A Cluster-Weighted Kernel *K*-Means Method for Multi-View Clustering

Jing Liu,[1,2] **Fuyuan Cao,**[1] **Xiao-Zhi Gao,**[3] **Liqin Yu,**[1] **Jiye Liang**[1,*]
[1]School of Computer and Information Technology, Shanxi University, Taiyuan 030006, P.R. China
[2]School of Software, Shanxi Agricultural University, Taigu 030801, P.R. China
[3]School of Computing, University of Eastern Finland, Kuopio 70211, Finland
{jingliu_sxu, liqinyu_sxu}@hotmail.com, {cfy, ljy}@sxu.edu.cn, xiao-zhi.gao@uef.fi

Abstract

Clustering by jointly exploiting information from multiple views can yield better performance than clustering on one single view. Some existing multi-view clustering methods aim at learning a weight for each view to determine its contribution to the final solution. However, the view-weighted scheme can only indicate the overall importance of a view, which fails to recognize the importance of each inner cluster of a view. A view with higher weight cannot guarantee all clusters in this view have higher importance than them in other views. In this paper, we propose a cluster-weighted kernel k-means method for multi-view clustering. Each inner cluster of each view is assigned a weight, which is learned based on the intra-cluster similarity of the cluster compared with all its corresponding clusters in different views, to make the cluster with higher intra-cluster similarity have a higher weight among the corresponding clusters. The cluster labels are learned simultaneously with the cluster weights in an alternative updating way, by minimizing the weighted sum-of-squared errors of the kernel k-means. Compared with the view-weighted scheme, the cluster-weighted scheme enhances the interpretability for the clustering results. Experimental results on both synthetic and real data sets demonstrate the effectiveness of the proposed method.

1 Introduction

Multi-view data widely exist in real-world applications, where the same set of instances are represented by multiple distinct feature sets from different perspectives. For example, images can be described by different visual descriptors; documents may be translated into various languages; and patients are diagnosed by several types of medical examinations. These heterogeneous views usually have consistent as well as complementary information with each other, which can be simultaneously learned to get better performance than learning one single view.

Multi-view clustering has gained much attention in recent years (Chao, Sun, and Bi 2017). It assumes that different views have a common clustering partition, which means the corresponding instances in different views belong to the same cluster. Simply concatenating features from different views into a single one clustered by traditional clustering algorithms often results in poor performance, since it ignores the heterogeneity of different feature spaces and may lead to dimension curse. Most existing multi-view algorithms tend to obtain a common clustering partition by jointly exploiting information of multiple views without breaking the inherent structure of each view. The general idea of these algorithms is to guarantee the consistency among different views by using common cluster discrimination information, which can be expressed by common eigenvector matrix for multi-view spectral clustering (Kumar and Daumé 2011; Kumar, Rai, and Daumé 2011; Li et al. 2015; Nie, Li, and Li 2016), common coefficient matrix for multi-view subspace clustering (Yin et al. 2015; Gao et al. 2015; Wang et al. 2016), and common indicator matrix for multi-view nonnegative matrix factorization clustering (Akata, Thurau, and Bauckhage 2011; Liu et al. 2013; Qian et al. 2016) and multi-view k-type clustering (Tzortzis and Likas 2012; Cai, Nie, and Huang 2013; Xu, Wang, and Lai 2016).

In some cases, the low-quality views (views with high clustering loss under the common clustering partition) may degrade the performance if equally using all available views. To determine the contribution of different views to the final clustering, many view-weighted methods for learning a weight for each view have been proposed. Some methods (Tzortzis and Likas 2012; Xia et al. 2010; Li et al. 2015) multiply each view with a weight factor, and the distribution of the weights is controlled by an extra hyper-parameter. Some methods (Nie, Li, and Li 2016; 2017; Huang, Kang, and Xu 2018) use a self-weighted scheme to automatically learn a weight for each view without introducing any extra hyper-parameter. Xu, Wang, and Lai (2016) proposed a method to jointly learn the view weights as well as the feature weights for high-dimensional feature selection. Xu, Tao, and Xu (2015) proposed a self-paced smoothed weighting scheme that dynamically assigns weights to views in clustering process for gradually training from 'easy' to 'complex' views. In general, most existing view-weighted methods determine the weight for each view

*Corresponding author: Jiye Liang. Email: ljy@sxu.edu.cn.

according to the clustering loss of each view, to make the view with lower loss possess a higher weight.

However, the view-weighted scheme can only reflect the overall quality of a view, which is like a 'black box' that cannot discover more detailed information within the view. In real data sets, different clusters in a view may quite vary in the degree of intra-cluster similarity, a view with lower loss cannot guarantee all clusters in this view are partitioned better than them in other views. The intra-cluster similarity of the corresponding clusters can be lower in some views while higher in others. Roughly assigning the weight to each view by regarding each view as a whole cannot recognize the importance of each inner cluster of a view. To deal with this problem, in this paper, we propose a Cluster-Weighted Kernel K-Means method for multi-view clustering (CWK^{2}M), which, as far as we know, is the first exploration on the cluster-weighted scheme in multi-view clustering. Instead of assigning the weight to the whole view, the weight is assigned to each inner cluster of the view, which determines the contribution of each inner cluster to the final solution. The cluster weight is learned based on the intra-cluster similarity of this cluster compared with all its corresponding clusters in different views. The kernel k-means is applied to obtain a common clustering partition for different views, by minimizing the weighted sum-of-squared errors in high-dimensional space, with keeping the weights of the corresponding clusters among different views summed to one. The cluster weights are learned together with the cluster labels in an alternative updating way, derived as a closed-form solution by the Lagrangian multiplier method. Compared with the view-weighted scheme, the cluster-weighted scheme enhances the interpretability for the clustering results. Experimental results on both synthetic and real-world data sets demonstrate the effectiveness of our method.

