Approximate Large-scale Multiple Kernel k-means Using Deep Neural Network

Yueqing Wang, Xinwang Liu*, Yong Dou, Rongchun Li
National Laboratory for Parallel and Distributed Processing, NUDT, Changsha, China, 410073
xinwangliu@nudt.edu.cn

Abstract

Multiple kernel clustering (MKC) algorithms have been extensively studied and applied to various applications. Although they demonstrate great success in both the theoretical aspects and applications, existing MKC algorithms cannot be applied to large-scale clustering tasks due to: i) the heavy computational cost to calculate the base kernels; and ii) insufficient memory to load the kernel matrices. In this paper, we propose an approximate algorithm to overcome these issues, and to make it be applicable to large-scale applications. Specifically, our algorithm trains a deep neural network to regress the indicating matrix generated by MKC algorithms on a small subset, and then obtains the approximate indicating matrix of the whole data set using the trained network, and finally performs the k-means on the output of our network. By mapping features into indicating matrix directly, our algorithm avoids computing the full kernel matrices, which dramatically decreases the memory requirement. Extensive experiments show that our algorithm consumes less time than most comparatively similar algorithms, while it achieves comparable performance with MKC algorithms.

1 Introduction

Multiple view clustering (MVC) algorithms have been extensively studied recently because they can exploit the complementary information among multiple views of data in an unsupervised way [Zhou and Burges, 2007; Weiran *et al.*, 2015; Jinglin *et al.*, 2016; Cao *et al.*, 2015; Zhang *et al.*, 2015]. Most of these MVC algorithms adopt an underlying assumption; that is, all data points are mapped into the Euclidean, Hamming, or other simple geometric spaces, so they cannot fully exploit the nonlinear similarities among data points. Kernel based algorithms overcome this limitation by mapping samples into a reproducing kernel Hilbert space and defining the similarity using a nonlinear kernel function. This enables kernel clustering to handle the linearly non-separable problem in an input space that MVC algorithms suffer from.

*The corresponding author.

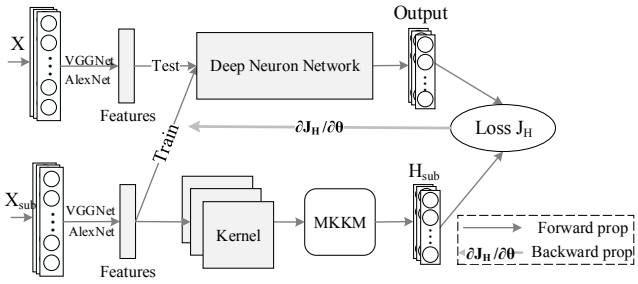

Figure 1: Architecture of our algorithm. In our algorithm, we first sample a subset from the large-scale dataset, and then use the classical multiple kernel k-means (MKKM) to obtain the indicating matrix, and train the deep neural network to regress the indicating matrix. Finally, we use the trained network to get the approximate indicating matrix for the whole dataset.

Continuing in this direction, many multiple kernel clustering (MKC) algorithms have been proposed [Zhao *et al.*, 2009; Huang *et al.*, 2012; Lu *et al.*, 2014; Xia *et al.*, 2014; Zhou *et al.*, 2015; Kumar *et al.*, 2011]. Although demonstrating promising performance, these MKC algorithms cannot be applied to large-scale applications due to the following two issues. Firstly, all MKC algorithms need to calculate the base kernels, and the computational complexity of this process which equals $\mathcal{O}(m * d^2 * N^2)$, is huge, where d, m, and N represent the dimension, the number of kernels and the number of samples, respectively. Secondly, the memory requirement of all MKC algorithms is large, which equals $\mathcal{O}(m * N^2)$. For example, when $N = 100,000$ and $m = 3$, MKC algorithms need over 100GB of running memory. The huge memory requirement cannot be satisfied even on a server. A direct remedy to address the memory issue is data blocking, which splits kernel matrices into several pieces and stores them on the hard disk. In this way, the kernel matrices stored in memory will be replaced frequently, which greatly increases the timing overhead.

