Eq. (7) are necessary to guarantee that the consensus kernel $\mathbf{K}^{(t)}$ and all updated base kernels $\mathbf{K}_b^{(i)}$ ($1 \le i \le m$) are valid kernel matrices, i.e., they are symmetric and positive semi-definite.

Next we describe how to solve the above objective function. Since the optimization problem in Eq. (7) involves minimizing the variables $\mathbf{K}^{(t)}$ and $\mathbf{K}_b^{(i)}$ ($1 \le i \le m$), we solve it using an alternating minimization algorithm.

4.1.1. Optimize $\mathbf{K}^{(t)}$ by fixing $\mathbf{K}_b^{(i)}$

When all $\mathbf{K}_b^{(i)}$ ($1 \le i \le m$) are fixed, by expanding the Frobenius norms, Eq. (7) can easily be rewritten as:

$$\begin{aligned} \min_{\mathbf{K}^{(t)}} \quad & \|\mathbf{K}^{(t)} - \mathbf{A}\|_F^2 + \tau \; rank(\mathbf{K}^{(t)}), \\ s.t. \quad & \mathbf{K}^{(t)T} = \mathbf{K}^{(t)}, \quad \mathbf{K}^{(t)} \succeq 0. \end{aligned} \tag{8}$$

where $\mathbf{A} = \frac{1}{m+1}\left(\sum_{i=1}^{m} \mathbf{K}_b^{(i)} + \mathbf{K}_c^{(t)}\right)$ and $\tau = \frac{\lambda_2}{m+1}$. Due to the low rank term $\tau \; rank(\mathbf{K}^{(t)})$, Eq. (8) is a non-convex discontinuous function. Fortunately, we can get the global optima of this sub-problem by applying an SVD based method as follows.

Since $\mathbf{K}^{(t)}$ and $\mathbf{A}$ are symmetric and positive semi-definite, we can write $\mathbf{K}^{(t)} = \mathbf{U}^{(t)}\mathbf{S}^{(t)}\mathbf{U}^{(t)T}$ and $\mathbf{A} = \mathbf{U}_A\mathbf{S}_A\mathbf{U}_A^T$ as the SVD of $\mathbf{K}^{(t)}$ and $\mathbf{A}$, respectively, where $\mathbf{U}^{(t)} = [\mathbf{u}_1, \ldots, \mathbf{u}_n]$ and $\mathbf{U}_A = [\mathbf{v}_1, \ldots, \mathbf{v}_n]$ are n eigenvectors of $\mathbf{K}^{(t)}$ and $\mathbf{A}$ respectively, and $\mathbf{S}^{(t)}$, $\mathbf{S}_A$ are diagonal matrices whose diagonal elements are $\sigma_1, \ldots, \sigma_n$ and $\theta_1, \ldots, \theta_n$ respectively. Here $\sigma_1, \ldots, \sigma_n$ are n eigenvalues of $\mathbf{K}^{(t)}$ and $\theta_1, \ldots, \theta_n$ are n eigenvalues of $\mathbf{A}$, and all of these eigenvalues should be greater than or equal to 0.

We present the following principal theorem used to solve Eq. (8).

Theorem 2. *Let $\mathbf{K}^{(t)} = \mathbf{U}^{(t)}\mathbf{S}^{(t)}\mathbf{U}^{(t)T}$ and $\mathbf{A} = \mathbf{U}_A\mathbf{S}_A\mathbf{U}_A^T$ be the SVD of $\mathbf{K}^{(t)}$ and $\mathbf{A}$, respectively as defined above. Then, under the following conditions:*

$$\mathbf{U}^{(t)} = \mathbf{U}_A, \quad \sigma_i = \begin{cases} \theta_i, & \theta_i \ge \sqrt{\tau} \\ 0, & \theta_i < \sqrt{\tau} \end{cases} \tag{9}$$

$\mathbf{K}^{(t)}$ is the global optima of Eq. (8).

Proof. See Appendix.

Therefore, we can obtain the solution of Eq. (8) by first computing the SVD $\mathbf{A} = \mathbf{U}_A\mathbf{S}_A\mathbf{U}_A^T$, then computing $\sigma_1, \ldots, \sigma_n$ using Eq. (9) which yields the diagonal matrix $\mathbf{S}^{(t)}$ whose diagonal elements are $\sigma_1, \ldots, \sigma_n$, and finally computing the consensus kernel $\mathbf{K}^{(t)} = \mathbf{U}_A\mathbf{S}^{(t)}\mathbf{U}_A^T$. By Theorem 2, $\mathbf{K}^{(t)}$ is the global optima of Eq. (8).

4.1.2. Optimize $\mathbf{K}_b^{(i)}$ by fixing $\mathbf{K}^{(t)}$

When $\mathbf{K}^{(t)}$ is fixed, Eq. (7) can be decoupled into m independent sub-problems each of which only involves one variable $\mathbf{K}_b^{(i)}$ and can be rewritten as:

$$\begin{aligned} \min_{\mathbf{K}_b^{(i)}} \quad & \|\mathbf{K}_b^{(i)} - \mathbf{B}\|_F^2 + \rho \; rank(\mathbf{K}_b^{(i)}), \\ s.t. \quad & \mathbf{K}_b^{(i)T} = \mathbf{K}_b^{(i)}, \quad \mathbf{K}_b^{(i)} \succeq 0. \end{aligned} \tag{10}$$

where $\mathbf{B} = \frac{\lambda_1\hat{\mathbf{K}}_b^{(i)} + \mathbf{K}^{(t)}}{\lambda_1 + 1}$ and $\rho = \frac{\lambda_2}{\lambda_1+1}$. Obviously, Eq. (10) can be solved in the same way as Eq. (8) by applying Theorem 2.

4.1.3. Algorithm for optimizing Eq. (7)

Algorithm 1 summarizes the above processes for optimizing Eq. (7), which jointly computes the consensus kernel $\mathbf{K}^{(t)}$ and the updated base kernels $\mathbf{K}_b^{(i)}$ ($1 \le i \le m$).

Algorithm 1 Learning a consensus kernel $\mathbf{K}^{(t)}$ ($t > m$) and updating the base kernels

Input: Current base kernels $\hat{\mathbf{K}}_b^{(i)}$ ($1 \le i \le m$), kernel $\mathbf{K}_c^{(t)}$ of a new view $\mathbf{X}^{(t)}$, parameters λ_1 and λ_2.

Output: Consensus kernel $\mathbf{K}^{(t)}$, and updated base kernels $\mathbf{K}_b^{(i)}$ ($1 \le i \le m$).

```
Initialize K_b^(i) = K̂_b^(i) (1 ≤ i ≤ m).
while not converge do
    Let A = 1/(m+1) (Σ_{i=1}^m K_b^(i) + K_c^(t)).
    Compute U_A and S_A such that A = U_A S_A U_A^T.
    Compute S^(t) using Eq. (9) and let K^(t) = U_A S^(t) U_A^T.
    for j = 1, 2, ..., m do
        Let B = (λ1 K̂_b^(j) + K^(t)) / (λ1 + 1).
        Compute U_B and S_B such that B = U_B S_B U_B^T.
        Compute S_b^(j) using Eq. (9) (similar to S^(t)) and let K_b^(j) = U_B S_b^(j) U_B^T.
    end for
end while
```

By Theorem 2, the consensus kernel $\mathbf{K}^{(t)}$ and the updated base kernels $\mathbf{K}_b^{(i)}$ $(1 \le i \le m)$ obtained in Algorithm 1 are the global optima of Eqs. (8) and (10), respectively. This means that in the minimization process via the **while** cycle (lines 3–10), the objective functions Eqs. (8) and (10) monotonically decrease and thus the objective function Eq. (7) monotonically decreases. As these objective functions are greater than 0, Algorithm 1 will always converge.

4.1.4. Learning an initial consensus kernel $\mathbf{K}^{(m)}$

It was mentioned earlier that we will learn an initial consensus spectral embedding $\mathbf{Y}^{(m)}$ by combining the m initial base kernels $\mathbf{K}_b^{(1)}, \mathbf{K}_b^{(2)}, \ldots, \mathbf{K}_b^{(m)}$ which are Gaussian kernels constructed from the first m views $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(m)}$, respectively. This requires us to learn an initial consensus kernel $\mathbf{K}^{(m)}$ from the m initial base kernels. We do this kernel combination by simplifying the objective function Eq. (7) to the following one:

$$\begin{aligned} &\min_{\mathbf{K}^{(m)}} \sum_{i=1}^{m} \|\mathbf{K}^{(m)} - \hat{\mathbf{K}}_b^{(i)}\|_F^2 + \lambda_2 \left(rank(\mathbf{K}^{(m)})\right), \\ &s.t. \quad \mathbf{K}^{(m)T} = \mathbf{K}^{(m)}, \quad \mathbf{K}^{(m)} \succeq 0 \end{aligned} \tag{11}$$

That is, we only require that the initial consensus kernel $\mathbf{K}^{(m)}$ be as close as possible to each initial base kernel $\mathbf{K}_b^{(i)}$, and be low rank. Eq. (11) can also easily be solved by Theorem 2.

4.2. Speedup strategies

It turns out that Algorithm 1 can be significantly improved both in space and time. Since it needs to store m n-by-n base kernels, the space complexity is $O(mn^2)$; it needs to do SVD operations on these kernels, so the time complexity is $O(l_1 mn^3)$, where l_1 is the number of iterations.

Recall that any n-by-n kernel matrix $\mathbf{K}$ can be represented as an economy SVD $\mathbf{USU}^T$, where $\mathbf{U} \in \mathbb{R}^{n\times r}$ and r is the rank of $\mathbf{K}$. In our method, we also impose the low rank property on the consensus kernel $\mathbf{K}^{(t)}$ and all base kernels $\mathbf{K}_b^{(i)}$ (i.e., $r \ll n$; see the objective function Eq. (7)). Therefore, in this section we present a way to improve the time and space complexity of Algorithm 1 by means of low rank kernel SVD decomposition.

Note that our initial m base kernels $\hat{\mathbf{K}}_b^{(1)}, \hat{\mathbf{K}}_b^{(2)}, \ldots, \hat{\mathbf{K}}_b^{(m)}$ and the current kernel $\mathbf{K}_c^{(t)}$ are Gaussian kernels constructed from the first m views $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(m)}$ and the current view $\mathbf{X}^{(t)}$ respectively. These Gaussian kernels are not necessarily low rank matrices. Therefore, we first use a low rank approximation of a Gaussian kernel by applying the *random Fourier features* [19] method introduced in Section 3.2.

In more details, according to Section 3.2, since all the kernels we use are Gaussian kernels which are properly scaled, we need to know the Fourier transformation of Gaussian kernel. It is easy to verify that the Fourier transformation of any Gaussian kernel function $k(\mathbf{x}-\mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|_2^2}{2\sigma^2}}$ is also a Gaussian distribution with zero means and $1/\sigma$ variances. Therefore, instead of directly constructing $\mathbf{K}$ ($\hat{\mathbf{K}}_b^{(i)}$ and $\mathbf{K}_c^{(t)}$) from $\mathbf{X}$ ($\mathbf{X}^{(i)}$ and $\mathbf{X}^{(t)}$), we can apply the above method of random Fourier features to learn an approximation $\mathbf{H}^T\mathbf{H}$ to $\mathbf{K}$, where $\mathbf{H} \in \mathbb{R}^{r\times n}$ is a low rank matrix obtained with Eqs.(5) and (6) by sampling $r/2$ vectors $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_{r/2}$ from the Gaussian distribution $\mathcal{N}(\mathbf{0}, (1/\sigma)\mathbf{I})$ with zero means and $1/\sigma$ variances. The time and space complexity of constructing $\mathbf{H}$ is $O(rdn)$ and $O(dn + rn)$ where d is the dimension of the data.