The rest of this paper is organized as follows: We review the fundamentals of the kernel k-means and the weighted multi-view kernel k-means in Section 2. Details of our proposed method are presented in Section 3. Experimental results are shown in Section 4. Finally, we conclude our contribution and point out the further work in Section 5.

2 Background and Notations

In this section, we first introduce the theory and notations of the kernel k-means algorithm, and then revisit the multi-view kernel k-means based on view-weighted scheme.

2.1 Overview of Kernel k-Means

The k-means clustering algorithm (MacQueen 1967) can only discover clusters that are linearly separable, but cannot work well if clusters are non-linearly separable. The kernel k-means algorithm (Scholkopf, Smola, and Müller 1998) is the kernelized version of traditional k-means, which can solve this limitation by mapping the input data into a non-linear high-dimensional feature space through a kernel function. Suppose the input data set $\{x_1, x_2, ..., x_N\} \in \mathbb{R}^d$ is aimed to be partitioned into M disjoint clusters $\{\pi_k\}_{k=1}^M$. Each data point is transformed from the input space to a reproducing kernel Hilbert space $\mathcal{H}$ by a non-linear mapping $\phi : \mathbb{R}^d \mapsto \mathcal{H}$, then the k-means is applied on the high-dimensional mappings $\{\phi(x_1), \phi(x_2), ..., \phi(x_N)\}$ to obtain a non-linear partition, by minimizing the sum of squared Euclidean distances between each mapping and its closest cluster center. The objective function of the kernel k-means is given by

$$\begin{aligned} &\min_{U,\mu_k} \sum_{k=1}^{M} \sum_{i=1}^{N} U_{ik} \|\phi(x_i) - \mu_k\|^2 \\ &s.t. U \in \{0,1\}^{N\times M}, \sum_{k=1}^{M} U_{ik} = 1, \end{aligned} \tag{1}$$

where U is the cluster indicator matrix, and μ_k is the centroid of the kth cluster, obtained by

$$\mu_k = \frac{\sum_{i=1}^{N} U_{ik}\phi(x_i)}{\sum_{i=1}^{N} U_{ik}}. \tag{2}$$

Usually the non-linear mapping $\phi(x_i)$ cannot be explicitly computed, instead, the inner product of any two mappings $\phi(x_i)^T\phi(x_j)$ can be computed by a kernel function $\mathcal{K}(x_i, x_j)$. Hence, the whole data set in high-dimensional space can be represented by a kernel matrix $K \in \mathbb{R}^{N\times N}$, with each matrix entry $K_{ij} = \mathcal{K}(x_i, x_j) = \phi(x_i)^T\phi(x_j)$ representing the pairwise inner product of any two mappings. Initially, M mappings are selected as the initial cluster centers. In the first iteration, the indicator matrix U is updated by assigning each mapping to the closest center by computing the squared Euclidean distance in the high-dimensional space

$$\|\phi(x_i) - \phi(x_j)\|^2 = K_{ii} - 2K_{ij} + K_{jj}. \tag{3}$$

In the next iteration, the cluster indicator matrix U is updated by assigning each mapping to the new closest center μ_k as follows

$$U_{ij} = \begin{cases} 1, & j = arg\min_k \|\phi(x_i) - \mu_k\|^2 \\ 0, & otherwise \end{cases}, \tag{4}$$

where the cluster center μ_k cannot be computed explicitly, but the squared Euclidean distance between $\phi(x_i)$ and μ_k can be obtained by

$$\begin{aligned} \|\phi(x_i) - \mu_k\|^2 = \quad & K_{ii} - \frac{2\sum_{j=1}^{N} U_{jk}K_{ij}}{\sum_{j=1}^{N} U_{jk}} \\ & + \frac{\sum_{l=1}^{N}\sum_{m=1}^{N} U_{lk}U_{mk}K_{lm}}{\sum_{l=1}^{N}\sum_{m=1}^{N} U_{lk}U_{mk}}. \end{aligned} \tag{5}$$

Iteratively update the cluster indicator matrix U until the algorithm converges.

2.2 Weighted Multi-View Kernel K-Means Revisit

The kernel k-means can be used in multi-view clustering. Tzortzis and Likas (2012) proposed a weighted multi-view kernel k-means clustering method. This method applies kernel k-means in the space induced by a weighted combination of each view's kernel matrix, which is derived equivalent to a weighted combination of the loss of kernel k-means in each view under a common clustering partition.

