Continual Multi-view Clustering

Xinhang Wan
wanxinhang@nudt.edu.cn
National University of Defense Technology
Changsha, Hunan, China

Jiyuan Liu
liujiyuan13@nudt.edu.cn
National University of Defense Technology
Changsha, Hunan, China

Weixuan Liang
weixuanliang@nudt.edu.cn
National University of Defense Technology
Changsha, Hunan, China

Xinwang Liu*
xinwangliu@nudt.edu.cn
National University of Defense Technology
Changsha, Hunan, China

Yi Wen
wenyi21@nudt.edu.cn
National University of Defense Technology
Changsha, Hunan, China

En Zhu
enzhu@nudt.edu.cn
National University of Defense Technology
Changsha, Hunan, China

ABSTRACT

With the increase of multimedia applications, data are often collected from multiple sensors or modalities, encouraging the rapid development of multi-view (also called multi-modal) clustering technique. As a representative, late fusion multi-view clustering algorithm has attracted extensive attention due to its low computation complexity yet promising performance. However, most of them deal with the clustering problem in which all data views are available in advance, and overlook the scenarios where data observations of new views are accumulated over time. To solve this issue, we propose a continual approach on the basis of late fusion multi-view clustering framework. In specific, it only needs to maintain a consensus partition matrix and update knowledge with the incoming one of a new data view rather than keep all of them. This benefits a lot by preventing the previously learned knowledge from recomputing over and over again, saving a large amount of computation resource/time and labor force. Nevertheless, we design an alternate and convergent strategy to solve the resultant optimization problem. Also, the proposed algorithm shows excellent clustering performance and time/space efficiency in the experiment.

CCS CONCEPTS

• **Computing methodologies** → **Cluster analysis**.

KEYWORDS

multi-view clustering, late fusion, consensus partition matrix, continual learning

ACM Reference Format:
Xinhang Wan, Jiyuan Liu, Weixuan Liang, Xinwang Liu, Yi Wen, and En Zhu. 2022. Continual Multi-view Clustering. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), October 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3503161.3547864

*Corresponding author

MM '22, October 10–14, 2022, Lisboa, Portugal

ACM ISBN 978-1-4503-9203-7/22/10…$15.00
https://doi.org/10.1145/3503161.3547864

1 INTRODUCTION

With the rapid development of multimedia applications, data observations are often collected from multiple sensors and sources, resulting in an explosive increase of multi-view[1] analysis techniques. Take the short video for an instance, a clip can be always disassembled into image flow, sound record and text description which are obtained from a camera, microphone and producer, respectively. How to label the videos on a large scale is a critical and popular research for pushing them to consumers personally.

Regarding each modality as a data view, multi-view clustering (MVC) becomes an ideal unsupervised option to solve the aforementioned issue. It uncovers the intrinsic clustering structure of data samples by optimally fusing the complementary information of each view. In literature, three popular paradigms are developed, including multi-view subspace clustering [2, 4, 11, 23], multi-view graph clustering [9, 20–22] and multiple kernel clustering [7, 15]. Some of the multi-view subspace clustering methods assume each data view lies on a common subspace, while the others compute a subspace with respect to each view and utilize them afterwards. For example, Liu et al. [10] first obtain robust representations by performing eigen-decomposition on the original data observations, then try to find a consensus subspace among them. In contrast, Kang et al. [6] first compute the subspaces of all views, then concatenate them for further clustering. Nevertheless, most multi-view graph clustering algorithms [5, 8] transform the data samples of multiple views into corresponding undirected graphs based on their pairwise similarities, then perform spectral clustering and graph fusion simultaneously. In multiple kernel clustering framework [1, 26], one always learns the weights of base kernels to obtain a linear kernel combination as the consensus kernel on which the final clustering results can be obtained.

Although the above three types of algorithms have achieved great success, there are still plenty of tough problems. One major concern is the relatively high time and space complexity. Denote the number of data samples as n, multi-view subspace clustering requires $O\left(n^2\right)$ space to store the consensus subspace structure and

[1] also called multi-modal, but use "multi-view" in the following.

$O(n^3)$ time to perform subspace decomposition, making it hard to apply on large-scale tasks. The other two multi-view strategies have the same issues. Fortunately, the emerging late fusion methods [14, 19, 25] greatly ease this problem by projecting data into base partition matrices (linear to sample number) in the early stage. Later, base partitions are fused into a consensus one, where the complexity is $O(n)$. It is worth noting that the storage complexity is reduced to $O(n)$, since only the partition matrices are saved. Due to the low complexity and promising performance, a cluster of researches are also proposed on the basis of [14, 19, 25].

However, existing late fusion methods always deal with the clustering problem in which all data views are available in advance, and overlook the scenarios where data observations of new views are accumulated over time. In a brain-computer interface system, the signal data at a moment will constitute a new view [27]. Under the circumstance of face recognition, the images of different perspectives or time will also form a new view. To deal with these real-time data, one obvious approach to current late fusion methods is recomputing with all data once a new view arrives, leading to a waste of computation resources and time. To address the above issues, we propose a novel algorithm termed continual multi-view clustering (CMVC), which combines continual learning and late fusion into a unified framework. The illustration of our framework is shown in Figure 1. Specifically, CMVC only maintains a consensus partition matrix, i.e., the fusion of all previous view information. Once a newly collected view is available, it first gets a base clustering partition matrix computed by its kernel matrix to update the consensus partition matrix, rather than keeping and re-computing all of them. Nevertheless, an alternate strategy is developed to solve the optimization problem, and the convergence can be guaranteed. Since the consensus partition matrix is the only variable in optimization, the storage complexity is largely reduced. Meanwhile, CMVC is more efficient in fusing the information of previous views due to only two partition matrices being utilized per iteration, reducing its time overhead. In addition, our experimental results show that CMVC achieves high clustering performance with low time and space complexity. Overall, the main contributions are summarized as follows:

(1) CMVC is the first attempt to handle real-time issues in late fusion multi-view clustering literature and will provide an inspiration for future research. By maintaining a consensus partition matrix and fusing only two matrices each time, CMVC only takes linear time and space complexity with respect to sample number and view number, respectively.
(2) We propose a simple, stable and efficient two-step alternate optimization strategy that is proven to be convergent theoretically. Furthermore, its computational and storage complexity are discussed.
(3) To validate the effectiveness of CMVC, we conduct extensive experiments on multiple datasets. Compared to existing algorithms, it has greatly improved the clustering performance as well as time/space efficiency.