Let the SVD of $\mathbf{H}$ be $\mathbf{U}_H\mathbf{S}_H\mathbf{V}_H^T = \mathbf{H}$, which can be computed in $O(nr^2)$ time. Then the SVD of $\mathbf{H}^T\mathbf{H}$ is $\mathbf{H}^T\mathbf{H} = \mathbf{V}_H\mathbf{S}_H^2\mathbf{V}_H^T$, which can be taken as an approximation of the SVD of $\mathbf{K}$.

When all input kernels $\mathbf{K}_c^{(t)}$ and $\hat{\mathbf{K}}_b^{(i)}$ $(1 \le i \le m)$ are low rank, we can compute their SVD more efficiently. The most time-consuming parts of Algorithm 1 are to compute the SVD $\mathbf{A} = \mathbf{U}_A\mathbf{S}_A\mathbf{U}_A^T$ and $\mathbf{B} = \mathbf{U}_B\mathbf{S}_B\mathbf{U}_B^T$, where $\mathbf{A} = \frac{1}{m+1}\left(\sum_{i=1}^{m}\mathbf{K}_b^{(i)} + \mathbf{K}_c^{(t)}\right)$ and $\mathbf{B} = \frac{\lambda_1\hat{\mathbf{K}}_b^{(j)} + \mathbf{K}^{(t)}}{\lambda_1+1}$. Observe that the main components of $\mathbf{A}$ and $\mathbf{B}$ are the sum of kernels, so their SVD can be computed in an incremental way [46], i.e., we initially compute the SVD $\mathbf{E}$ of the sum of the first two kernels, then compute the SVD of the sum of $\mathbf{E}$ and the third kernel, and so on.

Consider two low rank symmetric and positive semi-definite matrices $\mathbf{K}_1 \in \mathbb{R}^{n\times n}$ and $\mathbf{K}_2 \in \mathbb{R}^{n\times n}$ whose SVD are $\mathbf{K}_1 = \mathbf{U}_1\mathbf{S}_1\mathbf{U}_1^T$ and $\mathbf{K}_2 = \mathbf{U}_2\mathbf{S}_2\mathbf{U}_2^T$, where $\mathbf{U}_1 \in \mathbb{R}^{n\times r_1}$, $\mathbf{S}_1 \in \mathbb{R}^{r_1\times r_1}$, $\mathbf{U}_2 \in \mathbb{R}^{n\times r_2}$, $\mathbf{S}_2 \in \mathbb{R}^{r_2\times r_2}$, and r_1 and r_2 are the rank of $\mathbf{K}_1$ and $\mathbf{K}_2$ respectively. We have

$$\mathbf{K}_1 + \mathbf{K}_2 = \mathbf{U}_1\mathbf{S}_1\mathbf{U}_1^T + \mathbf{U}_2\mathbf{S}_2\mathbf{U}_2^T = [\mathbf{U}_1, \mathbf{U}_2]\begin{bmatrix} \mathbf{S}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_2 \end{bmatrix}[\mathbf{U}_1, \mathbf{U}_2]^T \tag{12}$$

Let $\mathbf{F}$ be an orthogonal basis of the column space of $(\mathbf{I} - \mathbf{U}_1\mathbf{U}_1^T)\mathbf{U}_2$, which is the component of $\mathbf{U}_2$ that is orthogonal to $\mathbf{U}_1$. Define $\mathbf{F}_{U_2} = \mathbf{F}^T(\mathbf{I} - \mathbf{U}_1\mathbf{U}_1^T)\mathbf{U}_2$. So the number $\hat{r}$ of columns of $\mathbf{F}$ is equal to the number of rows of $\mathbf{F}_{U_2}$ and is also equal to the rank of $(\mathbf{I} - \mathbf{U}_1\mathbf{U}_1^T)\mathbf{U}_2$ which is no greater than r_2. So the time complexity of computing $\mathbf{F}$ and $\mathbf{F}_{U_2}$ is no greater than $O(nr_1r_2 + nr_2^2)$. Then we have

$$[\mathbf{U}_1, \mathbf{U}_2] = [\mathbf{U}_1, \mathbf{F}]\begin{bmatrix} \mathbf{I} & \mathbf{U}_1^T\mathbf{U}_2 \\ \mathbf{0} & \mathbf{F}_{U_2} \end{bmatrix} \tag{13}$$

Taking Eq. (13) back into Eq. (12), we obtain

$$\mathbf{K}_1 + \mathbf{K}_2 = [\mathbf{U}_1, \mathbf{F}]\boldsymbol{\Omega}[\mathbf{U}_1, \mathbf{F}]^T \tag{14}$$

where

$$\boldsymbol{\Omega} = \begin{bmatrix} \mathbf{I} & \mathbf{U}_1^T\mathbf{U}_2 \\ \mathbf{0} & \mathbf{F}_{U_2} \end{bmatrix}\begin{bmatrix} \mathbf{S}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_2 \end{bmatrix}\begin{bmatrix} \mathbf{I} & \mathbf{U}_1^T\mathbf{U}_2 \\ \mathbf{0} & \mathbf{F}_{U_2} \end{bmatrix}^T \tag{15}$$

Note that $\begin{bmatrix} \mathbf{I} & \mathbf{U}_1^T\mathbf{U}_2 \\ \mathbf{0} & \mathbf{F}_{U_2} \end{bmatrix}$ is a $(r_1 + \hat{r})$-by-$(r_1 + r_2)$ matrix, $\begin{bmatrix} \mathbf{S}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_2 \end{bmatrix}$ is a $(r_1 + r_2)$-by-$(r_1 + r_2)$ matrix and $\hat{r} \le r_2$, so the size of $\boldsymbol{\Omega}$ is no greater than $(r_1 + r_2) \times (r_1 + r_2)$.

Then we compute the SVD of $\boldsymbol{\Omega} = \hat{\mathbf{U}}\hat{\boldsymbol{\Omega}}\hat{\mathbf{U}}^T$ in $O((r_1+r_2)^3)$ time, and obtain the SVD of $\mathbf{K}_1 + \mathbf{K}_2$:

$$\mathbf{K}_1 + \mathbf{K}_2 = ([\mathbf{U}_1, \mathbf{F}]\hat{\mathbf{U}})\hat{\boldsymbol{\Omega}}(\hat{\mathbf{U}}^T[\mathbf{U}_1, \mathbf{F}]^T) \tag{16}$$

in $O(n(r_1 + r_2)^2)$ time.

So the SVD of $\mathbf{K}_1 + \mathbf{K}_2$ can be calculated in $O(n(r_1 + r_2)^2 + (r_1 + r_2)^3)$ time and $O(n(r_1 + r_2))$ space. Thus it takes $O(m(nr_{max}^2 + r_{max}^3))$ time and $O(mnr_{max})$ space to compute the low rank SVD of $\mathbf{A} = \frac{1}{m+1}\left(\sum_{i=1}^{m}\mathbf{K}_b^{(i)} + \mathbf{K}_c^{(t)}\right)$ and $\mathbf{B} = \frac{\lambda_1\hat{\mathbf{K}}_b^{(j)} + \mathbf{K}^{(t)}}{\lambda_1+1}$, where r_{max} stands for the largest rank of the involved kernel matrices. In contrast, in Algorithm 1 this needs $O(mn^3)$ time and $O(mn^2)$ space. Since $r_{max} \ll n$, applying the low rank SVD would significantly reduce the time and space complexity.

4.3. Learning a consensus spectral embedding $\mathbf{Y}^{(t)}$

After a consensus kernel $\mathbf{K}^{(t)}$ $(t \ge m)$ has been learned from the base kernels $\mathbf{K}_b^{(1)}, \ldots, \mathbf{K}_b^{(m)}$, we can construct a Laplacian matrix $\mathbf{L}^{(t)}$ from $\mathbf{K}^{(t)}$ as introduced in Section 3.1.

For $t = m$, given a Laplacian matrix $\mathbf{L}^{(m)}$ that is constructed from the initial consensus kernel $\mathbf{K}^{(m)}$, we learn an initial spectral embedding $\mathbf{Y}^{(m)}$ using the following loss function [40]:

$$\mathbf{Y}^{(m)} = \arg\min_{\hat{\mathbf{Y}}} tr(\hat{\mathbf{Y}}^T\mathbf{L}^{(m)}\hat{\mathbf{Y}}), \tag{17}$$

$$s.t. \quad \hat{\mathbf{Y}}^T\hat{\mathbf{Y}} = \mathbf{I}.$$

For $t > m$, let $\mathbf{L}^{(t)}$ be a Laplacian matrix constructed above and $\mathbf{Y}^{(t-1)}$ be a consensus spectral embedding of the last $t-1$ views $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(t-1)}$. We next describe how to learn a consensus spectral embedding $\mathbf{Y}^{(t)}$ of the t views $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(t-1)}, \mathbf{X}^{(t)}$ from $\mathbf{L}^{(t)}$ and $\mathbf{Y}^{(t-1)}$.

In our incremental multi-view learning, we expect the model to be stable in the sense that the spectral embedding $\mathbf{Y}^{(t)}$ of the first t views should be as close as possible to the embedding $\mathbf{Y}^{(t-1)}$ of the first $t-1$ views. This can be achieved by adding a smooth term $\|\hat{\mathbf{Y}} - \mathbf{Y}^{(t-1)}\|_F^2$ to Eq. (17), i.e.,

$$\mathbf{Y}^{(t)} = \arg\min_{\hat{\mathbf{Y}}} tr(\hat{\mathbf{Y}}^T\mathbf{L}^{(t)}\hat{\mathbf{Y}}) + \gamma\|\hat{\mathbf{Y}} - \mathbf{Y}^{(t-1)}\|_F^2, \tag{18}$$
$$s.t. \quad \hat{\mathbf{Y}}^T\hat{\mathbf{Y}} = \mathbf{I}.$$

where γ is a balancing parameter.

Since Eq. (18) contains an orthogonal constraint, it is hard to find a closed form solution. Inspired by [47,48], we propose a constraint preserving descent method to solve this optimization problem. This method guarantees that in each iteration, the orthogonal constraint is preserved and the loss function Eq. (18) decreases.