Suppose a multi-view data set is composed of V views for N instances denoted by $\{x_1^{(v)}, x_2^{(v)}, ..., x_N^{(v)}\}_{v=1}^V \in \mathbb{R}^{d^{(v)}}$, where $x_i^{(v)}$ represents the ith instance from the vth view and $d^{(v)}$ is the dimensionality of features for the vth view. The high-dimensional mappings are $\{\phi^{(v)}(x_1^{(v)}), \phi^{(v)}(x_2^{(v)}), ..., \phi^{(v)}(x_N^{(v)})\}_{v=1}^V$, which are to be partitioned into M disjoint clusters. The loss of each view is multiplied by a weight ω_v, which is learned together with a common cluster indicator matrix U, by minimize the following objective:

$$\begin{aligned} \min_{\omega_v, U, \mu_k^{(v)}} & \sum_{v=1}^{V} \omega_v^p \sum_{k=1}^{M} \sum_{i=1}^{N} U_{ik} \|\phi^{(v)}(x_i^{(v)}) - \mu_k^{(v)}\|^2 \\ & s.t. U \in \{0,1\}^{N \times M}, \sum_{k=1}^{M} U_{ik} = 1, \\ & \omega_v > 0, \sum_{v=1}^{V} \omega_v = 1, p > 1, \end{aligned} \tag{6}$$

where $\mu_k^{(v)} = \frac{\sum_{i=1}^N U_{ik} \phi^{(v)}(x_i^{(v)})}{\sum_{i=1}^N U_{ik}}$ denotes the centroid of the kth cluster in the vth view. The exponent p is a hyper-parameter used to control the distribution of the view weights. When $p \to 1$, only one best view is selected, and when $p \to \infty$, ω_v on each view tend to be equal. The weight determins the importance of each view as a whole, which fails to recognize the importance of each inner cluster of a view. For this problem, we propose a cluster-weighted scheme for multi-view clustering in next section.

3 The Proposed Methodology

To describe the cluster-weighted mechanism of our method, we generate a synthetic data set shown in Figure 1, which consists of 900 data points balanced over three clusters and represented by two views. The intra-cluster similarity of cluster 1 is low in view 1 but high in view 2, while cluster 2 is exactly opposite to cluster 1 in each view, and cluster 3 has nearly the same level of intra-cluster similarity in each view. We can observe that using each single view cannot correctly separate all three clusters by its own, which, however, may be possible if using both views to jointly learn the cluster labels. Since the overall loss of each view is on the same level, performing view-weighted multi-view clustering may assign almost equal weight to each view, which fails to differentiate the loss of a cluster in different views. We hope the corresponding cluster with higher intra-cluster similarity (cluster 1 in view 2 and cluster 2 in view 1) to have higher importance in clustering. Therefore, we propose a cluster-weighted scheme by assigning the weight to each inner cluster of each view, in order to determine the importance of each corresponding cluster across different views.

3.1 Objective Function of CWK^{2}M

In our method, views are integrated by the weighted combination of the corresponding clusters among different views. Instead of weighting each view globally, we modify the equation Eq. (6) by multiplying the weight on the sum-of-squared errors of each cluster in each view, and keep the weights of the corresponding clusters among different views summed to one. The view weight ω_v on the vth view in Eq. (6) is replaced by the view-cluster weight ω_{vk} on the kth cluster of the vth view, which yields the following objective:

$$\begin{aligned} \min_{\omega_{vk}, U, \mu_k^{(v)}} & \sum_{v=1}^{V} \sum_{k=1}^{M} \omega_{vk}^p \sum_{i=1}^{N} U_{ik} \|\phi^{(v)}(x_i^{(v)}) - \mu_k^{(v)}\|^2 \\ & s.t. \ U \in \{0,1\}^{N \times M}, \sum_{k=1}^{M} U_{ik} = 1, \\ & \omega_{vk} > 0, \sum_{v=1}^{V} \omega_{vk} = 1, \forall k, \end{aligned} \tag{7}$$

where $p > 1$, controlling the distribution of the weights on the corresponding clusters. There are M sets of corresponding clusters in total. When $p \to 1$, only the best cluster in each set of corresponding clusters is selected, and when $p \to \infty$, ω_{vk} on corresponding clusters tend to be equal. Although the best p value may be different for each set of corresponding clusters, it is too complicated for optimization to introduce different hyper-parameters for different sets. In order to simplify the optimization process, we only use one hyper-parameter p to control all M sets of corresponding clusters by selecting a p value that has the best average effect on these sets. In order to make the intra-cluster loss in different views' feature spaces to be comparable, views should be normalized by dividing each view's kernel entries $K_{ij}^{(v)}$ by the average of the pairwise squared distances between instances of this view $\sum_{i=1}^N \sum_{j=1}^N (K_{ii}^{(v)} - 2K_{ij}^{(v)} + K_{jj}^{(v)})$.

3.2 Optimization

To solve the above optimization problem, the cluster indicator matrix U and the cluster weight ω_{vk} are updated alternatively, that is, when one variable is updated, the other one is fixed. Initially, M data points are selected as the initial cluster centers, and the cluster weight ω_{vk} is initialized as $1/V$. The following sections describe the optimization process for these two variables respectively.

Updating the cluster indicator matrix U: Fixing the cluster weight ω_{vk}, each entry of U is updated by assigning each data point to the cluster with the lowest loss which is computed by a weighted sum of the squared distances between $\phi^{(v)}(x_i^{(v)})$ and $\mu_k^{(v)}$ in different views. The indicator matrix U at indices (i, j) can be obtained by

$$U_{ij} = \begin{cases} 1, & j = arg\min_k \sum_{v=1}^{V} \omega_{vk}^p \|\phi^{(v)}(x_i^{(v)}) - \mu_k^{(v)}\|^2 \\ 0, & otherwise. \end{cases}. \tag{8}$$

In the first iteration, the initial cluster centers $\mu_k^{(v)}$ are real data points, $\|\phi^{(v)}(x_i^{(v)}) - \mu_k^{(v)}\|^2$ is computed by Eq. (3) in the vth view. In the subsequent iterations, $\mu_k^{(v)} = \frac{\sum_{i=1}^N U_{ik} \phi^{(v)}(x_i^{(v)})}{\sum_{i=1}^N U_{ik}}$, and $\|\phi^{(v)}(x_i^{(v)}) - \mu_k^{(v)}\|^2$ is computed by Eq. (5) in the vth view.