To address the time and memory issues, we revisit one classical representative of MKC algorithms, called the multiple kernel k-means (MKKM), and propose an approximate algorithm to avoid computing and loading kernel matrices. To further illustrate how to reduce the running time and memory, we divide the MKKM into three stages: i) multiple kernel generation; ii) an optimized process that updates the indicat-

ing matrix $\mathbf{H}$ and coefficients γ alternately; iii) k-means on $\mathbf{H}$, which is to obtain clustering labels. The first two stages of MKKM will map the input features $\mathbf{X}$ into a kernel matrix $\mathbf{K}$, and then into the indicating matrix $\mathbf{H}$. These two stages consume too much memory and time. To reduce the time and memory requirements of MKKM, we design a deep neural network to approximate the first two stages in MKKM. Specifically, we sample a subset from the whole dataset, and then use MKKM to get the indicating matrix of the subset $\mathbf{H}_{sub}$. After that, we design a deep neural network to regress the $\mathbf{H}_{sub}$. Next, we take the whole dataset as the input of the trained network, and get the approximate indicating matrix $\tilde{\mathbf{H}}$ of the large dataset using the trained network. Finally, we perform k-means on $\tilde{\mathbf{H}}$. Note that the full kernel matrices are required in MKKM because the optimized process needs the combined kernel matrix to obtain $\mathbf{H}$. By mapping features into $\tilde{\mathbf{H}}$ directly, we avoid computing the full kernel matrix. Therefore, the computational and memory requirements of our approach are only a small portion of the MKKM, which further leads to a significant speedup of MKKM. By observing that the dimension of the features will influence the running time of k-means, we use principal component analysis (PCA) to reduce the dimension of $\tilde{\mathbf{H}}$. As a byproduct, the dimensionality reduction may also be appropriate when the $\tilde{\mathbf{H}}$ is noisy. In this way, the output of our algorithm is of better quality and lower dimension, and is more suitable for clustering.

We end up this section by summarizing the main contributions of this work as follows:

i) We propose a deep neural network to approximate the MKKM. It can avoid computing kernels and completing singular value decomposition (SVD) in MKKM by mapping the features into the indicating matrix directly, which reduces the memory and time requirements.

ii) The experimental results show that our approach and MKKM give results of comparable accuracy on several small datasets, which means our approximate approach is effective. On some large-scale datasets, the memory requirement of MKKM is too large to satisfy, while our approach also works well and achieves promising performance.

2 Related Work

We first review the related work on MKC, and then give a brief introduction to large-scale clustering algorithms.

2.1 Multiple Kernel Clustering (MKC) Algorithms

Much effort have been devoted to improving MKC from all kinds of aspects. Existing MKC algorithms can roughly be grouped into two categories.

The first category optimizes a group of kernel coefficients, and uses the combined kernel for clustering [Yu *et al.*, 2012; Cortes *et al.*, 2012; Gönen and Margolin, 2014; Du *et al.*, 2015; Lu *et al.*, 2014; Liu *et al.*, 2016]. In [Yu *et al.*, 2012], the authors propose an MKKM algorithm that externs kernel k-means from a single kernel to multiple kernels. The work in [Gönen and Margolin, 2014] offers a similar algorithm with the difference being that the kernels are combined in a localized way to better capture their individual characteristics. In [Lu *et al.*, 2014], kernel alignment maximization is employed to jointly perform the k-means clustering and MKL. The algorithm described in [Cortes *et al.*, 2012], called co-regularized spectral clustering (CRSC), provides a co-regularization method to perform spectral clustering. Instead of explicitly optimizing the combined kernel, it learns the indicating matrix $\mathbf{H}$ directly. To reduce the redundancy and enforce the diversity of the selected kernels, MKKM clustering with matrix-induced regularization (MKKM-MR) is proposed [Liu *et al.*, 2016]. MKKM-MR uses a regularization term that is able to characterize the correlation of each pair of kernels.