We first rewrite Eq. (18) in a simpler form:

$$\min_{\hat{\mathbf{Y}}} \quad tr(\hat{\mathbf{Y}}^T\mathbf{P}\hat{\mathbf{Y}}) + 2tr(\hat{\mathbf{Y}}^T\mathbf{Q})$$
$$s.t. \quad \hat{\mathbf{Y}}^T\hat{\mathbf{Y}} = \mathbf{I} \tag{19}$$

where $\mathbf{P} = \mathbf{L}^{(t)} + \gamma\mathbf{I}$ and $\mathbf{Q} = -\gamma\mathbf{Y}^{(t-1)}$. By introducing the Lagrangian multiplier $\mathbf{\Lambda}$, we get the Lagrangian function

$$\mathcal{L} = tr(\hat{\mathbf{Y}}^T\mathbf{P}\hat{\mathbf{Y}}) + 2tr(\hat{\mathbf{Y}}^T\mathbf{Q}) - tr(\mathbf{\Lambda}(\hat{\mathbf{Y}}^T\hat{\mathbf{Y}} - \mathbf{I})). \tag{20}$$

Set the partial derivative w.r.t. $\hat{\mathbf{Y}}$ to zero:

$$\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{Y}}} = 2\left(\mathbf{P}\hat{\mathbf{Y}} + \mathbf{Q} - \hat{\mathbf{Y}}\mathbf{\Lambda}\right) = \mathbf{0} \tag{21}$$

By multiplying both sides of Eq. (21) by $\hat{\mathbf{Y}}^T$ and applying the constraint $\hat{\mathbf{Y}}^T\hat{\mathbf{Y}} = \mathbf{I}$, we can solve $\mathbf{\Lambda}$ as $\mathbf{\Lambda} = \hat{\mathbf{Y}}^T(\mathbf{P}\hat{\mathbf{Y}} + \mathbf{Q})$. Note that $\hat{\mathbf{Y}}^T\hat{\mathbf{Y}}$ is symmetric, and its corresponding Lagrangian multiplier $\mathbf{\Lambda}$ is also symmetric. So we rewrite $\mathbf{\Lambda}$ as $\mathbf{\Lambda} = (\mathbf{P}\hat{\mathbf{Y}} + \mathbf{Q})^T\hat{\mathbf{Y}}$. Putting it back into Eq. (21), we obtain

$$\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{Y}}} = 2\left(\mathbf{P}\hat{\mathbf{Y}}\hat{\mathbf{Y}}^T + \mathbf{Q}\hat{\mathbf{Y}}^T - \hat{\mathbf{Y}}(\mathbf{P}\hat{\mathbf{Y}} + \mathbf{Q})^T\right)\hat{\mathbf{Y}} \tag{22}$$

Let $\mathbf{R} = 2\left(\mathbf{P}\hat{\mathbf{Y}}\hat{\mathbf{Y}}^T + \mathbf{Q}\hat{\mathbf{Y}}^T - \hat{\mathbf{Y}}(\mathbf{P}\hat{\mathbf{Y}} + \mathbf{Q})^T\right)$. It is easy to verify that $\mathbf{R}$ is a skew-symmetric matrix, i.e., $\mathbf{R}^T = -\mathbf{R}$.

Let $\hat{\mathbf{Y}}^l$ be the result of $\hat{\mathbf{Y}}$ at the lth iteration in the descent process. At the $(l+1)$th iteration , we want to learn $\hat{\mathbf{Y}}^{l+1}$ from $\hat{\mathbf{Y}}^l$ with an iteration step size $\eta \geq 0$.

The following theorem provides a descent update formula and shows that the orthogonal constraint is preserved and the loss function decreases.

Theorem 3. *(1) Given any skew-symmetric matrix $\mathbf{R} \in \mathbb{R}^{n\times n}$ and $\hat{\mathbf{Y}}^l \in \mathbb{R}^{n\times c}$ such that $\hat{\mathbf{Y}}^{lT}\hat{\mathbf{Y}}^l = \mathbf{I}$, let*

$$\hat{\mathbf{Y}}^{l+1} = \left(\mathbf{I} + \frac{\eta}{2}\mathbf{R}\right)^{-1}\left(\mathbf{I} - \frac{\eta}{2}\mathbf{R}\right)\hat{\mathbf{Y}}^l \tag{23}$$

Then $(\hat{\mathbf{Y}}^{l+1})^T\hat{\mathbf{Y}}^{l+1} = \mathbf{I}$.

(2) Let $\mathbf{R} = 2\left(\mathbf{P}\hat{\mathbf{Y}}^l\hat{\mathbf{Y}}^{lT} + \mathbf{Q}\hat{\mathbf{Y}}^{lT} - \hat{\mathbf{Y}}^l(\mathbf{P}\hat{\mathbf{Y}}^l + \mathbf{Q})^T\right)$. Then updating $\hat{\mathbf{Y}}^{l+1}$ is in a descent direction. Since the objective function in Eq. (19) is lower bounded by 0, the iteration method converges. Moreover, it can converge to a stable point.

Proof. See Appendix.

Note that we choose the iteration step size η by a curvilinear search method as was done in [48], which can guarantee the convergence. Therefore we compute $\mathbf{Y}^{(t)}$ iteratively using the update formula Eq. (23) until the decent process converges. Clearly, the most expensive part here is to compute the inverse $(\mathbf{I} + \frac{\eta}{2}\mathbf{R})^{-1}$ which takes $O(n^3)$ time. Fortunately, we can speedup this process using low rank decomposition technique. In more details, we can rewrite $\frac{\eta}{2}\mathbf{R} = \mathbf{MN}$, where $\mathbf{M} = [\frac{\eta}{2}(\mathbf{P}\hat{\mathbf{Y}}^l + \mathbf{Q}), -\frac{\eta}{2}\hat{\mathbf{Y}}^l]$ and $\mathbf{N} = [\hat{\mathbf{Y}}^l, \mathbf{P}\hat{\mathbf{Y}}^l + \mathbf{Q}]^T$. By borrowing the result in [49], we have $(\mathbf{I} + \mathbf{MN})^{-1} = \mathbf{I} - \mathbf{M}(\mathbf{I} + \mathbf{NM})^{-1}\mathbf{N}$. Note that $\mathbf{I} + \mathbf{NM}$ is a $2c$-by-$2c$ matrix, where c is the number of clusters and often $c \ll n$. So the time complexity of the matrix inverse is reduced from $O(n^3)$ to $O(c^3)$. According to this, we analyze the time and space complexity of this step. Taking $\frac{\eta}{2}\mathbf{R} = \mathbf{MN}$ into Eq. (23), we obtain:

$$\begin{aligned}\hat{\mathbf{Y}}^{l+1} &= \left(\mathbf{I} + \frac{\eta}{2}\mathbf{R}\right)^{-1}\left(\mathbf{I} - \frac{\eta}{2}\mathbf{R}\right)\hat{\mathbf{Y}}^l \\ &= (\mathbf{I} + \mathbf{MN})^{-1}(\mathbf{I} - \mathbf{MN})\hat{\mathbf{Y}}^l \\ &= \left(\mathbf{I} - \mathbf{M}(\mathbf{I} + \mathbf{NM})^{-1}\mathbf{N}\right)(\mathbf{I} - \mathbf{MN})\hat{\mathbf{Y}}^l \\ &= \hat{\mathbf{Y}}^l - \mathbf{MN}\hat{\mathbf{Y}}^l - \mathbf{M}(\mathbf{I} + \mathbf{NM})^{-1}\mathbf{N}\hat{\mathbf{Y}}^l + \mathbf{M}(\mathbf{I} + \mathbf{NM})^{-1}\mathbf{NMN}\hat{\mathbf{Y}}^l\end{aligned} \tag{24}$$

Note that $\mathbf{M}$, $\mathbf{N}$ and $\mathbf{Y}^l$ are n-by-$2c$, $2c$-by-n and n-by-c matrices, respectively. So the time complexity of computing $\mathbf{MN}\hat{\mathbf{Y}}^l$, $\mathbf{M}(\mathbf{I} + \mathbf{NM})^{-1}\mathbf{N}\hat{\mathbf{Y}}^l$ and $\mathbf{M}(\mathbf{I} + \mathbf{NM})^{-1}\mathbf{NMN}\hat{\mathbf{Y}}^l$ are $O(nc^2)$, $O(nc^2 + c^3)$ and $O(nc^2 + c^3)$, respectively, by using the associative law of multiplication. In addition, the time complexity of computing $\mathbf{P}\hat{\mathbf{Y}}^l$ is $O(nkc)$ because $\mathbf{P}$ is a sparse matrix and there are only $O(k)$ non-zero elements in each row. Thus in this step, the whole time complexity is $O(l(nkc + nc^2 + c^3))$, where l is the number of iterations, and the space complexity is $O(nc+nk)$ because we only use n-by-$2c$, n-by-c and sparse n-by-n matrices. Summarizing the above subsections, we obtain the whole process of IMSC as shown in Algorithm 2.

Algorithm 2 Incremental Multi-view Spectral Clustering

Input: A multi-view data set $\mathcal{X} = \{\mathbf{X}^{(1)}, \cdots, \mathbf{X}^{(v)}\}$, balancing parameters λ_1, λ_2, γ, number $m \leq v$ of base kernels, number c of clusters, the bandwidth parameter σ, number r of samples in random Fourier feature step.

Output: Clustering result $\mathbf{C}^{(v)}$ of $\mathcal{X}$.

1: **for** $t = 1, 2, \cdots, m$ **do**
2: Construct from $\mathbf{X}^{(t)}$ a low rank approximation $\mathbf{U}_b^{(t)}\mathbf{S}_b^{(t)}\mathbf{U}_b^{(t)T}$ to the SVD of the initial base kernel $\mathbf{K}_b^{(t)}$ by random Fourier features method.
3: **end for**
4: Learn an initial consensus kernel $\mathbf{K}^{(m)}$ (see Section 4.1.4).
5: Construct a Laplacian matrix $\mathbf{L}^{(m)}$ from $\mathbf{K}^{(m)}$ and learn an initial consensus spectral embedding $\mathbf{Y}^{(m)}$ from $\mathbf{L}^{(m)}$ (see Section 4.3).
6: **for** $t = m+1, m+2, \cdots, v$ **do**
7: Learn a consensus kernel $\mathbf{K}^{(t)}$ and update the m base kernels (see Algorithm 1 and Section 4.2).
8: Construct a Laplacian matrix $\mathbf{L}^{(t)}$ from $\mathbf{K}^{(t)}$ and learn a consensus spectral embedding $\mathbf{Y}^{(t)}$ from $\mathbf{L}^{(t)}$ (see Section 4.3).
9: **end for**
10: Compute the clustering result $\mathbf{C}^{(v)}$ by means of spectral rotation of $\mathbf{Y}^{(v)}$ [42].

4.4. Time and space complexity

In this subsection, we discuss the time and space complexity of Algorithm 2. In Lines 1–3, we construct the low rank approximation of the first m views by random Fourier feature methods. This step costs $O(mrd_{max}n)$ time and $O(d_{max}n + mrn)$ space, where d_{max} is the highest number of dimension of all the views. Then

learning the initial kernel $\mathbf{K}^{(m)}$, the consensus $\mathbf{K}^{(t)}$ and updating the base kernels needs $O(m(nr_{max}^2 + r_{max}^3))$ time and $O(mnr_{max})$ space as discussed in Section 4.2).

When constructing Laplacian matrix $\mathbf{L}^{(t)}$, we need to construct the k-nn graph $\mathbf{W}^{(t)}$. Since each row of $\mathbf{W}^{(t)}$ is de-coupled, we can calculate $\mathbf{W}^{(t)}$ row by row which needs $O(nk)$ space and $O(n^2k)$ time. If n is too large, we can apply a probabilistic speedup strategy to find an approximate k-nn set of each row as introduced in [50]. In more details, when finding the largest element in a row, we can sample a random subset of fixed size, say κ, and pick the largest one in this subset rather than performing an exhaustive search over all n elements. According to [50], $\kappa = 59$ can already guarantee nearly as good performance as if we consider all n elements. Thus, the time complexity can be reduced to $O(nk\kappa)$ from $O(n^2k)$.

At last, the time and space complexity of updating $\mathbf{Y}$ is $O(l(nkc + nc^2 + c^3))$ and $O(nc + nk)$ as introduced in 4.3.

4.5. Handling new data and missing data

In the streaming view setting, it often happens that there are a small quantity of new data or missing data in the new view. In traditional multi-view spectral clustering methods, it is hard to handle this incomplete data setting. Fortunately, in our method, since we compute the low rank representation of kernels $\mathbf{USU}^T$, we can compute $\mathbf{H} = \mathbf{S}^{0.5}\mathbf{U}^T$ as the embedding of the instances in the kernel space. When we obtain this embedding, we can solve this problem naturally.