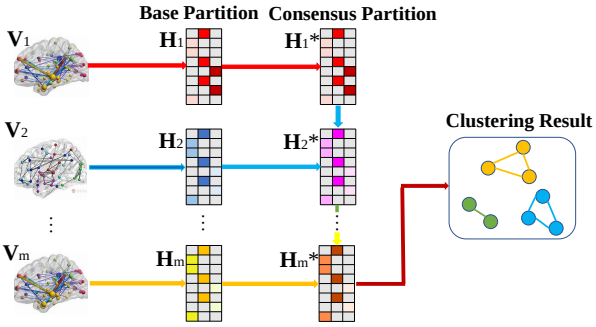

Figure 1: The basic framework of our proposed algorithm. Once the t-th view is available, the consensus partition matrix H_t^* will be updated by fusing H_{t-1}^* and the base partition H_t. CMVC fuses the views one by one until no more new view is available, the final clustering results will be obtained by conducting k-means on H_m^*, where m is the number of total views.

2 RELATED WORK

In this section, we will briefly discuss the most related researches, including multiple kernel k-means and late fusion multi-view clustering algorithms.

2.1 Multiple kernel k-means

Given a group of pre-defined kernels $\{\mathbf{H}_p\}_{p=1}^m$, MKKM assumes that the optimal kernel can be expressed by a linear combination of the pre-defined kernels and constructs the optimal kernel $\mathbf{K}_{\boldsymbol{\beta}}$ as $\mathbf{K}_{\boldsymbol{\beta}} = \sum_{p=1}^m \beta_p^2 \mathbf{K}_p$, where $\mathbf{K}_p$ and β_p are the kernel matrix of the p-th view and its corresponding weight. Based on this, it simultaneously learns the optimal kernel $\mathbf{K}_{\boldsymbol{\beta}}$ with the coefficients $\boldsymbol{\beta}$ and the clustering partition matrix $\mathbf{H}$ as follows

$$\begin{aligned} &\min_{\mathbf{H},\boldsymbol{\beta}} \operatorname{Tr}\left(\mathbf{K}_{\boldsymbol{\beta}}\left(\mathbf{I}_n - \mathbf{H}\mathbf{H}^{\mathrm{T}}\right)\right), \\ &\text{s.t. } \mathbf{H} \in \mathbb{R}^{n\times k}, \mathbf{H}^{\top}\mathbf{H} = \mathbf{I}_k, \boldsymbol{\beta}^{\top}\mathbf{1}_m = 1, \beta_p \geq 0, \forall p. \end{aligned} \tag{1}$$

The problem in Eq. (1) can be solved by an alternate optimization algorithm which is referred in [12]. After obtaining the clustering partition matrix $\mathbf{H}$, a subsequent standard k means is performed on $\mathbf{H}$ to get the final cluster assignments.

Due to the satisfactory clustering performance, a group of methods have been proposed based on MKKM. Upon the alternating minimization framework, [28] jointly optimizes the cluster assignments and kernel weights with proven local convergence. On the basis of Rayleigh quotient objective, it solves the resultant optimization problem in a novel way. Li et al. [13] concentrates on sample pairs with closer relationships by requiring each sample connecting to its k-nearest neighbors. By locally implementing this alignment, this method prevents dissimilar pairs from into the same clusters, thus promoting better performance. Different from the above approaches, [18] proposes a minimization-maximization optimization problem to minimize kernel alignment regarding kernel weight

and maximize it respecting the clustering partition matrix, then transforms this problem into a minimization one.

Despite existing MKKM methods can achieve great clustering performance, most of them need to store and perform eigen decomposition on a $n \times n$ matrix in each iteration, the basic space and time complexity are $O(n^2)$ and $O(n^3)$, respectively. It makes MKKM hard to handle large-scale datasets.

2.2 Late Fusion Multi-view Clustering

Late fusion multi-view clustering (LFMVC) [25, 30, 31] is proposed to reduce the high complexity of MKKM. LFMVC first performs eigen decomposition on base kernel matrices $\{\mathbf{K}_p\}_{p=1}^m$ to obtain base partition matrices $\{\mathbf{H}_p\}_{p=1}^m$, respectively. Then, LFMVC learns a consensus partition matrix $\mathbf{H}^*$ by the following objective:

$$\begin{aligned} \max_{\mathbf{H}^*,\{\mathbf{W}_p\}_{p=1}^m,\beta} & \operatorname{Tr}\left(\mathbf{H}^{*\,\mathrm{T}}\mathbf{X}\right) + \lambda \operatorname{Tr}\left(\mathbf{H}^{*\,\mathrm{T}}\mathbf{M}\right) \\ \text{s.t. } & \mathbf{H}^{*\,\mathrm{T}}\mathbf{H}^* = \mathbf{I}_k, \mathbf{W}_p^{\mathrm{T}}\mathbf{W}_p = \mathbf{I}_k, \\ & \sum_{p=1}^m \beta_p^2 = 1, \beta_p \geq 0, \mathbf{X} = \sum_{p=1}^m \beta_p \mathbf{H}_p \mathbf{W}_p, \end{aligned} \tag{2}$$

where β_p is the coefficient of the p-th view, $\mathbf{W}_p$ is the permutation matrix of p-th view, which makes $\mathbf{H}_p$ and $\mathbf{H}^*$ can be maximally aligned, $\mathbf{M}$ is the average partition region and a regularization parameter λ is introduced to balance the weights between two parts. The problem in Eq. (2) is also addressed by an alternate optimization algorithm [25]. As seen, the eigen decomposition is only performed once for each view. In the iterative process, only m base partition matrices are stored, thus the space complexity is basically $O(n)$. Meanwhile, the time complexity is also linear with the sample number n.

Owing to to low time/space complexity and promising performance, a number of extending algorithms appear on the basis of LFMVC. Based on the observation of the relationship between k-means and the alignment between base partitions and consensus partition, [25] connects the consensus partition with weighted base partitions to attain a better consensus partition. [14] proposes a unified framework to simultaneously optimize the consensus matrix and the generating of cluster labels, eliminating the effects of the information loss between consensus matrix learning and the subsequent k-means. Given the neglect of the inherent local structure, Zhang et al. [29] generate local kernels of each view using the knowledge regarding the nearest neighbor indicator matrix, which improves the robustness of the model.

Although LFMVC achieves high clustering performance with low complexity, a nonnegligible drawback limits its application range. In the setting of LFMVC, the number of views is fixed. In practice, all used views are rarely available at once. When a new view is collected, the existing LFMVC is not able to fuse it efficiently. To address this issue, we propose a novel late fusion algorithm in the following section.

3 CONTINUAL MULTI-VIEW CLUSTERING

In this section, we introduce the objective formulation of CMVC, and then develop a two-step alternate optimization method to solve the problem. After that, its convergence, complexity and extension are discussed.

3.1 Formulation

As we discussed in the previous section, existing LFMVC methods are difficult to deal with the situation when the number of views is increasing over time. When a new view is collected, e.g., new sensors equipped, LFMVC needs to fuse the information of previous views again. As a result, all the view information will be saved and fused repeatedly, inducing a huge waste of time and space resources. To solve this problem, we propose CMVC, which combines LFMVC with continual learning in a creative way. We only retain a consensus partition matrix, and further improve it when a new view is accumulated. In this way, the fusion process is more flexible and economical with respect to both time and space consumption than the original methods.