The second category learns a consensus matrix via low-rank optimization [Xia *et al.*, 2014; Zhou *et al.*, 2015; Kumar *et al.*, 2011]. In [Xia *et al.*, 2014], they firstly construct a transition probability matrix from each single view, and use these matrices to recover a shared low-rank transition probability matrix as an input to the standard Markov chain method for clustering. Another work [Zhou *et al.*, 2015] presents to capturing the structures of noises in each kernel, and integrating them into a robust and consensus framework to learn a low-rank matrix. In [Kumar *et al.*, 2011], the clustering is learned in one view and is then used to label data in other views to modify a similarity matrix.

Although these algorithms achieve promising performance, they cannot be applied to large-scale datasets due to i)the heavy computational cost to calculate the kernels; and ii) insufficient memory to store the kernel matrices and optimize the combined kernel.

2.2 Large-scale Clustering Algorithms

To efficiently cluster large-scale datasets, a number of algorithms have been developed. Some methods aim to improve the scalability of clustering algorithms [Cai *et al.*, 2013; Chen and Cai, 2011; Zhang and Lu, 2016; Wang *et al.*, 2016]. In [Cai *et al.*, 2013], they put forward a new, robust, large-scale multi-view clustering method to integrate heterogeneous representations of large-scale data. The work in [Wang *et al.*, 2016] proposes three new nonnegative matrix factorization (NMF) and nonnegative matrix tri-factorization (NMTF) models, which are robust to outliers. To avoid the presence of noise in the large-scale data, a large-scale sparse clustering (LSSC) algorithm is detailed in [Zhang and Lu, 2016].

Other algorithms have been proposed to reduce the high computational cost of clustering [Gong *et al.*, 2015; Li *et al.*, 2015; Zhang and Lu, 2016; Shen *et al.*, 2017]. In [Gong *et al.*, 2015], a fast binary k-means algorithm is presented that works directly on the similarity-preserving hashes of images and clusters them into binary centers. The work [Li *et al.*, 2015] offers a large-scale multiple view spectral clustering approach based on a bipartite graph. Compressed k-means (CKM) is suggested in [Shen *et al.*, 2017] which significantly reduces the memory and time requirements by mapping data points as binary codes, so it is well suited to fast clustering.

Most existing large-scale clustering methods can only achieve mediocre performance because i) they always adopt approximate ways to speedup the clustering process; and ii)

they do not use kernel functions to map data points into reproduced kernel Hilbert spaces, so they cannot fully exploit the non-linear similarity among data points, which limits the clustering performance. In our work, we propose a novel approach to approximate MKKM, which achieves the comparable performance to MKKM, while it consumes less time and memory than most contrasted algorithms.

3 Approximate Large-scale Multiple Kernel k-means Using Deep Neural Network

3.1 Notations and Background

In this part, we introduce some of the notations used in our paper. Let $\{\mathbf{x}_i\}_{i=1}^n \subseteq \mathcal{X}$ be a collection of n samples, and $\phi(\cdot) : \mathbf{x} \in \mathcal{X} \mapsto \mathcal{H}$ be a feature mapping which maps $\mathbf{x}$ onto a reproducing kernel Hilbert space $\mathcal{H}$. In the multiple kernel setting, each sample has multiple feature representations via a group of feature mappings $\{\phi_p(\cdot)\}_{p=1}^m$. Specifically, each sample is represented as $\phi_{\boldsymbol{\gamma}}(\mathbf{x}) = [\gamma_1\phi_1(\mathbf{x})^\top, \gamma_2\phi_2(\mathbf{x})^\top, \cdots, \gamma_m\phi_m(\mathbf{x})^\top]^\top$, where $\boldsymbol{\gamma} = [\gamma_1, \gamma_2, \cdots, \gamma_m]^\top$ denotes the coefficients of each base kernel. Correspondingly, the kernel function over the above mapping function can be written as:

$$\kappa_{\boldsymbol{\gamma}}(\mathbf{x}_i, \mathbf{x}_j) = \phi_{\boldsymbol{\gamma}}(\mathbf{x}_i)^\top \phi_{\boldsymbol{\gamma}}(\mathbf{x}_j) = \sum_{p=1}^{m} \gamma_p^2 \kappa_p(\mathbf{x}_i, \mathbf{x}_j). \quad (1)$$