In more details, the key problem is that when optimizing Eq. (7), $\hat{\mathbf{K}}_b^{(i)}$ and $\mathbf{K}_c^{(t)}$ are not aligned, i.e., some instances in $\hat{\mathbf{K}}_b^{(i)}$ are absence in $\mathbf{K}_c^{(t)}$ and vice versa. A natural way to handle this problem is filling $\hat{\mathbf{K}}_b^{(i)}$ and $\mathbf{K}_c^{(t)}$ before optimizing Eq. (7). Let $\hat{\mathbf{H}}_b^{(i)} = \hat{\mathbf{S}}_b^{(i)0.5}\hat{\mathbf{U}}_b^{(i)T}$, $\mathbf{H}_c^{(t)} = \mathbf{S}_c^{(t)0.5}\mathbf{U}_c^{(t)T}$ and $\mathbf{H}^{(t-1)} = \mathbf{S}^{(t-1)0.5}\mathbf{U}^{(t-1)T}$ denote the embedding of $\hat{\mathbf{K}}_b^{(i)}$, $\mathbf{K}_c^{(t)}$ and $\mathbf{K}^{(t-1)}$ respectively, where $\mathbf{K}^{(t-1)}$ is the consensus kernel of the first $t-1$ views. Note that $\hat{\mathbf{H}}_b^{(i)}$ and $\mathbf{H}^{(t-1)}$ are aligned. For an instance $\mathbf{x}$ which is in the $\hat{\mathbf{H}}_b^{(i)}$ (or $\mathbf{H}^{(t-1)}$) whereas is absence in $\mathbf{H}_c^{(t)}$, we denote $\mathbf{h}_x^{(t-1)}$ as the corresponding column of $\mathbf{x}$ in $\mathbf{H}^{(t-1)}$. We first find its k-nn instances in the common instances (appearing in $\mathbf{H}^{(t-1)}$ and $\mathbf{H}_c^{(t)}$ both), and learn the linear coefficient such that $\mathbf{h}_x^{(t-1)} \approx \Sigma_{j=1}^k w_j \mathbf{h}_j^{(t-1)}$ where $\mathbf{h}_j^{(t-1)}$ is a k-nn instance of $\mathbf{x}$ represented in $\mathbf{H}^{(t-1)}$ and w_j is the linear coefficient. We can learn the weight w_j by a similar way in Locally Linear Embedding (LLE) [51]:

$$\min_{\mathbf{w}} \left\| \mathbf{h}_x^{(t-1)} - \sum_{j=1}^{k} w_j \mathbf{h}_j^{(t-1)} \right\|_2^2 + \frac{\alpha}{2} \|\mathbf{w}\|_2^2, \quad s.t. \sum_{j=1}^{k} w_j = 1. \tag{25}$$

Then in $\mathbf{H}_c^{(t)}$, we reconstruct the corresponding instance by $\mathbf{h}_{x\_c}^{(t)} = \Sigma_{j=1}^k w_j \mathbf{h}_{j\_c}^{(t)}$ where $\mathbf{h}_{x\_c}^{(t)}$ is the estimated $\mathbf{x}$ in $\mathbf{H}_c^{(t)}$ and $\mathbf{h}_{j\_c}^{(t)}$ is the k-nn instance of $\mathbf{x}$ represented in $\mathbf{H}_c^{(t)}$. For the instances that appear in $\mathbf{H}_c^{(t)}$ whereas are absence in $\hat{\mathbf{H}}_b^{(i)}$, we can handle them similarly. Moreover, we can also use the similar technique to fill $\mathbf{Y}^{(t-1)}$ in Eq. (18) to handle new data.

5. Experiments

In this section, we empirically evaluate the effectiveness and efficiency of our incremental multi-view method IMSC on benchmark data sets.

Table 1
Five benchmark multi-view data sets.

	#instances	#features	#classes
Corel	3400	64, 9, 128, 10, 8, 104, 15	34
UCI Digit	2000	216, 76, 64, 6, 240, 47	10
AwA	30475	2688, 40960, 2000, 252, 2000, 2000, 2000, 4096	50
Sun397	39700	6300, 784, 256, 512, 512 512, 6300, 1239, 798, 230 2000,6300,10752,3072	397
Gas Sensor	17922	72 features × 100 views	11
CM	2205	8 features × 60 views	3

5.1. Data set descriptions

Our experiments were conducted on five benchmark multi-view data sets as summarized in Table 1. These data sets include widely used multi-view data, such as Corel data set[1] [52], UCI Digit data set[2] [53], AwA (Animal with Attributes) data set[3] [54] and Sun397 data set[4] [55]. To simulate the streaming view setting, we also use two time-series data sets Gas Sensor[5] [18] and Condition Monitoring of hydraulic systems (CM)[6] [56]. Gas Sensor data set contains time-series measurement recordings collected from 72 metal-oxide gas sensors utilized in the identification of potentially-dangerous chemical gaseous. Since the data set is a time-series recording, we sample 100 views from 100 time points, and in each view the recordings of the 72 sensors are the features. By this way, we obtain a 100-view data set. Similarly, CM is experimentally obtained with a hydraulic test rig which contains 2205 instances. It is also a time-series recording, and we sample 60 views from 60 time points, and in each view the features are the records sampled from 8 sensors.

5.2. Baseline methods

We compare our IMSC with the following baseline methods:

- Single view spectral clustering (SC): at time t we do standard single view spectral clustering [40] only on the tth view without using any other views.
- CoregSC [4]: it is a coregularization based multi-view spectral clustering method.
- RMSC [7]: it is a robust multi-view spectral clustering method by building a Markov chain.
- AMGL [31]: it is a parameter-free multi-view spectral clustering method which learns the weights of the views automatically.
- AWP [34]: it learns the spectral embedding from each view and combined them via adaptively weighted procrustes.
- MCGC [57]: it learns the spectral embedding from a consensus graph with minimizing disagreement between different views.

For the multi-view methods, at time t, we apply them on all of the available t views. For clustering results, single view SC is a

1 http://www.cs.virginia.edu/%7exj3a/research/CBIR/Download.htm.
2 http://archive.ics.uci.edu/ml/datasets/Multiple+Features.
3 http://attributes.kyb.tuebingen.mpg.de/.
4 http://vision.cs.princeton.edu/projects/2010/SUN/.
5 http://archive.ics.uci.edu/ml/datasets/Gas+sensor+arrays+in+open+sampling+settings.
6 https://archive.ics.uci.edu/ml/datasets/Condition+monitoring+of+hydraulic+systems.

weak baseline as it uses only one view; and multi-view methods are five strong baselines as they make use of all views. In the experiments, we expect that our IMSC outperforms single view SC and is close to the multi-view methods in clustering results, and that IMSC runs significantly faster than them.

5.3. Experimental setup

In our experiments on all data sets, we choose $m = 2$, i.e., we use only the first 2 views to construct two base kernels and start incremental multi-view clustering with the third view. When the tth view ($t > 2$) is available, we run our method and the baseline methods and report the clustering results and the running time.

Due to applying random Fourier features to construct low rank approximation to Gaussian kernels, in our method we do random sampling from the Gaussian distribution $\mathcal{N}(\mathbf{0}, (1/\sigma)\mathbf{I})$. Therefore, we run our method 20 times (with 20 different samplings) and report the average results. We set the bandwidth parameter σ to the median of the pair-wise Euclidean distances of all instances in the same way as in [4], and set the number c of clusters to the true number of classes for all data sets and all methods. We fix the sample number $r = 100$ for random Fourier features and set the k-nn parameter k to 10 on all data sets. We tune the balancing parameters λ_1, λ_2 and γ by grid search in the range $[10^{-4}, 10^4]$. We also tune the parameters in baseline methods as suggested in their papers. Two clustering evaluation metrics are adopted to measure the clustering performance, including clustering Accuracy (ACC) and Normalized Mutual Information (NMI). Higher quality results will have higher ACC and NMI values.

All experiments were conducted using Matlab on a PC computer with Windows 7, 3.4 GHz CPU and 32 GB memory.

5.4. Experimental results on clustering quality

Fig. 1 shows the ACC and NMI results of IMSC and the baseline methods on the benchmark multi-view data sets. Note that due to their high space complexity, CoregSC, RMSC, AMGL, AWP and MCGC yield no clustering results on the two larger data sets AwA and Sun397, because they run out of memory. On the Gas Sensor data set, RMSC and MCGC suffers the out-of-memory error; CoregSC, AMGL and AWP can only handle 3, 8 and 7 views respectively and when one more view arrives, they also suffer the out-of-memory error. It demonstrates that those multi-view methods which collect all views to learn a consensus result is infeasible to handle the data which contains a large number of views. Since our method is in an incremental scheme, the memory it uses is independent of the number of views, and it can always work no matter how many views there are.

We see that the ACC and NMI values of our incremental method IMSC keep growing with the increase of the number of views on most data sets. This indicates that using a new view $\mathbf{X}^{(t)}$ to update the current model (2 base kernels $\mathbf{K}_b^{(1)}$ and $\mathbf{K}_b^{(2)}$, and a spectral embedding $\mathbf{Y}^{(t-1)}$) in our method can indeed improve the performance of the model. The clustering results in the experiments clearly demonstrate that our method significantly outperforms the single view SC and is comparable with or even better than the 5 state-of-the-art multi-view baselines CoregSC, RMSC, AMGL, AWP and MCGC.

5.5. Experimental results on efficiency

In the experiments, for each data set $\{\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(v)}\}$, IMSC works in an incremental way and its running time consists of two parts: (1) initially it constructs from the first 2 views a model, i.e., 2 base kernels and a spectral embedding $\mathbf{Y}^{(2)}$; (2) then for each new view $\mathbf{X}^{(t)}$ ($t \geq 3$), it learns a consensus clustering result $\mathbf{C}^{(t)}$. Obviously, in the streaming view setting, what we are more interested in is the second part; in particular we expect that IMSC responds fast whenever a new view is available. Therefore, for each new view $\mathbf{X}^{(t)}$ ($t > 2$) we report the second part of the running time of IMSC and ALL time of IMSC used to process all of the v views. Tables 2–5 show the experimental results, where (1) for each $t > 2$ we report the time of IMSC used for learning the clustering result $\mathbf{C}^{(t)}$ from the current model and the view $\mathbf{X}^{(t)}$, the time of multi-view methods for learning $\mathbf{C}^{(t)}$ from the first t views, and the time of the single view SC for learning $\mathbf{C}^{(t)}$ from the tth view; and (2) we also report ALL time of IMSC used to process all of the v views. Note that in Tables 4 and 5 on the data sets AwA and Sun397, CoregSC, RMSC, AMGL, AWP and MCGC run out of memory without outputs. Fig. 2 shows the running time of our method and baseline methods on streaming view data sets, i.e., Gas Sensor and CM data sets.

As expected, the experimental results show that our incremental method is significantly faster than the multi-view methods on all data sets which demonstrates that our method is more feasible to the streaming view setting than those multi-view methods. We further stress the following two major reasons: (1) Once the initial model is built, for any new view $\mathbf{X}^{(t)}$ ($t > 2$), our incremental method works constantly by combining only

Table 2
Running time (seconds) on UCI Digit.