We will introduce a novel algorithm to fuse the information efficiently. As mentioned above, we utilize a permutation matrix on the base partition matrix to match the two parts suitably. Meanwhile, CMVC introduces a regularization parameter to balance the weights of the two parts of information. Furthermore, unlike previous late fusion methods, CMVC directly learns the base partition $\mathbf{H}_t$ after a linear transformation $\mathbf{H}_t\mathbf{W}_t$.

Suppose that the t-view is collected and $\mathbf{H}_t$ is the base partition, while the consensus partition matrix $\mathbf{H}_{t-1}^*$ has been already acquired. We can attain the optimal permutation matrix $\mathbf{W}_t$ by maximizing the objective formulation as

$$\begin{aligned} \max_{\widetilde{\mathbf{H}}_t,\mathbf{W}_t} & \operatorname{Tr}\left(\widetilde{\mathbf{H}}_t^\top \mathbf{H}_t \mathbf{W}_t\right) + \lambda \operatorname{Tr}\left(\widetilde{\mathbf{H}}_t^\top \mathbf{H}_{t-1}^*\right), \\ \text{s.t. } & \widetilde{\mathbf{H}}_t^\top \widetilde{\mathbf{H}}_t = \mathbf{I}_k, \mathbf{W}_t^\top \mathbf{W}_t = \mathbf{I}_k. \end{aligned} \tag{3}$$

After getting the optimal permutation matrix $\mathbf{W}_t$, CMVC will update $\mathbf{H}_t^*$ by directly combining the information of two parts as follows

$$\mathbf{H}_t^* = \mathbf{H}_{t-1}^* + \mathbf{H}_t\mathbf{W}_t. \tag{4}$$

In order to make the value of $\mathbf{H}_{t-1}^*$ in Eq. (3) more reasonable, let $\widetilde{\mathbf{H}}_{t-1}^* = \mathbf{H}_{t-1}^*/(t-1)$, and Eq. (3) can be expressed as

$$\begin{aligned} \max_{\widetilde{\mathbf{H}}_t,\mathbf{W}_t} & \operatorname{Tr}\left(\widetilde{\mathbf{H}}_t^\top \mathbf{H}_t \mathbf{W}_t\right) + \lambda \operatorname{Tr}\left(\widetilde{\mathbf{H}}_t^\top \widetilde{\mathbf{H}}_{t-1}^*\right), \\ \text{s.t. } & \widetilde{\mathbf{H}}_t^\top \widetilde{\mathbf{H}}_t = \mathbf{I}_k, \mathbf{W}_t^\top \mathbf{W}_t = \mathbf{I}_k. \end{aligned} \tag{5}$$

CMVC fuses the views one by one until no more new view is available, the final clustering results will be obtained by conducting k-means on $\mathbf{H}_m^*$, where m is the number of total views.

As seen from Eq. (5), CMVC just unites two matrices in each update, which is more flexible and time-efficient. By maintaining the consensus partition matrix rather than the partition matrix of each view, it takes less space to store the information of previous views. In total, CMVC is more nimble and economical in time and space.

3.2 Alternate Optimization

There are two variables in Eq. (5) to optimize. Therefore, optimizing them simultaneously is a tough task. To solve the optimization problem, we design a two-step alternate optimization to optimize each variable while the other one is fixed.

Optimization $\widetilde{\mathbf{H}}_t$. Fixing $\mathbf{W}_t$, the optimization in Eq. (5) w.r.t $\widetilde{\mathbf{H}}_t$ can be reduced to

$$\max_{\widetilde{\mathbf{H}}_t} \operatorname{Tr}\left(\widetilde{\mathbf{H}}_t^\top \mathbf{A}\right), \text{ s.t. } \widetilde{\mathbf{H}}_t^\top \widetilde{\mathbf{H}}_t = \mathbf{I}_k. \tag{6}$$

where $\mathbf{A} = \mathbf{H}_t\mathbf{W}_t + \lambda\widetilde{\mathbf{H}}_{t-1}^*$. If the matrix $\mathbf{A}$ has the singular value decomposition (SVD) form as $\mathbf{A} = \mathbf{S}\Sigma\mathbf{V}^{\mathrm{T}}$, the optimization problem in Eq. (6) can be solved by a closed-form solution in [25] as follows

$$\widetilde{\mathbf{H}}_t = \mathbf{S}\mathbf{V}^{\mathrm{T}} \tag{7}$$

Optimization $\mathbf{W}_t$. Fixing $\widetilde{\mathbf{H}}_t$, the optimization in Eq. (5) w.r.t $\mathbf{W}_t$ is equivalent to

$$\max_{\mathbf{W}_t} \operatorname{Tr}\left(\mathbf{W}_t^\top \mathbf{B}\right) \text{ s.t. } \mathbf{W}_t^\top \mathbf{W}_t = \mathbf{I}_k \tag{8}$$

where $\mathbf{B} = \mathbf{H}_t^\top \widetilde{\mathbf{H}}_t$. Similar to Eq. (6), Eq. (8) can be efficiently solved by SVD with computational complexity $O\left(nk^2\right)$.

Algorithm 1 Continual Multi-view Clustering (CMVC)

Input: $\{\mathbf{H}_t\}_{t=1}^m$, k, λ and ε_0.
Output: $\mathbf{H}_m^*$.

```
Initialize the consensus partition matrix H_1^* = H_1.
for t = 2 to m do
    H̃_{t-1}^* = H_{t-1}^* / (t − 1).
    Initialize W_t = I_k and i = 1.
    while not converged do
        Update H̃_t by solving Eq. (6).
        Update W_t by solving Eq. (8).
        i ← i + 1
    end while (obj^i − obj^{i−1}) / obj^i ≤ ε_0
    Update H_t^* based on Eq. (4).
end for
```

The optimization process of the entire algorithm is summarized in Algorithm 1. It is worth noting that the views arrive in sequence in practical applications and $\mathbf{H}_t$ is generated when a new view is obtained. In the following part, we will prove the convergence of CMVC in theory and then discuss about its complexity and extension.