Given the combined kernel matrix $\mathbf{K}_{\boldsymbol{\gamma}}$, the optimized problem for MKKM can be written as,

$$\begin{aligned} \min_{\mathbf{H}\in\mathbb{R}^{n\times k}, \boldsymbol{\gamma}} \ & \mathrm{Tr}(\mathbf{K}_{\boldsymbol{\gamma}}(\mathbf{I}_n - \mathbf{H}\mathbf{H}^\top)) \\ s.t.\ & \mathbf{H}^\top\mathbf{H} = \mathbf{I}_k,\ \boldsymbol{\gamma}^\top\mathbf{1}_m = 1,\ \boldsymbol{\gamma} \succeq \mathbf{0}, \end{aligned} \quad (2)$$

where $\mathbf{H}$ is the indicating matrix of all data. The problem described in Eq.(2) can be solved by alternatively updating $\mathbf{H}$ and $\boldsymbol{\gamma}$.

The memory requirement of MKKM is at least $\mathcal{O}(m*N^2)$, and the computational complexity of it equals $\mathcal{O}(N^3)$ due to the SVD. These two issues limit the MKKM to being applied to large-scale tasks.

3.2 Approximate Large-scale Multiple Kernel k-means Using Deep Neural Network

In our paper, we use a deep neural network to approximate MKKM, which can achieve comparable performance to MKKM. Our network includes one 1D convolutional layer, one max pooling layer, and four fully connected layers. After the pooling layer, we concatenate the multiple features and send the combined feature into the fully connected layer. By combining multiple views of data, our network can achieve better performance.

In our implementation, we fix the network structure except for the output layer. The dimension of our output layer is decided by the number of classes. For example, the handwritten digital dataset has 10 classes, so the dimension of our output layer equals 10.

There are two stages in our approximate algorithm. In the first stage, we sample data from the whole dataset and obtain multiple views of the data. To facilitate effective feature learning, all views are normalized into the same unit box. Then, multiple kernels of the subset are generated and MKKM is performed on these kernels to obtain the $\mathbf{H}_{sub}$, which is the target of our network. After that, we train a deep neural network to regress $\mathbf{H}_{sub}$, as shown in Figure 2. The corresponding loss function of our network can be written as follows:

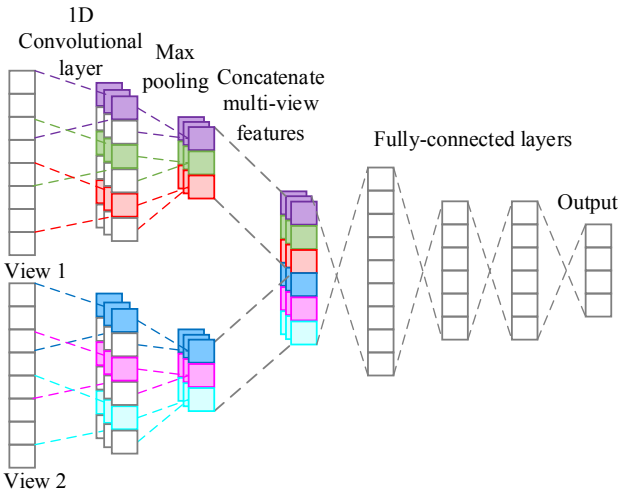

Figure 2: The network structure used in our work. It includes one convolutional layer, one pooling layer, and four fully connected layers.

$$J_{\mathbf{H}}(\theta) = \|f_\theta(\mathbf{X}) - \mathbf{H}_{sub}\|^2 + \|\theta\|_F, \quad (3)$$

Where θ is the parameters of our network and $f_\theta(\mathbf{X})$ is the output of our network, which equals $g_n(\cdots g_2(\mathbf{W}_2^\top(g_1(\mathbf{W}_1^\top * \mathbf{X} + \mathbf{b}_1)) + \mathbf{b}_2)\cdots)$. g_l, $\mathbf{W}_l$ and $\mathbf{b}_l$ denote the activating function, weight matrix and bias vector of the l-th layer, respectively. To regress $\mathbf{H}_{sub}$, we use stochastic gradient descent (SGD) method to minimize the loss function. The training stage is shown in Algorithm 1.