Algorithms	Views				
	3	4	5	6	ALL
IMSC	2.15	2.85	3.12	3.70	12.57
CoregSC	15.79	18.40	21.14	23.54	23.54
RMSC	410.75	465.23	526.91	431.21	431.21
AMGL	10.86	11.18	13.55	14.94	14.94
AWP	4.23	11.82	13.36	14.59	14.59
MCGC	54.40	74.29	90.93	98.76	98.76
Single view SC	0.23	0.84	0.28	0.24	–

Table 3
Running time (seconds) on Corel.

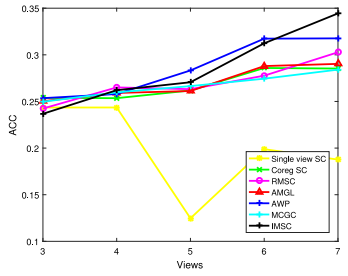
Algorithms	Views					
	3	4	5	6	7	ALL
IMSC	8.09	11.93	13.60	18.15	23.13	78.12
CoregSC	81.28	93.28	105.03	119.07	134.56	134.56
RMSC	1976.6	2145.9	2141.1	2156.6	2186.1	2186.1
AMGL	70.31	74.31	80.01	88.15	90.47	90.47
AWP	23.73	38.03	52.45	67.08	81.77	81.77
MCGC	358.23	453.80	515.78	600.07	692.89	692.89
Single view SC	0.62	0.54	0.55	0.58	0.57	–

Table 4
Running time (seconds) on AwA.

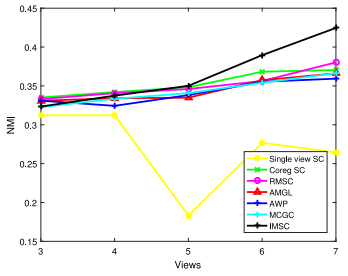
Algorithms	Views					
	3	4	5	6	7	8
IMSC	123.52	145.30	142.50	145.37	185.71	191.5
Single view SC	99.22	45.23	96.28	95.14	97.22	161.75

Table 5
Running time (seconds) on Sun397.

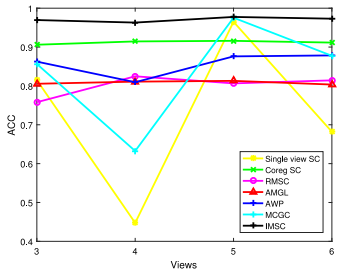
Algorithms	Views					
	3	4	5	6	7	8
IMSC	1372.3	1223.4	1412.1	1452.9	2438.3	1321.1
Single view SC	285.30	649.63	714.42	671.90	1007.0	633.00
Algorithms	Views					
	9	10	11	12	13	14
IMSC	1542.1	1498.3	1622.1	1575.3	1531.2	1634.1
Single view SC	675.36	675.25	825.48	916.02	1142.5	636.79



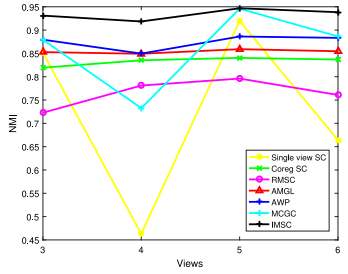
(a) ACC result of all methods on Corel data set



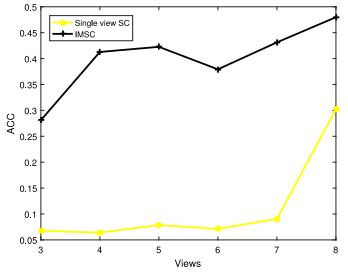
(b) NMI result of all methods on Corel data set



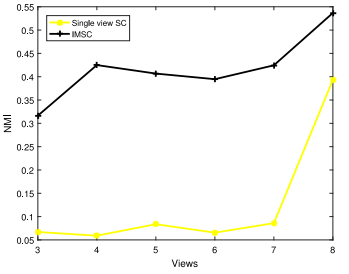
(c) ACC result of all methods on UCI Digit data set



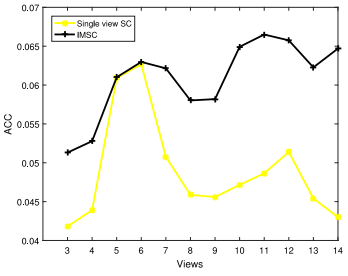
(d) NMI result of all methods on UCI Digit data set



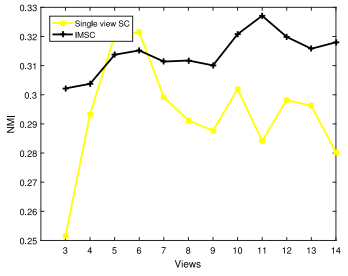
(e) ACC result of all methods on AwA data set



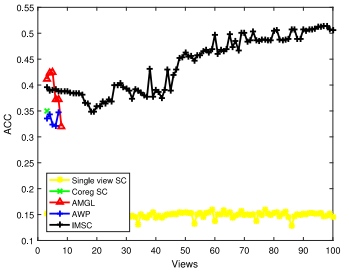
(f) NMI result of all methods on AwA data set



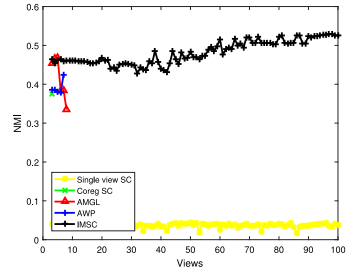
(g) ACC result of all methods on Sun397 data set



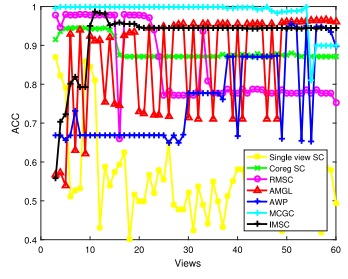
(h) NMI result of all methods on Sun397 data set



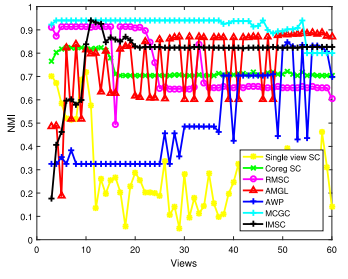
(i) ACC result of all methods on Gas Sensor data set



(j) NMI result of all methods on Gas Sensor data set

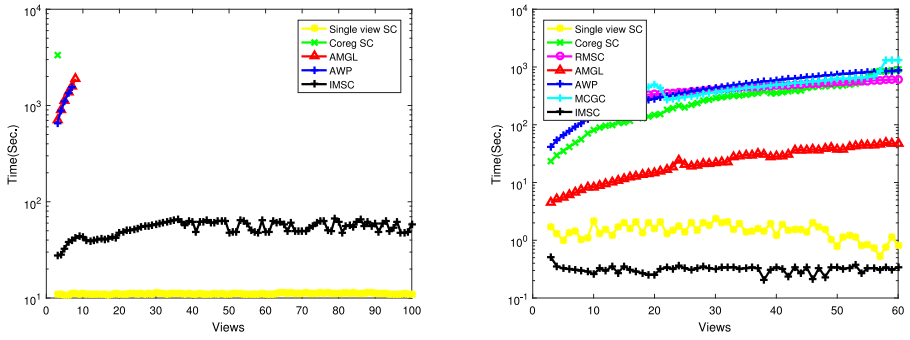


(k) ACC result of all methods on CM data set



(l) NMI result of all methods on CM data set

Fig. 1. ACC and NMI on multi-view data sets.



(a) Running time on Gas Sensor data set (b) Running time on CM data set

Fig. 2. Running time on streaming view data sets.

Table 6
Clustering results of IMSC under 5 different order of views.

Data sets	Metric	Different view orders				
		1	2	3	4	5
Corel	ACC	0.3213	0.3267	0.3541	0.3201	0.3312
	NMI	0.4076	0.4070	0.4200	0.4122	0.4079
UCI Digit	ACC	0.9612	0.9597	0.9761	0.9751	0.9573
	NMI	0.9132	0.9117	0.9477	0.9429	0.9077
AwA	ACC	0.4913	0.4871	0.4940	0.4969	0.4372
	NMI	0.5222	0.5381	0.5253	0.5397	0.4513
Sun397	ACC	0.0663	0.0601	0.0713	0.0602	0.0648
	NMI	0.3270	0.3100	0.3292	0.3150	0.3186
Gas Sensor	ACC	0.4910	0.4214	0.4542	0.4762	0.5057
	NMI	0.5390	0.4866	0.5001	0.5056	0.5255
CM	ACC	0.9447	0.8916	0.8957	0.8789	0.9420
	NMI	0.8274	0.7460	0.7569	0.7186	0.8171

Table 7
Clustering results of IMSC with 4 different number of base kernels.

Data sets	Metric	Number of base kernels			
		1	2	3	4
Corel	ACC	0.3035	0.3312	0.3481	0.3444
	NMI	0.3850	0.4079	0.4307	0.4272
UCI Digit	ACC	0.9568	0.9761	0.9723	0.9721
	NMI	0.9071	0.9477	0.9362	0.9348
AwA	ACC	0.1619	0.4918	0.3782	0.4212
	NMI	0.1872	0.5222	0.4161	0.4613
Sun397	ACC	0.0551	0.0663	0.0701	0.0694
	NMI	0.3017	0.3270	0.3319	0.3316
Gas Sensor	ACC	0.4551	0.5057	0.4111	0.5017
	NMI	0.5259	0.5255	0.4501	0.5470
CM	ACC	0.8136	0.9420	0.8503	0.7583
	NMI	0.5885	0.8171	0.7876	0.5605

3 kernels (2 base kernels and the current kernel), whereas the compared methods need to combine all t views/kernels, which would be quite costly when t gets larger. As a result, they run out of memory on the data sets Gas Sensor, AwA and Sun397. (2) the compared methods do SVD on n-by-n kernel or Laplacian matrices. In contrast, our method makes n-by-r low rank approximation to n-by-n Gaussian kernels by means of random Fourier features and does low rank SVD. This significantly saves both space and time since $r \ll n$.

5.6. Evaluation on sensitivity to the order of views

An important property of our incremental multi-view clustering method is that it is not sensitive to the order of views. We have empirically evaluated this property by randomly shuffling the views of each data set 5 times. Table 6 shows the experimental clustering results of each data set under 5 different order of views. It is easy to check that the 5 results in each row are similar, meaning that our incremental method produces similar clustering results under whatever order of views.

5.7. Evaluation on different number of base kernels

Another important property of our incremental method is that it needs to keep only a very few number of base kernels in the model. We have empirically tested this property on the benchmark multi-view data sets by choosing 1 to 4 base kernels, respectively. Table 7 shows the experimental clustering results. We see that it makes no big difference to choose 2, 3 or 4 base kernels. Therefore, we chose only 2 base kernels in our experiments.