Updating the cluster weight ω_{vk}: Fixing the cluster indicator matrix U, the cluster weight ω_{vk} can be updated by using the Lagrangian multiplier method. First, we denote the

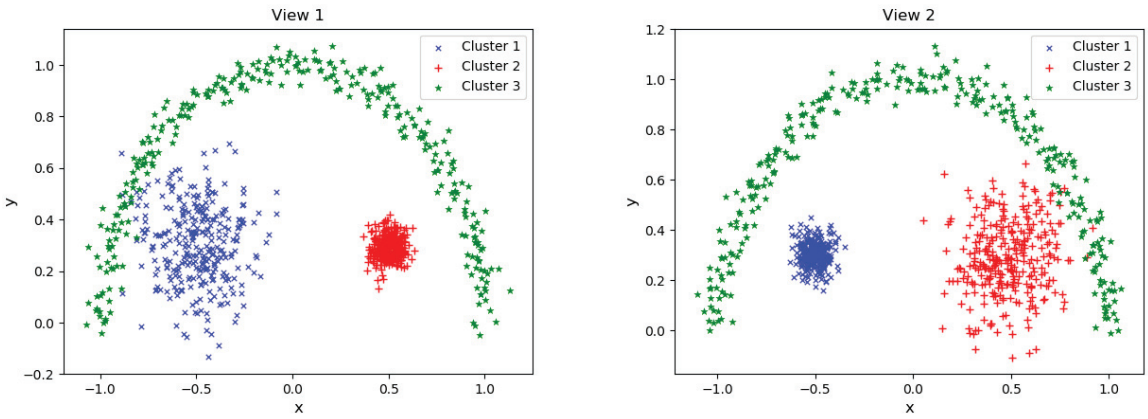


Figure 1: The synthetic data set balanced over three clusters and represented by two views.

sum-of-squared errors of the kth cluster in the vth view as

$$D_{vk} = \sum_{i=1}^{N} U_{ik} \| \phi^{(v)}(x_i^{(v)}) - \mu_k^{(v)} \|^2. \tag{9}$$

Then we get the Lagrangian formula of Eq. (7) with regard to ω_{vk} and λ_k as follows:

$$L(\omega_{vk}, \lambda_k) = \sum_{v=1}^{V} \sum_{k=1}^{M} \omega_{vk}^p D_{vk} + \sum_{k=1}^{M} \lambda_k (\sum_{v=1}^{V} \omega_{vk} - 1). \tag{10}$$

Taking derivative with regard to ω_{vk} gives

$$\frac{\partial L(\omega_{vk}, \lambda_k)}{\partial \omega_{vk}} = p \omega_{vk}^{p-1} D_{vk} + \lambda_k. \tag{11}$$

Setting this derivative to zero, we can get

$$p \omega_{vk}^{p-1} D_{vk} + \lambda_k = 0 \Rightarrow \omega_{vk} = (\frac{-\lambda_k}{p D_{vk}})^{\frac{1}{p-1}}. \tag{12}$$

Substitute Eq. (12) into the constraints $\sum_{v=1}^{V} \omega_{vk} = 1$, we have

$$(-\lambda_k)^{\frac{1}{p-1}} = \frac{1}{\sum_{v'=1}^{V} (\frac{1}{p D_{v'k}})^{\frac{1}{p-1}}}, \quad p > 1. \tag{13}$$

Taking Eq. (13) into Eq. (12) yields

$$\omega_{vk} = \frac{1}{\sum_{v'=1}^{V} (\frac{D_{vk}}{D_{v'k}})^{\frac{1}{p-1}}}, \quad p > 1, \tag{14}$$

where the smaller the cluster loss D_{vk}, the larger the cluster weight ω_{vk}. A smaller D_{vk} reflects higher intra-cluster similarity of a cluster in a view. Therefore, the cluster with higher intra-cluster similarity has larger weight in clustering.

The parameter p is used to control the distribution of weights on corresponding clusters. For any two corresponding clusters which belong to rth view and sth view respectively, from Eq.(14), $\frac{\omega_{rk}}{\omega_{sk}} = (\frac{D_{sk}}{D_{rk}})^{\frac{1}{p-1}}$ and $\frac{\omega_{rk}^p}{\omega_{sk}^p} = (\frac{D_{sk}}{D_{rk}})^{\frac{p}{p-1}}$. As p increases, the exponents $\frac{1}{p-1} \to 0$ and $\frac{p}{p-1} \to 1$, thus the ratio $\frac{\omega_{rk}}{\omega_{sk}}$ gets closer to 1, meaning that the distribution of ω_{vk} on each corresponding clusters tends to be uniform, and the ratio $\frac{\omega_{rk}^p}{\omega_{sk}^p}$ gets closer to $\frac{D_{sk}}{D_{rk}}$. Therefore, as p increase, the relative differences in D_{vk} among corresponding clusters are suppressed.

Since the first-iteration partition based on the initial centers is usually very rough, we start updating the cluster weight ω_{vk} in the second iteration (after the cluster centers change for once). In the first iteration, only the cluster indicator matrix U is computed. In the subsequent iterations, the cluster indicator matrix U and the cluster weight ω_{vk} are updated alternatively. We summarize the above optimization process in Algorithm 1.

Algorithm 1 The *CWK²M*.