3.3 Discussion

3.3.1 Convergence. By Cauchy-Schwartz inequality, we divide Eq. (5) into two parts as

$$\operatorname{Tr}\left(\widetilde{\mathbf{H}}_t^\top \mathbf{H}_t\mathbf{W}_t\right) \le \|\widetilde{\mathbf{H}}_t^\top\|_F \|\mathbf{H}_t\mathbf{W}_t\|_F = k, \tag{9}$$

and

$$\operatorname{Tr}\left(\widetilde{\mathbf{H}}_t^\top \widetilde{\mathbf{H}}_{t-1}^*\right) \le \|\widetilde{\mathbf{H}}_t^\top\|_F \|\widetilde{\mathbf{H}}_{t-1}^*\|_F. \tag{10}$$

Since $\widetilde{\mathbf{H}}_{t-1}^*$ is a fixed, we denote that $\|\widetilde{\mathbf{H}}_{t-1}^*\|_F = c$, where c is a constant. Then, Eq. (10) can be upper bounded by

$$\operatorname{Tr}\left(\widetilde{\mathbf{H}}_t^\top \widetilde{\mathbf{H}}_{t-1}^*\right) \le c\sqrt{k}. \tag{11}$$

Combining Eq. (9) and Eq. (11), we can obtain

$$\operatorname{Tr}\left(\widetilde{\mathbf{H}}_t^\top \mathbf{H}_t\mathbf{W}_t\right) + \lambda \operatorname{Tr}\left(\widetilde{\mathbf{H}}_t^\top \widetilde{\mathbf{H}}_{t-1}^*\right) \le k + c\lambda\sqrt{k}. \tag{12}$$

Therefore, the objective function has an upper bound. We will verify in the following part that the objective value of Eq. (5) monotonically increases. For the ease of expression, we simplify the objective in Eq. (5) as

$$\max_{\widetilde{\mathbf{H}}_t, \mathbf{W}_t} f\left(\widetilde{\mathbf{H}}_t, \mathbf{W}_t\right), \quad \text{s.t. } (\widetilde{\mathbf{H}}_t, \mathbf{W}_t) \in \Delta. \tag{13}$$

As demonstrated in Algorithm 1, the optimization process consists of two iterative parts in each iteration, i.e. $\widetilde{\mathbf{H}}_t$ subproblem and $\mathbf{W}_t$ subproblem. It is worth mentioning that superscript i denotes the optimization at round i. The convergence analysis is given as follows:

1) $\widetilde{\mathbf{H}}_t$-subproblem: Given $\mathbf{W}_t^{(i)}$, $\widetilde{\mathbf{H}}_t^{(i+1)}$ can be obtained via optimizing (6), leading to
$$f\left(\widetilde{\mathbf{H}}_t^{(i+1)}, \mathbf{W}_t^{(i)}\right) \ge f\left(\widetilde{\mathbf{H}}_t^{(i)}, \mathbf{W}_t^{(i)}\right). \tag{14}$$
2) $\mathbf{W}_t$-subproblem: Given $\widetilde{\mathbf{H}}_t^{(i+1)}$, $\mathbf{W}_t^{(i+1)}$ can be obtained via optimizing (8), leading to
$$f\left(\widetilde{\mathbf{H}}_t^{(i+1)}, \mathbf{W}_t^{(i+1)}\right) \ge f\left(\widetilde{\mathbf{H}}_t^{(i+1)}, \mathbf{W}_t^{(i)}\right). \tag{15}$$

Combining Eq. (14) and Eq. (15), the following inequality holds that:

$$f\left(\widetilde{\mathbf{H}}_t^{(i+1)}, \mathbf{W}_t^{(i+1)}\right) \ge f\left(\widetilde{\mathbf{H}}_t^{(i)}, \mathbf{W}_t^{(i)}\right), \tag{16}$$

which demonstrates that the objective value monotonically increases with iterations.

Based on Eq. (12), we can conclude that Eq. (5) exists an upper bound. Therefore, the algorithm is theoretically convergent. Furthermore, we will verify that the algorithm is convergent in the experiment.

3.3.2 Computation Complexity. According to the optimization process outlined in Algorithm 1, the computational complexity of CMVC in each iteration is $O\left(nk^2\right)$. Let T denote the maximum number of iterations and m represent the number of views, the computation complexity of CMVC is $O\left(Tmnk^2\right)$. Compared with traditional LFMVC methods we have mentioned before, they need to fuse all the views when a fresh view is available, so the computation complexity of traditional LFMVC is $O\left(m^2\right)$ with respect to m, which is $O\left(m\right)$ in CMVC.

3.3.3 Space Complexity. As can be seen in Algorithm 1, CMVC only needs to save a consensus partition matrix instead of the partition matrices of all views. It is obvious that CMVC requires less space and the space complexity is $O\left(nk\right)$.

3.3.4 Extension. To our knowledge, for the first time, CMVC combines late fusion and continual learning, which will provide an inspiration for future research. Furthermore, by keeping a consensus partition matrix rather than the partition matrix of each view, CMVC is more flexible and takes less space, which can be extended to other multi-view clustering methods.

4 EXPERIMENTAL RESULTS

In this section, we conduct experiments to compare CMVC with several multiple kernel clustering and late fusion methods on several representative datasets. After that, its convergence analysis, running time and parameter sensitivity are discussed. In addition,

Table 1: Datasets used in our experiments.

Dataset	Samples	Kernels	Clusters
3Sources	169	3	6
Olympics	464	9	29
BBCSport	544	2	5
Cora	2708	2	7
Flower102	8189	4	102
CCV	6773	3	20
AR10P	130	6	10
SUNRGBD	10335	2	45
NUS-WIDE	30000	5	31
AwA	30475	6	50

we investigate the effect of the view order and number of views in our algorithm, respectively.

4.1 Experimental Settings

4.1.1 Datasets. Ten benchmark datasets of different categories are used to testify the effectiveness of CMVC, including 3Sources[2], Olympics[3], BBCSport[4], Cora[5], Flower102[6], CCV[7], AR10P[8], SUNRGBD[9], NUS-WIDE[10] and AwA[11]. The information of each dataset is summarized in Table 1. It is clear that the first seven datasets are regular datasets and the last three are large-scale datasets. Since multiple kernel learning will result in out-of-memory error on large-scale datasets, we only compare IV-LFMC to late fusion methods for the latter three datasets.

4.1.2 Compared Algorithms.

(1) **Average kernel k-means (Avg-KKM)**. All kernels are used to construct the optimal kernel by assigning the same weights and then use the optimal kernel as the input of kernel k-means.
(2) **Localized multiple kernel k-means (LMKKM)** [3]. LMKKM combines kernels in a localized way to get better data characteristics.
(3) **Optimal neighborhood kernel clustering (ONKC)** [16]. ONKC builds a bridge between kernel learning and clustering. It improves the represent ability of the optimal kernel.
(4) **Simplemkkm: Simple multiple kernel k-means (SMKKM)** [17]. SMKKM performs multiple kernel learning by extending supervised kernel alignment and then transfers the problem into a smooth minimization one.
(5) **Multiple kernel k-means with matrix-induced regularization (MKKM-MiR)** [13]. MKKM-MiR utilizes a matrix-induced regularization in order to reduce the redundancy of the base kernels.
(6) **Late fusion multiple kernel clustering with local kernel alignment maximization (LF-LKA)** [29]. LF-LKA is a late fusion based method. It proposes a multiple kernel clustering method with a local kernel alignment to concentrate on closer sample pairs.
(7) **Late fusion multi-view clustering via global and local alignment maximization(LF-GAM)** [24]. LF-GAM is a late fusion based method. It fuses partition level information from every individual view and aligns the consensus partition information with the weighted ones.
(8) **One pass late fusion multi-view clustering (OP-LFMVC)** [14]. OP-LFMVC is also a late fusion based method. It optimally fuses the base partition matrices of each view to learn a consensus one.