5.8. Handling new data and missing data

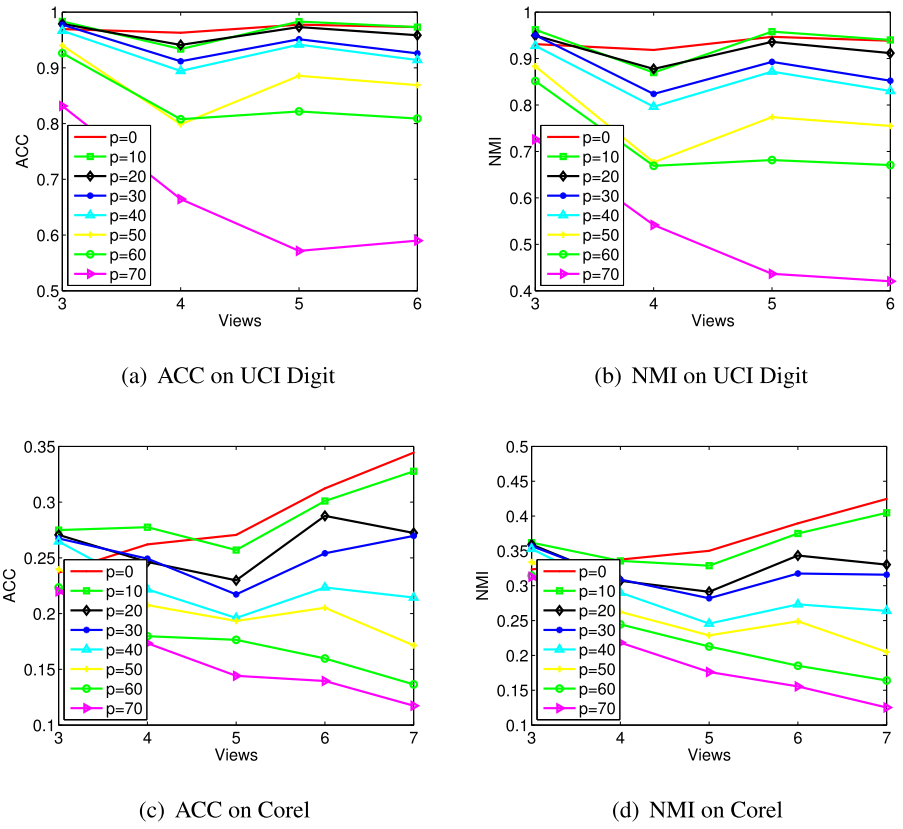
In this subsection, we evaluate the ability of our method to handle new data and missing data. Given a multi-view data set, in the first two views, we randomly remove p% of data as missing data. Then, in following each view, we add $p/(v-2)$% of data as new data and simultaneously remove $p/(v-2)$% of data as missing data, where v is the number of views. Thus in each view, the ratio of missing data is p%. We show the ACC and NMI on UCI Digit and Corel data sets in Fig. 3. The results on other data sets are similar.

In Fig. 3, we show the results of missing data from 10% to 70%. Note that $p = 0$ means that the data is complete without missing data. As expected, the clustering performance is degraded with the increasing of missing data. Despite all this, the clustering performance of our method on the data whose ratio of missing data is lower than 30% is close to it on the complete data. It demonstrates that our method can handle the case with a small quantity of missing data and new data well.

5.9. Parameter study

We study the effects of parameters λ_1, λ_2 and γ by tuning them in the range $\{10^{-4}, 10^{-2}, 10^{0}, 10^{2}, 10^{4}\}$. The experimental results on the Corel and UCI Digit data sets are shown in Fig. 4. Experiments on the other data sets have similar results.

We see that the parameters can be chosen in a wide range; it is suitable to choose $\lambda_1 \geq 1$, $\lambda_2 \leq 100$ and $\gamma \leq 1$. λ_2 is used to tune the rank of kernels. If λ_2 is too large (greater than 100), the rank of the learned kernels will be too small, so that the



(a) ACC on UCI Digit (b) NMI on UCI Digit

(c) ACC on Corel (d) NMI on Corel

Fig. 3. Clustering performance in the missing data setting.

kernels may be like an all-zero matrix, which will lead to worse clustering performance. Similarly, γ is used to balance the effect of the previous views and the new view. When γ is greater than 1, the performance will not get further improved by the new view.

6. Conclusion

In this paper, we introduced a novel incremental multi-view spectral clustering method. We learned a model from a few views, and when a new view was available, we updated the model with the new view and learned a consensus result by applying the updated model. This essentially differs from the existing multi-view learning methods, which learn a consensus result over the collection of all views. To further reduce the time and space complexity, we applied low rank approximation by means of random Fourier features to construct the base kernels and did low rank SVD accordingly. Our extensive experimental study on benchmark multi-view data sets demonstrated that while being comparable in clustering quality, IMSC was significantly faster than the state-of-the-art multi-view spectral clustering methods.

As future work, we are considering applying the idea to the task of incremental multi-view classification, i.e., we learn a classifier from multi-view data in an incremental way.

Acknowledgments

This work is supported by the National Natural Science Fund of China grants 61806003, and 61502289, the China National 973 program 2014CB340301, and the Key Natural Science Project of Anhui Provincial Education Department KJ2018A0010 and KJ2018A0011.

Appendix. Proofs

Proof of Theorem 2. We have:

$$\|\mathbf{K}^{(t)} - \mathbf{A}\|_F^2 = tr(\mathbf{K}^{(t)T}\mathbf{K}^{(t)}) - 2tr(\mathbf{K}^{(t)}\mathbf{A}^T) + tr(\mathbf{A}\mathbf{A}^T)$$
$$= \sum_{i=1}^{n}\sigma_i^2 - 2tr(\mathbf{K}^{(t)}\mathbf{A}^T) + tr(\mathbf{A}\mathbf{A}^T) \quad (26)$$

According to Von Neumann's trace inequality, we have $tr(\mathbf{K}^{(t)}\mathbf{A}^T) \le tr(\mathbf{S}^{(t)}\mathbf{S}_A)$, then

$$tr(\mathbf{U}^{(t)}\mathbf{S}^{(t)}\mathbf{U}^{(t)T}\mathbf{A}) = tr(\mathbf{K}^{(t)}\mathbf{A}^T) \le tr(\mathbf{S}^{(t)}\mathbf{S}_A) = tr(\mathbf{U}_A\mathbf{S}^{(t)}\mathbf{U}_A^T\mathbf{U}_A\mathbf{S}_A\mathbf{U}_A^T)$$
$$= tr(\mathbf{U}_A\mathbf{S}^{(t)}\mathbf{U}_A^T\mathbf{A})$$

which leads to

$$\|\mathbf{K}^{(t)} - \mathbf{A}\|_F^2 \ge \|\mathbf{U}_A\mathbf{S}^{(t)}\mathbf{U}_A^T - \mathbf{A}\|_F^2 = \sum_{i=1}^{n}(\sigma_i - \theta_i)^2 \quad (27)$$

Thus, to minimize Eq. (8), we should set $\mathbf{U}^{(t)} = \mathbf{U}_A$, i.e., the eigenvectors of $\mathbf{K}^{(t)}$ should be the same as eigenvectors of $\mathbf{A}$.

To handle the rank function, we define the function f:

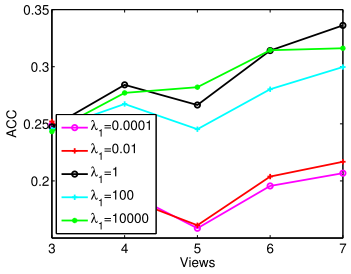
$$f(x) = \begin{cases} 1, & x \ne 0 \\ 0, & x = 0 \end{cases}$$

Since the rank of $\mathbf{K}^{(t)}$ is the number of non-zero singular values of $\mathbf{K}^{(t)}$, Eq. (8) can be rewritten as:

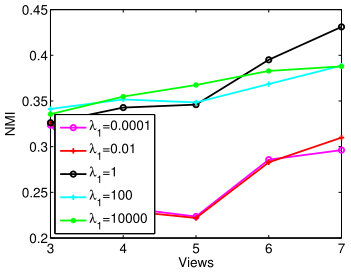
$$\min_{\sigma_1,\ldots,\sigma_n} \sum_{i=1}^{n}(\sigma_i - \theta_i)^2 + \tau\sum_{i=1}^{n}f(\sigma_i), \quad s.t. \quad \sigma_i \ge 0. \quad (28)$$

It is obvious that the solution of Eq. (28) is:

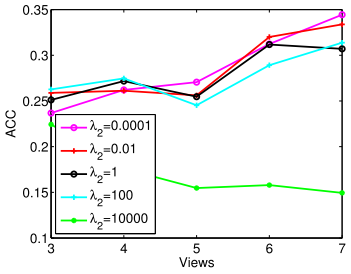
$$\sigma_i = \begin{cases} \theta_i, & \theta_i \ge \sqrt{\tau} \\ 0, & \theta_i < \sqrt{\tau} \end{cases} \quad (29)$$



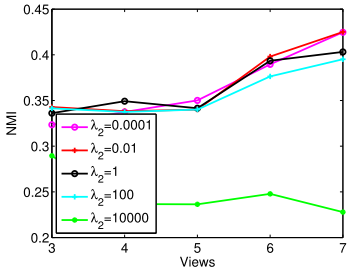
(a) ACC results in different values of λ_1 on Corel data set



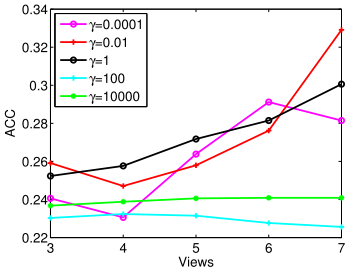
(b) NMI results in different values ofλ_1 on Corel data set



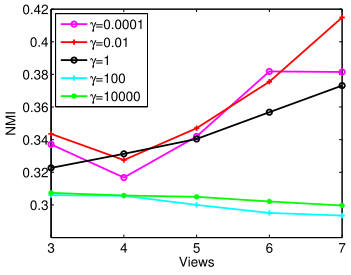
(c) ACC results in different values of λ_2 on Corel data set



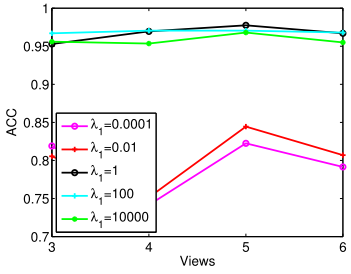
(d) NMI results in different values of λ_2 on Corel data set



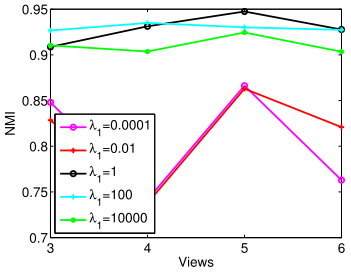
(e) ACC results in different values of γ on Corel data set



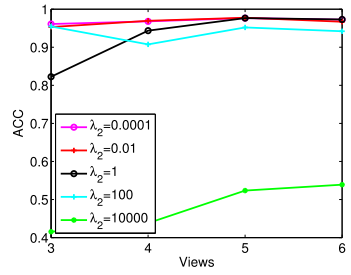
(f) NMI results in different values of γ on Corel data set



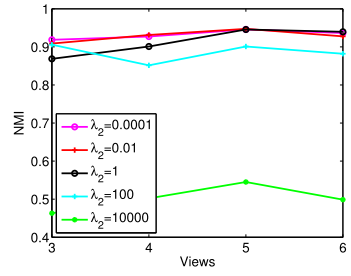
(g) ACC results in different values of λ_1 on UCI Digit data set



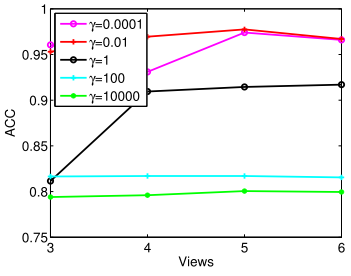
(h) NMI results in different values of λ_1 on UCI Digit data set



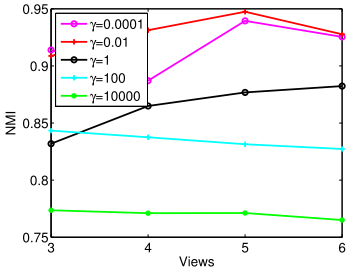
(i) ACC results in different values of λ_2 on UCI Digit data set



(j) NMI results in different values of λ_2 on UCI Digit data set



(k) ACC results in different values of γ on UCI Digit data set



(l) NMI results in different values of γ on UCI Digit data set

Fig. 4. ACC and NMI w.r.t λ_1, λ_2 and γ on UCI Digit and Corel.