1: **Input:** A multi-view data set $\{x_1^{(v)}, x_2^{(v)}, ..., x_N^{(v)}\}_{v=1}^{V}$, $x_i^{(v)} \in \mathbb{R}^{d^{(v)}}$, number of clusters M, parameter p.
2: **Output:** Cluster indicator matrix $U \in \mathbb{R}^{N \times M}$.
3: **Method:**
4: Initialze: M cluster centers, cluster weight $\omega_{vk} = 1/V$;
5: Compute U by using Eq. (8);
6: **repeat**
7: Update U by using Eq. (8);
8: Update ω_{vk} by using Eq. (14);
9: **until** Converges.

3.3 Convergence Analysis

It is known that the kernel k-means monotonically converges to a local minimum, therefore, when fixing ω_{vk}, the objective value of Eq.(7) monotonically decreases by alternatively updating U and μ_k. When fixing U and μ_k, the update on ω_{vk} further reduces the objective value, because the objective function is convex with respect to ω_{vk}, and the feasible set of the constraint on ω_{vk} is a convex set. Therefore, by solving each variable alternatively, our method will converge to a local minimum.

4 Experiments

In this section, we evaluate the performance of the proposed method on one synthetic data set and four real data sets. For the synthetic data set, the proposed method is compared with the view-weighted method MVKKM (Tzortzis and Likas 2012). For the real data sets, the proposed method is compared with four state-of-the-art methods including MVKKM. To avoid poor local solution resulting from randomly initialized cluster centers, the global kernel k-means initialization algorithm (Tzortzis and Likas 2009) is applied for both CWK^{2}M and MVKKM.

4.1 Experiments on Synthetic Data Set

We compare our cluster-weighted method (CWK^{2}M) and the view-weighted method (MVKKM) on the synthetic data set shown in Figure 1. The standard deviation of each cluster in each view is shown in Table 1, which reflects the intra-cluster similarity of each cluster. Guassian kernel $\mathcal{K}(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$ is applied for representing each view. To clearly compare the weighting mechanism of these two methods, we set $\sigma = 0.15$ for each view. We run the kernel k-means on each single view, and execute MVKKM and CWK^{2}M under various p values (p=1.2, and $p = 2^r$, r=1, 2, 3, 4, 5, 6). In both two methods, the initial centers are selected by running the global kernel k-means initialization algorithm on view 1. For the experimental results, we demonstrate the weights and the clustering performance (Normalized Mutual Information (NMI) and Accuracy (ACC)) for two methods under different p values. To better demonstrate the variation trend of ω_1^p/ω_2^p and $\omega_{1k}^p/\omega_{2k}^p$, we output the view loss ratio D_2/D_1 and the cluster loss ratio D_{2k}/D_{1k} for MVKKM and CWK^{2}M respectively .

Table 1: Standard deviation of each cluster in each view of the synthetic data set.

	View 1	View 2
Cluster 1	0.15	0.05
Cluster 2	0.05	0.15
Cluster 3	0.04	0.04

Table 2: Performance for single-view kernel k-means on the synthetic data set.

	NMI	ACC
View 1	0.8054	0.9200
View 2	0.8261	0.9289

Table 2 shows the performance of the kernel k-means on each single view. Table 3 and 4 show the weights and performance under different p values for MVKKM and CWK^{2}M, respectively. We can observe that the kernel k-means on each single view cannot correctly partition all data points by its own, and MVKKM shows even poorer performance than single-view method, while CWK^{2}M can correctly separate all data points under all p values. From Table 3, the weights on two views are very close to each other. This is because the weights are learned based on the overall loss of each view, and two views with nearly the same level of overall loss can be assigned nearly equal weights. And as p increases, ω_1 and ω_2 get to be equal, and the ratio ω_1^p/ω_2^p gets closer to D_2/D_1. For CWK^{2}M, since it can reflect the quality of the inner cluster of the view, it shows better performance than MVKKM. From Table 4, smaller p value tends to assign most portion of the weight to the good cluster (cluster 1 in view 2, and cluster 2 in view 1). As p increases, ω_{1k} and ω_{2k} for each cluster get closer to each other, but the relatively difference between ω_{1k}^p and ω_{2k}^p on cluster 1&2 is still large, and the ratio $\omega_{1k}^p/\omega_{2k}^p$ gets closer to D_{2k}/D_{1k}. Since the difference between ω_{1k}^p and ω_{2k}^p is always not lower than the difference between D_{1k} and D_{2k} when $p \to \infty$, and the difference between D_{1k} and D_{2k} for cluster 1&2 is large enough, the good clusters can always make the major contribution to the clustering results when $p \to \infty$. Therefore, CWK^{2}M can always correctly partition all data points under all p values. The distribution of ω_{vk}^p reflects the importance of each corresponding cluster in clustering, which demonstrates the interpretability of the cluster-weighted scheme.

Table 3: View weights and performance for MVKKM on the synthetic data set under different p values.

	ω_1	ω_2	ω_1^p/ω_2^p	D_2/D_1	NMI	ACC
p=1.2	0.4833	0.5167	0.9228	0.9867	0.7659	0.9156
p=2	0.4972	0.5028	0.9781	0.9890	0.7617	0.9133
p=4	0.4991	0.5009	0.9853	0.9890	0.7617	0.9133
p=8	0.4996	0.5004	0.9874	0.9890	0.7617	0.9133
p=16	0.4998	0.5002	0.9883	0.9890	0.7617	0.9133
p=32	0.4999	0.5001	0.9886	0.9890	0.7617	0.9133
p=64	0.5000	0.5000	0.9888	0.9890	0.7617	0.9133

Table 4: Cluster weights and performance for CWK^{2}M on the synthetic data set under different p values.