The implementations of the above algorithms are publicly available on corresponding websites. So we directly use these codes without any changes in our experiments. As for algorithms with hyper-parameters, we adopt the same method as these papers to tune the hyper-parameters and choose the best results to compare. And in CMVC, we set $\lambda \in 2. \wedge [-10, -8, \cdots, 10]$. Then We evaluate the performance of algorithms through three widely used metrics, including accuracy (ACC), normalized mutual information (NMI) and purity. By the way, to reduce the impact of the randomness of k-means, each algorithm performs k-means 50 times and takes their average as the final result. Our experiments are implemented on a desktop computer with an Intel(R) Core(TM) i9-10850K CPU and 128 GB RAM, MATLAB 2020b (64-bit). Moreover, the code of CMVC is available on Github [12].

4.2 Experimental Results

4.2.1 Clustering Performance Comparison on Regular Datasets. Table 2 demonstrates the results of CMVC and the comparison algorithms on seven regular datasets based on the three clustering metrics mentioned above. From this table, we have the following observations:

(1) As a method of LFMVC, CMVC can handle situations where the number of views is not fixed and can grow over time. Furthermore, CMVC is a great improvement over LFMVC and achieves the best clustering performance among the compared algorithms. For instance, CMVC exceeds the second best algorithm by 2.88%, 1.20% , 0.26%, 9.10%, 2.90%, 3.97% and 0.91% in terms of ACC on the benchmark datasets, and the improvements on other metrics are similar, illustrating that CMVC can effectively combine the information between different views.
(2) Late fusion based multi-view clustering methods outperform the traditional multiple kernel clustering ones by a large margin. LF-LKA [29], LF-GAM [24] and OP-LFMVC [14] show better results on most benchmark datasets in terms of ACC, NMI and Purity. For example, the worst result of LFMVC methods exceeds the best one of kernel-based methods by 19.49%, 59.95% and 8.99% with respect to ACC on 3Sources, BBCSport and CCV. These results verify the excellent clustering performance of LFMVC. Compared to MKKC, LFMVC is

[2]http://mlg.ucd.ie/datasets/3sources.html
[3]http://mlg.ucd.ie/aggregation/
[4]http://mlg.ucd.ie/datasets/segment.html
[5]http://linqs-data.soe.ucsc.edu/public/lbc/
[6]http://www.robots.ox.ac.uk/~vgg/data/flowers/102/
[7]http://www.ee.columbia.edu/ln/dvmm/CCV/
[8]http://featureselection.asu.edu/
[9]http://rgbd.cs.princeton.edu/
[10]https://lms.comp.nus.edu.sg/wp-content/
[11]https://cvml.ist.ac.at/AwA/
[12]https://github.com/wanxinhang/CMVC

Table 2: Empirical evaluation and comparison of CMVC with nine baseline methods on 7 regular benchmark datasets in terms of clustering accuracy (ACC), normulaized mutual information (NMI) and Purity.

datasets	Avg-KKM	MKKM	LMKKM	ONKC	SMKKM	MKKM-MiR	LF-LKA	LF-GAM	OP-LFMVC	CMVC
ACC(%)										
3Sources	40.83	41.42	34.91	45.56	32.54	45.56	58.58	54.44	61.54	**63.31**
Olympic	74.78	69.83	59.70	80.17	74.78	78.02	81.47	77.16	76.29	**81.68**
BBCSport	38.97	38.97	38.97	39.34	38.97	38.97	59.56	59.38	61.03	**61.76**
Cora	31.50	25.15	22.64	40.84	35.82	35.86	35.08	47.45	43.72	**51.77**
Flower102	27.29	22.81	22.57	40.93	40.97	40.13	43.84	43.24	31.21	**45.11**
CCV	19.74	17.94	18.68	22.46	21.53	20.97	26.22	25.48	24.48	**27.26**
AR10P	38.46	40.00	43.85	44.62	40.00	40.77	43.85	44.97	36.92	**45.38**
NMI(%)										
3Sources	30.67	31.27	14.22	34.78	15.55	31.81	47.43	53.32	51.42	**55.19**
Olympics	80.41	78.72	66.50	85.05	80.07	84.82	87.07	84.89	86.89	**87.17**
BBCSport	15.44	15.44	15.30	15.87	15.44	15.44	41.00	39.03	**46.27**	**44.07**
Cora	16.78	9.44	6.72	23.15	19.16	19.14	19.43	29.74	24.33	**30.51**
Flower102	46.32	42.92	43.24	57.06	57.75	56.91	57.33	57.29	47.35	**59.47**
CCV	17.16	15.52	14.41	18.78	18.33	18.05	21.08	19.81	18.44	**21.90**
AR10P	37.27	39.53	41.54	39.33	38.95	37.35	40.14	45.80	36.47	**50.04**
Purity(%)										
3Sources	56.21	56.80	47.34	56.21	46.75	56.21	71.01	**75.74**	71.60	**74.56**
Olympics	82.11	79.53	66.81	81.90	82.97	87.07	88.69	87.28	85.78	**88.79**
BBCSport	48.71	48.71	48.71	48.71	48.71	48.71	68.38	67.28	**72.06**	**70.22**
Cora	41.47	35.78	35.08	44.72	47.16	47.16	46.27	54.03	51.44	**56.72**
Flower102	32.28	27.88	28.79	46.49	47.54	43.07	50.12	49.37	35.00	**50.54**
CCV	23.98	22.21	21.87	24.67	25.20	23.83	28.60	28.01	27.09	**30.67**
AR10P	39.23	40.00	44.62	40.77	40.77	36.92	43.85	44.97	37.69	**46.92**

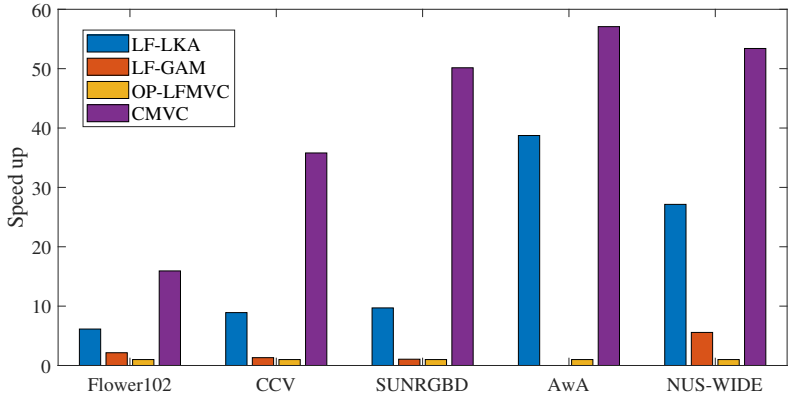

Figure 2: Speed up comparison of three LFMVC algorithms on five benchmark datasets. In each dataset, the algorithm that consumes the most time are listed as a reference and the corresponding speed up is 1.

more time and space efficient and achieves better clustering performance.