To sum up, we get the global optima of this sub-problem $\mathbf{K}^{(t)} = \mathbf{U}_A\mathbf{S}^{(t)}\mathbf{U}_A^T$, where $\mathbf{S}^{(t)}$ is obtained by Eq. (29). Since $\mathbf{K}_b^{(i)}$ $(1 \le i \le m)$ and $\mathbf{K}_c^{(t)}$ are kernel matrices which are symmetric, $\mathbf{A}$ is also symmetric and it leads to that $\mathbf{K}^{(t)}$ is also symmetric.

Proof of Theorem 3. (1)

$$
\begin{aligned}
&(\hat{\mathbf{Y}}^{l+1})^T\hat{\mathbf{Y}}^{l+1} \qquad (30)\\
&= \hat{\mathbf{Y}}^{lT}\left(\mathbf{I}-\frac{\eta}{2}\mathbf{R}\right)^T\left(\left(\mathbf{I}+\frac{\eta}{2}\mathbf{R}\right)^T\right)^{-1}\left(\mathbf{I}+\frac{\eta}{2}\mathbf{R}\right)^{-1}\left(\mathbf{I}-\frac{\eta}{2}\mathbf{R}\right)\hat{\mathbf{Y}}^l\\
&= \hat{\mathbf{Y}}^{lT}\left(\mathbf{I}+\frac{\eta}{2}\mathbf{R}\right)\left(\mathbf{I}-\frac{\eta}{2}\mathbf{R}\right)^{-1}\left(\mathbf{I}+\frac{\eta}{2}\mathbf{R}\right)^{-1}\left(\mathbf{I}-\frac{\eta}{2}\mathbf{R}\right)\hat{\mathbf{Y}}^l\\
&= \hat{\mathbf{Y}}^{lT}\left(\mathbf{I}+\frac{\eta}{2}\mathbf{R}\right)\left(\left(\mathbf{I}+\frac{\eta}{2}\mathbf{R}\right)\left(\mathbf{I}-\frac{\eta}{2}\mathbf{R}\right)\right)^{-1}\left(\mathbf{I}-\frac{\eta}{2}\mathbf{R}\right)\hat{\mathbf{Y}}^l
\end{aligned}
$$

Furthermore, we have

$$
\left(\mathbf{I}+\frac{\eta}{2}\mathbf{R}\right)\left(\mathbf{I}-\frac{\eta}{2}\mathbf{R}\right) = \mathbf{I}-\frac{\eta^2}{4}\mathbf{R}\mathbf{R} = \left(\mathbf{I}-\frac{\eta}{2}\mathbf{R}\right)\left(\mathbf{I}+\frac{\eta}{2}\mathbf{R}\right) \qquad (31)
$$

Take it back into Eq. (30),

$$
\begin{aligned}
&(\hat{\mathbf{Y}}^{l+1})^T(\hat{\mathbf{Y}}^{l+1}) \qquad (32)\\
&= \hat{\mathbf{Y}}^{lT}\left(\mathbf{I}+\frac{\eta}{2}\mathbf{R}\right)\left(\left(\mathbf{I}-\frac{\eta}{2}\mathbf{R}\right)\left(\mathbf{I}+\frac{\eta}{2}\mathbf{R}\right)\right)^{-1}\left(\mathbf{I}-\frac{\eta}{2}\mathbf{R}\right)\hat{\mathbf{Y}}^l\\
&= \hat{\mathbf{Y}}^{lT}\left(\mathbf{I}+\frac{\eta}{2}\mathbf{R}\right)\left(\mathbf{I}+\frac{\eta}{2}\mathbf{R}\right)^{-1}\left(\mathbf{I}-\frac{\eta}{2}\mathbf{R}\right)^{-1}\left(\mathbf{I}-\frac{\eta}{2}\mathbf{R}\right)\hat{\mathbf{Y}}^l\\
&= \hat{\mathbf{Y}}^{lT}\hat{\mathbf{Y}}^l = \mathbf{I}
\end{aligned}
$$

(2) To prove that updating $\hat{\mathbf{Y}}^{l+1}$ is in a descent direction, we first introduce the following lemma:

Lemma 1. *If $\hat{\mathbf{Y}}^{l+1}$ follows Eq. (23) and let $\mathcal{J}(\hat{\mathbf{Y}}^{l+1}) = tr(\hat{\mathbf{Y}}^{l+1T}\mathbf{P}\hat{\mathbf{Y}}^{l+1}) + 2tr(\hat{\mathbf{Y}}^{l+1T}\mathbf{Q})$ which is the objective function of our method, then*

$$
\left.\frac{\partial\mathcal{J}(\hat{\mathbf{Y}}^{l+1})}{\partial\eta}\right|_{\eta=0} = -\frac{1}{2}\|\mathbf{R}\|_F^2 \le 0. \qquad (33)
$$

Proof of Lemma 1. According to the chain rule, we have

$$
\frac{\partial\mathcal{J}(\hat{\mathbf{Y}}^{l+1})}{\partial\eta} = tr\left(\left(\frac{\partial\mathcal{J}(\hat{\mathbf{Y}}^{l+1})}{\partial\hat{\mathbf{Y}}^{l+1}}\right)^T\frac{\partial\hat{\mathbf{Y}}^{l+1}}{\partial\eta}\right) \qquad (34)
$$

When $\eta = 0$, $\hat{\mathbf{Y}}^{l+1} = \hat{\mathbf{Y}}^l$, and $\left.\frac{\partial\mathcal{J}(\hat{\mathbf{Y}}^{l+1})}{\partial\hat{\mathbf{Y}}^{l+1}}\right|_{\eta=0} = 2(\mathbf{P}\hat{\mathbf{Y}}^l+\mathbf{Q})$, $\left.\frac{\partial\hat{\mathbf{Y}}^{l+1}}{\partial\eta}\right|_{\eta=0} = -\mathbf{R}\hat{\mathbf{Y}}^l$.

On one hand,

$$
\begin{aligned}
&\left.\frac{\partial\mathcal{J}(\hat{\mathbf{Y}}^{l+1})}{\partial\eta}\right|_{\eta=0} = -2tr\left((\mathbf{P}\hat{\mathbf{Y}}^l+\mathbf{Q})^T\mathbf{R}\hat{\mathbf{Y}}^l\right) \qquad (35)\\
&= -4tr\left((\mathbf{P}\hat{\mathbf{Y}}^l+\mathbf{Q})^T(\mathbf{P}\hat{\mathbf{Y}}^l+\mathbf{Q}) - (\mathbf{P}\hat{\mathbf{Y}}^l+\mathbf{Q})^T\hat{\mathbf{Y}}^l(\mathbf{P}\hat{\mathbf{Y}}^l+\mathbf{Q})^T\hat{\mathbf{Y}}^l\right)
\end{aligned}
$$

On the other hand, we have

$$
\begin{aligned}
&\|\mathbf{R}\|_F^2 = tr(\mathbf{R}^T\mathbf{R}) \qquad (36)\\
&= 4tr\left(\left((\mathbf{P}\hat{\mathbf{Y}}^l+\mathbf{Q})\hat{\mathbf{Y}}^{lT} - \hat{\mathbf{Y}}^l(\mathbf{P}\hat{\mathbf{Y}}^l+\mathbf{Q})^T\right)^T\right.\\
&\quad\left.\times\left((\mathbf{P}\hat{\mathbf{Y}}^l+\mathbf{Q})\hat{\mathbf{Y}}^{lT} - \hat{\mathbf{Y}}^l(\mathbf{P}\hat{\mathbf{Y}}^l+\mathbf{Q})^T\right)\right)\\
&= 8tr\left((\mathbf{P}\hat{\mathbf{Y}}^l+\mathbf{Q})^T(\mathbf{P}\hat{\mathbf{Y}}^l+\mathbf{Q}) - (\mathbf{P}\hat{\mathbf{Y}}^l+\mathbf{Q})^T\hat{\mathbf{Y}}^l(\mathbf{P}\hat{\mathbf{Y}}^l+\mathbf{Q})^T\hat{\mathbf{Y}}^l\right)
\end{aligned}
$$

So we have $\left.\frac{\partial\mathcal{J}(\hat{\mathbf{Y}}^{l+1})}{\partial\eta}\right|_{\eta=0} = -\frac{1}{2}\|\mathbf{R}\|_F^2 \le 0$

Now we have $\left.\frac{\partial\mathcal{J}(\hat{\mathbf{Y}}^{l+1})}{\partial\eta}\right|_{\eta=0} = -\frac{1}{2}\|\mathbf{R}\|_F^2$, which means if $\hat{\mathbf{Y}}$ moves a small step $\Delta\eta > 0$ in the update direction, the objective function $\mathcal{J}$ will have a change $-\frac{1}{2}\|\mathbf{R}\|_F^2\Delta\eta$ and since $-\frac{1}{2}\|\mathbf{R}\|_F^2 \le 0$, the objective function $\mathcal{J}$ will decrease. Thus the update direction is a descent direction. Moreover, since the objective function is lower bounded by 0, the algorithm will converge.

To prove that it will converge to a stable point, we introduce the following lemma which shows the first-order optimality condition of the objective function:

Lemma 2. *Let $\mathcal{L} = tr(\hat{\mathbf{Y}}^T\mathbf{P}\hat{\mathbf{Y}}) + 2tr(\hat{\mathbf{Y}}^T\mathbf{Q}) - tr(\mathbf{\Lambda}(\hat{\mathbf{Y}}^T\hat{\mathbf{Y}}-\mathbf{I}))$ be the Lagrangian function of our objective function, then $\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{Y}}} = \mathbf{0}$ if and only if $\mathbf{R} = \mathbf{0}$, so $\mathbf{R} = \mathbf{0}$ is the first-order optimality condition of our objective function.*

Proof of Lemma 2. On one hand, according to the definition of $\mathbf{R}$, we have $\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{Y}}} = \mathbf{R}\hat{\mathbf{Y}}$, so if $\mathbf{R} = \mathbf{0}$, then $\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{Y}}} = \mathbf{0}$.

On the other hand, if $\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{Y}}} = \mathbf{0}$, i.e., $\left(\mathbf{P}\hat{\mathbf{Y}}\hat{\mathbf{Y}}^T + \mathbf{Q}\hat{\mathbf{Y}}^T - \hat{\mathbf{Y}}(\mathbf{P}\hat{\mathbf{Y}}+\mathbf{Q})^T\right)\hat{\mathbf{Y}} = \mathbf{0}$. Let $\mathbf{Z} = \mathbf{P}\hat{\mathbf{Y}}+\mathbf{Q}$, then we have $\mathbf{Z} = \hat{\mathbf{Y}}\mathbf{Z}^T\hat{\mathbf{Y}}$ due to $\hat{\mathbf{Y}}^T\hat{\mathbf{Y}} = \mathbf{I}$. Thus,

$$
\mathbf{Z} = \hat{\mathbf{Y}}\mathbf{Z}^T\hat{\mathbf{Y}} = \hat{\mathbf{Y}}(\hat{\mathbf{Y}}\mathbf{Z}^T\hat{\mathbf{Y}})^T\hat{\mathbf{Y}} = \hat{\mathbf{Y}}\hat{\mathbf{Y}}^T\mathbf{Z} \qquad (37)
$$

here we also use the fact that $\hat{\mathbf{Y}}^T\hat{\mathbf{Y}} = \mathbf{I}$.