		ω_{1k}	ω_{2k}	$\omega_{1k}^p/\omega_{2k}^p$	D_{2k}/D_{1k}	NMI	ACC
p=1.2	Cluster 1	0.0002	0.9998	0.0000	0.1725	1.000	1.000
	Cluster 2	0.9998	0.0002	36284	5.7538		
	Cluster 3	0.5039	0.4961	1.0188	1.0031		
p=2	Cluster 1	0.1471	0.8529	0.0298	0.1725	1.000	1.000
	Cluster 2	0.8519	0.1481	33.106	5.7538		
	Cluster 3	0.5008	0.4992	1.0062	1.0031		
p=4	Cluster 1	0.3576	0.6424	0.0960	0.1725	1.000	1.000
	Cluster 2	0.6418	0.3582	10.310	5.7538		
	Cluster 3	0.5003	0.4997	1.0042	1.0031		
p=8	Cluster 1	0.4376	0.5624	0.1342	0.1725	1.000	1.000
	Cluster 2	0.5622	0.4378	7.3878	5.7538		
	Cluster 3	0.5001	0.4999	1.0036	1.0031		
p=16	Cluster 1	0.4707	0.5293	0.1535	0.1725	1.000	1.000
	Cluster 2	0.5291	0.4709	6.4657	5.7538		
	Cluster 3	0.5001	0.4999	1.0033	1.0031		
p=32	Cluster 1	0.4858	0.5142	0.1630	0.1725	1.000	1.000
	Cluster 2	0.5141	0.4859	6.0879	5.7538		
	Cluster 3	0.5000	0.5000	1.0032	1.0031		
p=64	Cluster 1	0.4930	0.5070	0.1678	0.1725	1.000	1.000
	Cluster 2	0.5069	0.4931	5.9158	5.7538		
	Cluster 3	0.5000	0.5000	1.0032	1.0031		

4.2 Experiments on Real Data Sets

Data Set Description The proposed method is further assessed on four real-world data sets as follows.

- MSRC-v1 [1]: This is an image data set consisting of 240 images over eight categories. We follow (Lee and Grauman 2009) to select seven categories including tree, building, airplane, cow, face, car, and bicycle, and each category has 30 images. Five visual features are extracted from each image to form five views: 24-dimension Color Moments (View 1), 576-dimension HOG feature (View 2), 512-dimension GIST feature (View 3), 256-dimension LBP feature (View 4), and 254 Centrist feature (View 5).
- Caltech101-7 [2]: This is a subset of Caltech101 image data set that contains 101 categories. Following (Dueck and Frey 2007), the subset containing seven categories is selected from Caltech101, including Dollar Bill, Faces, Garfiel, Motobikes, Snoopy, Stop-sign and Windor-Chair, with 441 images in total. Three visual features are extracted from each image to form three views: 254-dimension Centrist feature (View 1), 48-dimension Gabor feature (View 2) and 40-dimension wavelet moments (View 3).
- Handwritten numerals (HW) [3]: This data set consists of 2000 instances over ten digit classes from 0 to 9 digit, with 200 instances per class. The digits are represented by multiple published feature sets, and we use four of them to compose our multi-view data set. These four feature sets are 76-dimension Fourier coefficients of the character shapes (View 1), 216-dimension profile correlations (View 2), 64-dimension Karhunen-Love coefficients (View 3), and 240-dimension pixel averages in 2×3 windows (View 4).
- Reuters [4]: This is a multilingual data set consisting of documents originally written in five different languages and translated into the other four languages. All the documents are classified into six categories. We choose the documents written in English (View 1) and translated in French (View 2), German (View 3), Italian (View 4), Spain (View 5). Each language can be regarded as a view. Following (Bisson and Grimal 2012), 1200 documents are randomly sampled over six categories in a balanced manner, and 2000 words have been selected with the k-medoids algorithm [5].

Experimental Setup The proposed method is compared with the single-view kernel k-means and four other kernel-based multi-view clustering methods as follows:

- Kernel K-Means on single view **(KKM)**: Running the kernel k-means on each single view. (e.g.,KKM(1) means performing KKM on View 1.)
- Co-regularized Spectral Clustering **(CoregSC)**: A state-of-the-art multi-view clustering method by using the co-regularized idea on spectral clustering of multiple views. We set the parameter $\lambda = 0.01$ in this algorithm as suggested in their paper (Kumar, Rai, and Daumé 2011).
- Auto-weighted Multiple Graph Learning **(AWGL)**:An automatically weighted multi-view spectral clustering method for learning the view weight without introducing any extra parameters (Nie, Li, and Li 2016).
- Multi-Model Spectral Clustering **(MMSC)**: A multi-view spectral clustering algorithm that learns a commonly shared graph Laplacian matrix by unifying different views. We set the parameter $\alpha = 0.1$ in this algorithm (Cai et al. 2011).
- Multi-View Kernel K-Means **(MVKKM)**: A weighted multi-view kernel k-means method with the weight added to each view (Tzortzis and Likas 2012).