(3) CMVC shows an advantage when processing data of large size. For example, among the 7 regular benchmark datasets, there are more data samples in Folwer102 and CCV. Meanwhile, CMVC shows more wonderful results in terms of ACC, NMI and Purity on the two datasets.

To be summarized, CMVC demonstrates superior clustering performance on all datasets and can be used in situations where the number of views can increase over time. In addition, LFMVC shows better results than multiple kernel clustering, which provides some inspirations for future research.

4.2.2 Clustering Performance Comparison on Large-scale Datasets. Since the multiple kernel clustering algorithms assume huge time and space resources, it is tough to process large-scale data. Thus, we just compare CMVC with three LFMVC algorithms, and the clustering results are shown in Table 3. Note that '-' indicates the algorithm cannot be executed completely due to out-of-memory error. From the table, we can conclude that CMVC also performs

Table 3: Empirical evaluation and comparison of CMVC with three baseline methods on 3 large-scale benchmark datasets in terms of clustering accuracy (ACC), normulaized mutual information (NMI) and Purity.

datasets	LF-LKA	LF-GAM	OP-LFMVC	CMVC
ACC(%)				
SUNRGBD	18.90	19.38	**19.91**	18.37
AwA	10.64	-	10.02	**11.26**
NUS-WIDE	12.96	13.36	10.47	**14.97**
NMI(%)				
SUNRGBD	23.02	23.15	21.03	**23.37**
AwA	12.53	-	11.55	**13.78**
NUS-WIDE	11.53	11.31	8.35	**13.65**
Purity(%)				
SUNRGBD	38.98	38.70	36.67	**39.68**
AwA	13.06	-	11.63	**13.59**
NUS-WIDE	24.24	23.98	19.77	**25.87**

well on large-scale datasets. As the number of data increases, the algorithm shows better performance. For example, CMVC exceeds the second best one by 12.06%, 18.39% and 6.72% in ACC, NMI and Purity on NUS-WIDE. Similarly, the clustering results on AwA are superior to LF-MKA and OP-LFMVC, outperforming the second best method by 5.83%, 9.98% and 4.06% on the three metrics, while LF-GAM gets trouble handling this dataset. In addition, CMVC is more flexible because it is able to deal with continual multi-view data. Overall, the algorithm has a good application scenario because of its low computation complexity and excellent clustering performance.

4.2.3 Running Time Comparison. We conduct experiments to evaluate the time efficiency of our proposed algorithm and the results are shown in Figure 2. Note that the running time of CMVC is obtained from the arrival of the first view until the fusion of the last view, and the speed up of the method which consumes the most time on each dataset is set to 1 as a reference. To compare the results precisely, we also record them in Table 4. It can be concluded that CMVC consumes much less time to process multi-view data. Besides, the clustering performance of CMVC is satisfactory.

4.3 Convergence and Parameter Sensitivity Study

4.3.1 Convergence. We have theoretically proved the convergence of our proposed algorithm. In this section, we experimentally verify that the algorithm will eventually converge to a local optimum. We fix λ to 2 and obtain the objective values of CMVC with the iteration number on Flower102 and NUS-WIDE datasets (one regular dataset and one large-scale dataset). Corresponding curves are plotted in Figure 3. The results on other datasets are similar and omitted because of space limit. It can be observed that its objective value increases monotonically and the algorithm converges in less than 20 iterations.

4.3.2 Parameter Sensitivity Study. As seen in Eq. (5), CMVC introduces a regularization parameter λ to balance the learning efficiency between the learned information and the information of the coming view. In our experiments, we tune it in $\left[2^{-10}, 2^{-9}, \cdots, 2^{10}\right]$ by grid

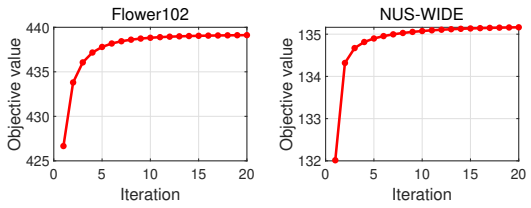

Figure 3: The objective values of CMVC varies with iterations ($\lambda = 2^2$). The results on other datasets are similar and omitted due to space limit.

research. In this section, we conduct experiments to demonstrate the influence of this parameter on two datasets. Other datasets are not presented because of the space limit, but their results are also favorable in most cases. The results are shown in Figure 4. It is worth noting that the experimental results of the second best method are also plotted in these figures as a reference. From these figures, we can observe that: 1) CMVC is stable and performs well when λ ranges from $\left[2^0, 2^1, \cdots, 2^{10}\right]$, which inspires us how to to set the hyperparameter in practical applications; 2) Most of the experimental results increase at first and then keep steady, outperforming the second best algorithm.

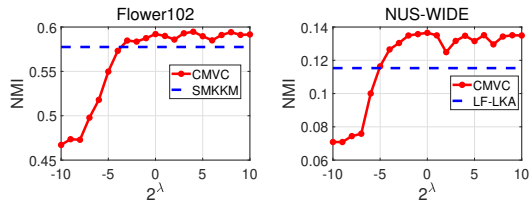

Figure 4: The sensitivity of CMVC with the variation of λ in terms of NMI on two datasets compared with the second best method. The curves on other datasets are similar and we omit them due to space limit.

4.4 Experiment on the effect of view order

Considering the algorithm deals with the scenarios where the number of views can increase, we conduct experiments to investigate the effect of different fusion orders on the results. For instance, suppose that there are three views $\{V_1, V_2, V_3\}$ in total. If V_1, V_2 and V_3 are fused in turn, the final experimental results may differ from other fusion sequences, such as the fusing order of V_3, V_2 and V_1 . Let m denote the number of views, there are $m! = m*(m-1)*\cdots*1$ fusing orders. For example, there are 4 views in Flower102, so its number of fusing orders is 24. Figure 5 reports the results of CMVC compared with the second and third best methods with respect to different orders on Flower102. The clustering performance on other datasets is also excellent but not shown because of space limit. From Figure 5 we can conclude that the different fusing orders of views have little effect on the results, and the results are far better than those of the comparison algorithms. Therefore, it can be obtained that our approach is robust and efficient.

Table 4: Running time comparison of compared methods under several datasets.