Take the transposition of both sides, we have $\mathbf{Z}^T = \mathbf{Z}^T\hat{\mathbf{Y}}\hat{\mathbf{Y}}^T$. Then we obtain

$$
\hat{\mathbf{Y}}\mathbf{Z}^T = \hat{\mathbf{Y}}\mathbf{Z}^T\hat{\mathbf{Y}}\hat{\mathbf{Y}}^T = \mathbf{Z}\hat{\mathbf{Y}}^T \qquad (38)
$$

which means $\mathbf{Z}\hat{\mathbf{Y}}^T - \hat{\mathbf{Y}}\mathbf{Z}^T = \mathbf{0}$. Note that $\mathbf{R} = 2(\mathbf{Z}\hat{\mathbf{Y}}^T - \hat{\mathbf{Y}}\mathbf{Z}^T)$, so $\mathbf{R} = \mathbf{0}$.

In summary, $\mathbf{R} = \mathbf{0}$ is the first-order optimality conditions.

Now, get back to Theorem 3. The algorithm converges when $\left.\frac{\partial\mathcal{J}(\hat{\mathbf{Y}}^{l+1})}{\partial\eta}\right|_{\eta=0} = 0$, which means $\hat{\mathbf{Y}}$ cannot move a small step in the descent direction to make the objective function decreases. Since $\left.\frac{\partial\mathcal{J}(\hat{\mathbf{Y}}^{l+1})}{\partial\eta}\right|_{\eta=0} = -\frac{1}{2}\|\mathbf{R}\|_F^2$, $\|\mathbf{R}\|_F^2 = 0$, i.e., $\mathbf{R} = \mathbf{0}$. Due to Lemma 2, it satisfies the first-order optimality condition, so the algorithm converges to a stable point.

References

[1] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[2] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: CVPR, vol. 1, IEEE, 2005, pp. 886–893.
[3] S. Rüping, T. Scheffer, Learning with multiple views, in: Proc. ICML Workshop on Learning with Multiple Views, 2005.
[4] A. Kumar, P. Rai, H. Daume, Co-regularized multi-view spectral clustering, in: Advances in NIPS, 2011, pp. 1413–1421.
[5] X. Cai, F. Nie, H. Huang, Multi-view k-means clustering on big data, in: IJCAI, AAAI Press, 2013, pp. 2598–2604.
[6] M.D. Collins, J. Liu, J. Xu, L. Mukherjee, V. Singh, Spectral clustering with a convex regularizer on millions of images, in: Computer Vision–ECCV 2014, Springer, 2014, pp. 282–298.
[7] R. Xia, Y. Pan, L. Du, J. Yin, Robust multi-view spectral clustering via low-rank and sparse decomposition, in: AAAI, 2014, pp. 2149–2155.
[8] F. Nie, J. Li, X. Li, Self-weighted multiview clustering with multiple graphs, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, 2017, pp. 564–2570.
[9] Z. Zhang, Z. Zhai, L. Li, Uniform projection for multi-view learning, IEEE Trans. Pattern Anal. Mach. Intell. 39 (8) (2017) 1675–1689.
[10] S. Sun, Multi-view laplacian support vector machines, in: Advanced Data Mining and Applications, Springer, 2011, pp. 209–222.
[11] Q. Chen, S. Sun, Hierarchical multi-view fisher discriminant analysis, in: Neural Information Processing, Springer, 2009, pp. 289–298.
[12] J. Liu, C. Wang, J. Gao, J. Han, Multi-view clustering via joint nonnegative matrix factorization, in: Proc. of SDM, vol. 13, 2013, SIAM, pp. 252–260.
[13] H. Wang, Y. Yang, B. Liu, H. Fujita, A study of graph-based system for multi-view clustering, Knowl.-Based Syst. 163 (2019) 1009–1019.

[14] Y. Zhang, Y. Yang, T. Li, H. Fujita, A multitask multiview clustering algorithm in heterogeneous situations based on LLE and LE, Knowl.-Based Syst. 163 (2019) 776–786.
[15] Q. Wang, M. Chen, F. Nie, X. Li, Detecting coherent groups in crowd scenes by multiview clustering, IEEE Trans. Pattern Anal. Mach. Intell. (2018).
[16] W. Ding, T.F. Stepinski, Y. Mu, L. Bandeira, R. Ricardo, Y. Wu, Z. Lu, T. Cao, X. Wu, Subkilometer crater discovery with boosting and transfer learning, ACM Trans. Intell. Syst. Technol. (TIST) 2 (4) (2011) 39.
[17] K. Yu, W. Ding, D.A. Simovici, H. Wang, J. Pei, X. Wu, Classification with streaming features: An emerging-pattern mining approach, Trans. Knowl. Discov. Data 9 (4) (2015) 30.
[18] A. Vergara, J. Fonollosa, J. Mahiques, M. Trincavelli, N. Rulkov, R. Huerta, On the performance of gas sensor arrays in open sampling systems using inhibitory support vector machines, Sensors Actuators B 185 (2013) 462–477.
[19] A. Rahimi, B. Recht, Random features for large-scale kernel machines, in: Advances in NIPS, 2007, pp. 1177–1184.
[20] S. Günnemann, I. Färber, T. Seidl, Multi-view clustering using mixture models in subspace projections, in: SIGKDD, 2012, pp. 132–140.
[21] X. Cao, C. Zhang, H. Fu, S. Liu, H. Zhang, Diversity-induced multi-view subspace clustering, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.
[22] C. Zhang, H. Fu, Q. Hu, X. Cao, Y. Xie, D. Tao, D. Xu, Generalized latent multi-view subspace clustering, IEEE Trans. Pattern Anal. Mach. Intell. (2018).
[23] Y. Xie, D. Tao, W. Zhang, Y. Liu, L. Zhang, Y. Qu, On unifying multi-view self-representations for clustering by tensor multi-rank minimization, Int. J. Comput. Vis. 126 (11) (2018) 1157–1179.
[24] P. Ren, Y. Xiao, P. Xu, J. Guo, X. Chen, X. Wang, D. Fang, Robust auto-weighted multi-view clustering, in: IJCAI, 2018, pp. 2644–2650.
[25] X. Liu, L. Wang, G.B. Huang, J. Zhang, J. Yin, Multiple kernel extreme learning machine, Neurocomputing 149 (2015) 253–264.
[26] X. Liu, Y. Dou, J. Yin, L. Wang, E. Zhu, Multiple kernel k-means clustering with matrix-induced regularization, in: AAAI, 2016, pp. 1888–1894.
[27] X. Liu, S. Zhou, Y. Wang, M. Li, Y. Dou, E. Zhu, J. Yin, H. Li, Optimal neighborhood kernel clustering with multiple kernels, in: AAAI, 2017.
[28] D. Peluffo-Ordónez, S. Garcia-Vega, R. Langone, J.A. Suykens, G. Castellanos-Dominguez, Kernel spectral clustering for dynamic data using multiple kernel learning, in: Neural Networks (IJCNN), The 2013 International Joint Conference on, IEEE, 2013, pp. 1–6.
[29] C. Alzate, J.A. Suykens, Multiway spectral clustering with out-of-sample extensions through weighted kernel pca, IEEE Trans. Pattern Anal. Mach. Intell. 32 (2) (2010) 335–347.
[30] A. Kumar, H. Daumé, A co-training approach for multi-view spectral clustering, in: ICML, 2011, pp. 393–400.
[31] F. Nie, J. Li, X. Li, Parameter-free auto-weighted multiple graph learning: A framework for multiview clustering and semi-supervised classification, in: IJCAI, 2016.
[32] F. Nie, G. Cai, J. Li, X. Li, Auto-weighted multi-view learning for image clustering and semi-supervised classification, IEEE Trans. Image Process. 27 (3) (2018) 1501–1511.
[33] Y. Li, F. Nie, H. Huang, J. Huang, Large-scale multi-view spectral clustering via bipartite graph, in: AAAI, 2015, pp. 2750–2756.
[34] F. Nie, L. Tian, X. Li, Multiview clustering via adaptively weighted procrustes, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, 2018, pp. 2022–2030.
[35] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Found. Trends Mach. Learn. 3 (1) (2011) 1–122.
[36] I. Ahmad, Multi-view video: get ready for next-generation television, IEEE Distrib. Syst. Online 8 (3) (2007).
[37] D. Vlasic, I. Baran, W. Matusik, J. Popović, Articulated mesh animation from multi-view silhouettes, in: ACM Trans. Graph., vol. 27, 2008, ACM, p. 97.
[38] R. Langone, O.M. Agudelo, B. De Moor, J.A. Suykens, Incremental kernel spectral clustering for online learning of non-stationary data, Neurocomputing 139 (2014) 246–260.
[39] U. Von Luxburg, A tutorial on spectral clustering, Stat. Comput. 17 (4) (2007) 395–416.
[40] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.
[41] M. Meila, J. Shi, A random walks view of spectral segmentation, in: International conference on artificial intelligence and statistics, 2001.
[42] S.X. Yu, J. Shi, Multiclass spectral clustering, in: Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on, IEEE, 2003, pp. 313–319.
[43] W. Rudin, Fourier Analysis on Groups, vol. 12, John Wiley & Sons, 1990.
[44] B. Kulis, M. Sustik, I. Dhillon, Learning low-rank kernel matrices, in: ICML, ACM, 2006, pp. 505–512.
[45] P. Zhou, L. Du, L. Shi, H. Wang, Y.D. Shen, Recovery of corrupted multiple kernels for clustering, in: IJCAI, AAAI Press, 2015, pp. 4105–4111.
[46] M. Brand, Fast low-rank modifications of the thin singular value decomposition, Linear Algebra Appl. 415 (1) (2006) 20–30.
[47] D. Goldfarb, Z. Wen, W. Yin, A curvilinear search method for p-harmonic flows on spheres, SIAM J. Imaging Sci. 2 (1) (2009) 84–109.
[48] Z. Wen, W. Yin, A feasible method for optimization with orthogonality constraints, Math. Program. 142 (1–2) (2013) 397–434.
[49] K.B. Petersen, M.S. Pedersen, et al., The matrix cookbook, Tech. Univ. Denmark 7 (2008) 15.
[50] A.J. Smola, B. Schokopf, Sparse greedy matrix approximation for machine learning, in: ICML, 2000, pp. 911–918.
[51] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[52] J.C. French, J.V. Watson, X. Jin, W. Martin, Integrating multiple multi-channel CBIR systems, in: Proc. Inter. Workshop on Multimedia Information Systems (MIS), Citeseer, 2003.
[53] M. Van Breukelen, R.P. Duin, D.M. Tax, J. Den Hartog, Handwritten digit recognition by combined classifiers, Kybernetika 34 (4) (1998) 381–386.
[54] C.H. Lampert, H. Nickisch, S. Harmeling, Learning to detect unseen object classes by between-class attribute transfer, in: CVPR, IEEE, 2009, pp. 951–958.
[55] J. Xiao, J. Hays, K.A. Ehinger, A. Oliva, A. Torralba, Sun database: Large-scale scene recognition from abbey to zoo, in: CVPR, IEEE, 2010, pp. 3485–3492.
[56] N. Helwig, E. Pignanelli, A. Schutze, Condition monitoring of a complex hydraulic system using multivariate statistics, in: Instrumentation and Measurement Technology Conference, 2015, pp. 210–215.
[57] K. Zhan, F. Nie, J. Wang, Y. Yang, Multiview consensus graph clustering, IEEE Trans. Image Process. 28 (3) (2019) 1261–1270.