For fair comparison, the same kernel settings is applied for all methods on the same data set. We use Guassian kernel for the MSRC-v1, Caltech101-7 and HW data sets by setting the standard deviation to be the median of the pairwise Euclidean distances between instances of each view, and use linear kernel for the Reuters data set. For MVKKM and CWK^{2}M, the parameter p for each data set is searched in logarithm form ($\log_{10} p$ from 0.1 to 2 with step size 0.2), and the initial centers are selected on a single view with the best performance by the global kernel k-means initialization algorithm. Because of fixed initialization of centers for MVKKM and CWK^{2}M, these two methods run only once. The other methods are repeated for 30 times, and the average results are reported. For the experimental results, three metrics: ACC, NMI, and Adjusted Rand Index (ARI) are reported.

Table 5: Performance comparisons on MSRC-v1 data set.

Method	NMI	ACC	ARI
KKM(1)	0.2722	0.3476	0.1389
KKM(2)	0.6233	0.7143	0.4894
KKM(3)	0.6633	0.8000	0.6138
KKM(4)	0.4687	0.5905	0.3314
KKM(5)	0.5021	0.5619	0.3608
CoregSC	0.6548	0.7606	0.5752
AWGL	0.5960	0.6570	0.4760
MMSC	0.5723	0.6519	0.4567
MVKKM	0.7096	0.7762	0.6082
CWK^{2}M	**0.7451**	**0.8429**	**0.6690**

Table 6: Performance comparisons on Caltech101-7 data set.

Method	NMI	ACC	ARI
KKM(1)	0.3430	0.4512	0.2791
KKM(2)	0.2776	0.4014	0.2245
KKM(3)	0.2897	0.4036	0.2221
CoregSC	0.3503	0.4582	0.2806
AWGL	0.3240	**0.4976**	0.2812
MMSC	0.3135	0.4845	0.2817
MVKKM	0.3090	0.4150	0.2491
CWK^{2}M	**0.3513**	0.4875	**0.2961**

[1] https://www.microsoft.com/en-us/research/project/image-understanding/

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

[3] https://archive.ics.uci.edu/ml/datasets/Multiple+Features

[4] https://archive.ics.uci.edu/ml/datasets

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

Table 7: Performance comparisons on HW data set.

Method	NMI	ACC	ARI
KKM(1)	0.5932	0.5885	0.4468
KKM(2)	0.7513	0.8540	0.7044
KKM(3)	0.6977	0.7580	0.6006
KKM(4)	0.7533	0.8035	0.6884
CoregSC	0.7667	0.8278	0.7162
AWGL	0.7712	0.8169	0.7110
MMSC	0.7768	0.8334	0.7213
MVKKM	0.8684	0.9325	0.8563
CWK^{2}M	**0.8685**	**0.9325**	**0.8564**

Table 8: Performance comparisons on Reuters data set.

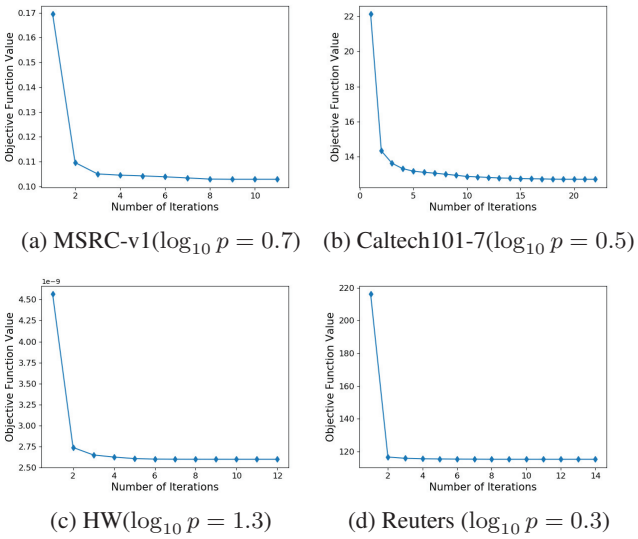
Method	NMI	ACC	ARI
KKM(1)	0.3027	0.4000	0.1553
KKM(2)	0.3054	0.3925	0.1516
KKM(3)	0.2918	0.3792	0.1507
KKM(4)	0.1545	0.1992	0.0058
KKM(5)	0.3352	0.4508	0.2020
CoregSC	0.2837	0.4556	0.1985
AWGL	0.2885	0.3990	0.1554
MMSC	0.3083	0.4457	0.1930
MVKKM	0.3556	0.4633	0.2224
CWK^{2}M	**0.3556**	**0.4633**	**0.2224**

Clustering Performance Evaluation Tables 5, 6, 7 and 8 show the performance comparisons on the four data sets. First, CWK^{2}M outperforms each single-view kernel k-means on all data sets. Second, CWK^{2}M apparently outperforms the spectral clustering based methods CoregSC, AWGL and MMSC on all data sets, except for its performance under ACC metric being slightly lower than AWGL on Caltech101-7 data set. Third, CWK^{2}M outperforms MVKKM on MSRC-v1, Caltech101-7 and HW data sets, and achieves the same performance as MVKKM on the Reuters data set, which demonstrates that the cluster-weighted scheme is more effective than the view-weighted scheme in general. In summary, CWK^{2}M demonstrates better performance than the compared methods.

Convergence Study Figure 2 shows the convergence curve of our method for each data set under the best p value. As can be seen, the objective function value decreases in each iteration and finally converges to a stable value. Since the instances of Caltech101-7 data set are unbalanced over different categories, the method converges relatively slow on this data set compared with the other three.