Method	Flower102		CCV		SUNRGBD		AwA		NUS-WIDE	
	Time	Speed up	Time	Speed up	Time	Speed up	Time	Speed up	Time	Speed up
LF-LKA	8.31	× 6.14	0.95	× 8.91	2.69	× 9.70	15.19	× 38.74	6.14	× 27.14
LF-GAM	23.70	× 2.15	6.45	× 1.32	24.68	× 1.06	-	-	29.89	× 5.57
OP-LFMVC	51.00	× 1.00	8.51	× 1.00	26.08	× 1.00	588.48	× 1.00	166.61	× 1.00
CMVC	3.20	× 15.93	0.24	× 35.80	0.52	× 50.15	10.31	× 57.08	3.12	× 53.40

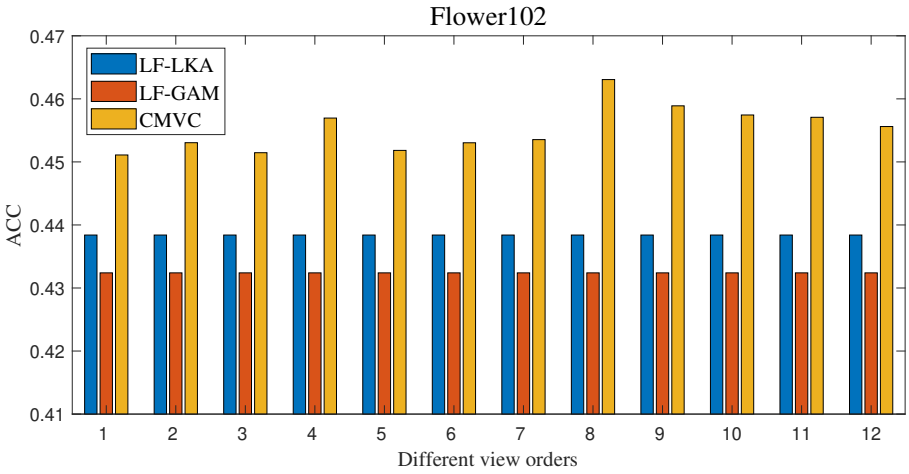

Figure 5: The results of CMVC varies with different orders of views compared with the second and third best methods on Flower102. The results on other datasets are similar and we omit them due to space limit.

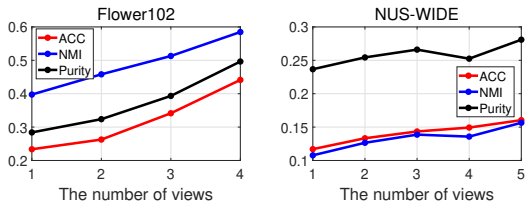

Figure 6: The results of the proposed method with variations in the number of views on two datasets ($\lambda = 2^2$). The ones on other datasets are similar and omitted due to space limit.

4.5 Experiment on the effect of view number

Considering that CMVC can handle the situations where the number of views can grow with time, we perform experiments to investigate whether CMVC fuses the information greatly with the increase of view number. In our experiments, we document the clustering results by increasing the number of views on different datasets one by one. Similar to the previous section, due to space limit, we only demonstrate the results of two datasets. Also, CMVC is still working on other datasets. The results are shown in Figure 6. We can see that as the number of views increases, the experiment results become more and more exceptional. It can be obtained that the algorithm integrates the information of the previous views well, and for the newly added view, it can also be adequately combined with the known information. So we can conclude that the CMVC can be greatly applied to the situation where the number of views increases.

5 CONCLUSION

In this paper, we address the shortcomings of existing LFMVC algorithms that are difficult to handle with scenarios where the number of views increases. Based on this, we combine LFMVC with continual learning in a unified framework and propose a continual multi-view clustering algorithm. By keeping a consensus partition matrix that contains the view information fused before, CMVC only needs to fuse two matrices when a newly collected view is available, which is more flexible and economical in time and space. In addition, comprehensive experiments demonstrate the excellent efficiency of the proposed method. In future research, we intend to investigate how this algorithm can be used to deal with incomplete views.

ACKNOWLEDGMENTS

This work was supported by the National Key R&D Program of China 2020AAA0107100 and the National Natural Science Foundation of China (project no. 61773392, 61872377, 61922088 and 61976196).

REFERENCES

[1] Liang Du, Peng Zhou, Lei Shi, Hanmo Wang, Mingyu Fan, Wenjian Wang, and Yi-Dong Shen. 2015. Robust Multiple Kernel K-Means Using l2;1-Norm. In *Proceedings of the 24th International Conference on Artificial Intelligence* (Buenos Aires, Argentina) *(IJCAI'15)*. AAAI Press, 3476–3482.

[2] Hongchang Gao, Feiping Nie, Xuelong Li, and Heng Huang. 2015. Multi-view Subspace Clustering. In *2015 IEEE International Conference on Computer Vision (ICCV)*. 4238–4246. https://doi.org/10.1109/ICCV.2015.482

[3] Mehmet Gönen and Adam A Margolin. 2014. Localized Data Fusion for Kernel k-Means Clustering with Application to Cancer Biology. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*.

[4] Liping Jing, Kuang Tian, and Joshua Zhexue Huang. 2015. Stratified feature sampling method for ensemble clustering of high dimensional data. *Pattern Recognit.* 48 (2015), 3688–3702.

[5] Zhao Kang, Guoxin Shi, Shudong Huang, Wenyu Chen, Xiaorong Pu, Joey Tianyi Zhou, and Zenglin Xu. 2020. Multi-graph fusion for multi-view spectral clustering. *Knowledge-Based Systems* 189 (2020), 105102. https://doi.org/10.1016/j.knosys.2019.105102

[6] Zhao Kang, Wangtao Zhou, Zhitong Zhao, Junming Shao, Meng Han, and Zenglin Xu. 2020. Large-Scale Multi-View Subspace Clustering in Linear Time. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 4412–4419. https://ojs.aaai.org/index.php/AAAI/article/view/5867

[7] Liang Li, Siwei Wang, Xinwang Liu, En Zhu, Li Shen, Kenli Li, and Keqin Li. 2022. Local Sample-weighted Multiple Kernel Clustering with Consensus Discriminative Graph. *arXiv preprint arXiv:2207.02846* (2022).

[8] Yeqing Li, Feiping Nie, Heng Huang, and Junzhou Huang. 2015. Large-Scale Multi-View Spectral Clustering via Bipartite Graph. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (Austin, Texas) *(AAAI'15)*. AAAI Press, 2750–2756.

[9] Weixuan Liang, Sihang Zhou, Jian Xiong, Xinwang Liu, Siwei Wang, En Zhu, Zhiping Cai, and Xin Xu. 2022. Multi-View Spectral Clustering With High-Order Optimal Neighborhood Laplacian Matrix. *IEEE Transactions on Knowledge and Data Engineering* 34, 7 (2022), 3418–3430. https://doi.org/10.1109/TKDE.2020.3025100

[10] Jiyuan Liu, Xinwang Liu, Yuexiang Yang, Xifeng Guo, Marius Kloft, and Liangzhong He. 2021. Multiview Subspace Clustering via Co-Training Robust Data Representation. *IEEE Transactions on Neural Networks and Learning Systems* (2021), 1–13. https://doi.org/10.1109/TNNLS.2021.3069424

[11] Suyuan Liu, Siwei Wang, Pei Zhang, Kai Xu, Xinwang Liu, Changwang Zhang, and Feng Gao. 2022. Efficient one-pass multi-view subspace clustering with consensus anchors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 7576–7584.

[12] Xinwang Liu, Yong Dou, Jianping Yin, Lei Wang, and En Zhu. 2016. Multiple Kernel k-Means Clustering with Matrix-Induced Regularization. In *AAAI*.

[13] Xinwang Liu, Yong Dou, Jianping Yin, Lei Wang, and En Zhu. 2016. Multiple Kernel k-Means Clustering with Matrix-Induced Regularization. *AAAI Press* (2016).

[14] Xinwang Liu, Li Liu, Qing Liao, Siwei Wang, Yi Zhang, Wenxuan Tu, Chang Tang, Jiyuan Liu, and En Zhu. 2021. One Pass Late Fusion Multi-view Clustering. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 6850–6859. https://proceedings.mlr.press/v139/liu21l.html

[15] Xinwang Liu, Lei Wang, Xinzhong Zhu, Miaomiao Li, En Zhu, Tongliang Liu, Li Liu, Yong Dou, and Jianping Yin. 2020. Absent Multiple Kernel Learning Algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 6 (2020), 1303–1316. https://doi.org/10.1109/TPAMI.2019.2895608

[16] Xinwang Liu, Sihang Zhou, Yueqing Wang, Miaomiao Li, Yong Dou, En Zhu, and Jianping Yin. 2017. Optimal Neighborhood Kernel Clustering with Multiple Kernels. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (San Francisco, California, USA) *(AAAI'17)*. AAAI Press, 2266–2272.

[17] X. Liu, E. Zhu, J. Liu, T. Hospedales, Y. Wang, and M. Wang. 2020. SimpleMKKM: Simple Multiple Kernel K-means. (2020).

[18] Xinwang Liu, En Zhu, Jiyuan Liu, Timothy M. Hospedales, Yang Wang, and Meng Wang. 2020. SimpleMKKM: Simple Multiple Kernel K-means. *CoRR* abs/2005.04975 (2020). arXiv:2005.04975 https://arxiv.org/abs/2005.04975

[19] Xinwang Liu, Xinzhong Zhu, Miaomiao Li, Lei Wang, Chang Tang, Jianping Yin, Dinggang Shen, Huaimin Wang, and Wen Gao. 2019. Late Fusion Incomplete Multi-View Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 10 (2019), 2410–2423. https://doi.org/10.1109/TPAMI.2018.2879108

[20] Yue Liu, Wenxuan Tu, Sihang Zhou, Xinwang Liu, Linxuan Song, Xihong Yang, and En Zhu. 2022. Deep Graph Clustering via Dual Correlation Reduction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 7603–7611.

[21] Yue Liu, Xihong Yang, Sihang Zhou, and Xinwang Liu. 2022. Simple Contrastive Graph Clustering. *arXiv preprint arXiv:2205.07865* (2022).

[22] Feiping Nie, Jing Li, and Xuelong Li. 2016. Parameter-Free Auto-Weighted Multiple Graph Learning: A Framework for Multiview Clustering and Semi-Supervised Classification. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, Subbarao Kambhampati (Ed.). IJCAI/AAAI Press, 1881–1887. http://www.ijcai.org/Abstract/16/269

[23] Xiukun Sun, Miaomiao Cheng, Chen Min, and Liping Jing. 2019. Self-Supervised Deep Multi-View Subspace Clustering. In *Proceedings of The 11th Asian Conference on Machine Learning, ACML 2019, 17-19 November 2019, Nagoya, Japan (Proceedings of Machine Learning Research, Vol. 101)*, Wee Sun Lee and Taiji Suzuki (Eds.). PMLR, 1001–1016. http://proceedings.mlr.press/v101/sun19a.html

[24] Siwei Wang, Xinwang Liu, Miaomiao Li, Huiying Xu, Xinzhong Zhu, Feng Gao, and En Zhu. 2021. Late Fusion Multi-view Clustering via Global and Local Alignment Maximization. *IEEE Transactions on Image Processing* (12 2021).

[25] Siwei Wang, Xinwang Liu, En Zhu, Chang Tang, Jiyuan Liu, Jingtao Hu, Jingyuan Xia, and Jianping Yin. 2019. Multi-view Clustering via Late Fusion Alignment Maximization. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 3778–3784. https://doi.org/10.24963/ijcai.2019/524

[26] Yaqiang Yao, Yang Li, Bingbing Jiang, and Huanhuan Chen. 2021. Multiple Kernel k-Means Clustering by Selecting Representative Kernels. *IEEE Transactions on Neural Networks and Learning Systems* 32, 11 (2021), 4983–4996. https://doi.org/10.1109/TNNLS.2020.3026532

[27] Hongwei Yin, Wenjun Hu, Zhao Zhang, Jungang Lou, and Minmin Miao. 2021. Incremental multi-view spectral clustering with sparse and connected graph learning. *Neural Networks* 144 (2021), 260–270. https://doi.org/10.1016/j.neunet.2021.08.031

[28] Shi Yu, Leon Tranchevent, Xinhai Liu, Wolfgang Glanzel, Johan A.K. Suykens, Bart De Moor, and Yves Moreau. 2012. Optimized Data Fusion for Kernel k-Means Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, 5 (2012), 1031–1039. https://doi.org/10.1109/TPAMI.2011.255

[29] Tiejian Zhang, Xinwang Liu, Lei Gong, Siwei Wang, Xin Niu, and Li Shen. 2021. Late Fusion Multiple Kernel Clustering with Local Kernel Alignment Maximization. *IEEE Transactions on Multimedia* (2021), 1–1. https://doi.org/10.1109/TMM.2021.3136094

[30] Yi Zhang, Xinwang Liu, Jiyuan Liu, Sisi Dai, Changwang Zhang, Kai Xu, and En Zhu. 2022. Fusion Multiple Kernel K-means. (2022).

[31] Yi Zhang, Xinwang Liu, Siwei Wang, Jiyuan Liu, Sisi Dai, and En Zhu. 2021. One-Stage Incomplete Multi-view Clustering via Late Fusion. In *Proceedings of the 29th ACM International Conference on Multimedia*. 2717–2725.