5 Conclusions

In this paper, we proposed a cluster-weighted kernel k-means method for multi-view clustering. Our method assigns reasonable weights to corresponding clusters among different views. The weight determines the importance of each cluster of each view to the final solution, and it is learned automatically based on the intra-cluster similarity of the cluster compared with all its corresponding clusters in different views. The kernel k-means algorithm is utilized to simultaneously learn the cluster labels as well as the cluster weights. Our method considers the weight on a more fine-grained level (cluster level) than MVKKM (view level). In this sense, the cluster-weighted scheme is more interpretable than the view-weighted scheme, which is further demonstrated in the experiment on the synthetic data set. Experimental results on four real data sets have demonstrated the superiority of our method over the state-of-the-art multi-view clustering methods. In our further work, more factors that influence the cluster weights will be considered by incorporating other constraints on the cluster weights.



(a) MSRC-v1($\log_{10} p = 0.7$) (b) Caltech101-7($\log_{10} p = 0.5$)

(c) HW($\log_{10} p = 1.3$) (d) Reuters ($\log_{10} p = 0.3$)

Figure 2: Convergence curves on four data sets.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (under grants 61573229, 61976128, 61432011, 61876103, 61773247, 61603230), the Key Research and Development (R&D) Projects of Shanxi Province (under grant 201803D31022), and the 1331 Engineering Project of Shanxi Province, China.

References

Akata, Z.; Thurau, C.; and Bauckhage, C. 2011. Non-negative matrix factorization in multimodality data for segmentation and label prediction. In *16th Computer vision winter workshop*.

Bisson, G., and Grimal, C. 2012. Co-clustering of multi-view datasets: a parallelizable approach. In *2012 IEEE 12th International Conference on Data Mining*, 828–833. IEEE.

Cai, X.; Nie, F.; Huang, H.; and Kamangar, F. 2011. Heterogeneous image feature integration via multi-modal spectral clustering. In *Computer Vision and Pattern Recognition*, 1977–1984. IEEE.

Cai, X.; Nie, F.; and Huang, H. 2013. Multi-view k-means clustering on big data. In *Proceedings of the Twenty-*

Third International Joint Conference on Artificial Intelligence, 2598–2604. AAAI Press.

Chao, G.; Sun, S.; and Bi, J. 2017. A survey on multi-view clustering. *arXiv preprint arXiv:1712.06246*.

Dueck, D., and Frey, B. J. 2007. Non-metric affinity propagation for unsupervised image categorization. In *2007 IEEE 11th International Conference on Computer Vision*, 1–8.

Gao, H.; Nie, F.; Li, X.; and Huang, H. 2015. Multi-view subspace clustering. In *Proceedings of the IEEE international conference on computer vision*, 4238–4246.

Huang, S.; Kang, Z.; and Xu, Z. 2018. Self-weighted multi-view clustering with soft capped norm. *Knowledge-Based Systems* 158:1–8.

Kumar, A., and Daumé, H. 2011. A co-training approach for multi-view spectral clustering. In *Proceedings of the 28th International Conference on Machine Learning*, 393–400. Omnipress.

Kumar, A.; Rai, P.; and Daumé, H. 2011. Co-regularized multi-view spectral clustering. In *Advances in neural information processing systems*, 1413–1421.

Lee, Y. J., and Grauman, K. 2009. Foreground focus: Unsupervised learning from partially matching images. *International Journal of Computer Vision* 85(2):143–166.

Li, Y.; Nie, F.; Huang, H.; and Huang, J. 2015. Large-scale multi-view spectral clustering via bipartite graph. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2750–2756. AAAI Press.

Liu, J.; Wang, C.; Gao, J.; and Han, J. 2013. Multi-view clustering via joint nonnegative matrix factorization. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, 252–260. SIAM.

MacQueen, J. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, 281–297. Oakland, CA, USA.

Nie, F.; Li, J.; and Li, X. 2016. Parameter-free auto-weighted multiple graph learning: A framework for multi-view clustering and semi-supervised classification. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 1881–1887. AAAI Press.

Nie, F.; Li, J.; and Li, X. 2017. Self-weighted multi-view clustering with multiple graphs. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2564–2570. AAAI Press.

Qian, B.; Shen, X.; Gu, Y.; Tang, Z.; and Ding, Y. 2016. Double constrained nmf for partial multi-view clustering. In *2016 International Conference on Digital Image Computing: Techniques and Applications*, 1–7. IEEE.

Scholkopf, B.; Smola, A.; and Müller, K.-R. 1998. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation* 10(5):1299–1319.

Tzortzis, G., and Likas, A. 2009. The global kernel k-means algorithm for clustering in feature space. *IEEE Transactions on Neural Networks* 20(7):1181–1194.

Tzortzis, G., and Likas, A. 2012. Kernel-based weighted multi-view clustering. In *2012 IEEE 12th international conference on data mining*, 675–684. IEEE.

Wang, Y.; Zhang, W.; Wu, L.; Lin, X.; Fang, M.; and Pan, S. 2016. Iterative views agreement: An iterative low-rank based structured optimization method to multi-view spectral clustering. *arXiv preprint arXiv:1608.05560*.

Xia, T.; Tao, D.; Mei, T.; and Zhang, Y. 2010. Multiview spectral embedding. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 40(6):1438–1446.

Xu, C.; Tao, D.; and Xu, C. 2015. Multi-view self-paced learning for clustering. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*. AAAI Press.

Xu, Y.-M.; Wang, C.-D.; and Lai, J.-H. 2016. Weighted multi-view clustering with feature selection. *Pattern Recognition* 53:25–35.

Yin, Q.; Wu, S.; He, R.; and Wang, L. 2015. Multi-view clustering via pairwise sparse subspace representation. *Neurocomputing* 156:12–21.