Algorithm 1: Training stage

Input: $\mathbf{X}_{sub}$
Output: $\boldsymbol{\theta}$

```
Generate multiple kernels {K_i}_{i=1}^m with X_sub;
Compute indicating matrix H_sub using MKKM;
Initialize θ randomly;
repeat
    Complete forward pass to get the output f_θ(X_sub);
    Perform backward pass to compute ∂J_H/∂θ;
    Update θ;
until Convergence;
```

After the training stage, our approach is performed on the whole datasets, as shown in Algorithm 2. In this stage, we fix the deep neural network, and send all data points into it to obtain the approximate indicating matrix $\mathbf{H}_o$. To further decrease the dimension of the indicating matrix, we adopt PCA, and get $\tilde{\mathbf{H}}$. Finally, we perform k-means on $\tilde{\mathbf{H}}$ to obtain the clustering results $\tilde{\mathbf{Y}}$.

Algorithm 2: Testing stage

Input: $\mathbf{X}$, $\boldsymbol{\theta}$
Output: $\tilde{\mathbf{H}}$, $\tilde{\mathbf{Y}}$

Compute $\mathbf{H}_o = f_\theta(\mathbf{X})$;
Perform PCA on $\mathbf{H}_o$ and obtain $\tilde{\mathbf{H}}$;
Perform k-means on $\tilde{\mathbf{H}}$ to obtain labels $\tilde{\mathbf{Y}}$.

3.3 Discussion

Although it includes a training stage and testing stage, our algorithm is an unsupervised algorithm. In fact, the "label" used in our training stage is obtained in an unsupervised way. By using the indicating matrix $\mathbf{H}_{sub}$ of the subset, the performance of our algorithm is nearly identical to that of MKKM. Moreover, when performed on large-scale datasets, our algorithm does not need to compute kernels and perform SVD, which indicates the time and memory requirements are only a small portion of MKKM, so our algorithm is more suitable for large-scale datasets than MKKM. It is well understood that the sampling strategy is crucial to the performance of the proposed algorithm. In fact, sampling strategy is an important area of machine learning research. In our paper, we do not consider the influence of other strategies, and only adopt the random uniform sampling without replacement.

4 Experimental Results

4.1 Datasets and Experimental Settings

We evaluate our algorithm on eight datasets detailed in Table 1. To compare the performance of our algorithm with MKKM, we use four small-scale datasets, such as mnist-10k, cifar100-10k, Oxford 102 Category Flowers (102flowers), and birds200, for which the numbers of samples are fewer than 15,000. To display the performance of our algorithm on large-scale datasets, we use four datasets, such as Caltech256, cifar100, mnist, and ImageNet, for which the numbers of samples are greater than 30000. It is worth mentioning that ImageNet has more than 1.2 million samples. For all datasets, we use two 4096-dimensional features extracted using the Alexnet model [Krizhevsky *et al.*, 2012] and Visual Geometry Group-19 (VGG19) model [Simonyan and Zisserman, 2014] to represent the images. We also generate six kernels, including two linear kernels and four Gaussian kernels for several small datasets, such as 102flowers, mnist-10k, cifar100-10k and birds200. Furthermore, to test the performance of all algorithms with respect to the number of samples, we generate ten subsets by randomly selecting 1,000, 2,000, $\cdots$, and 10,000 samples on Caltech256, cifar100 and birds200. The widely used clustering accuracy (ACC), normalized mutual information (NMI) and purity are applied in our paper to evaluate the clustering performance. For all algorithms, we repeat each experiment 20 times with random initialization to reduce the effect of randomness caused by k-means, and report the mean result.

Figure 3 shows some samples for 102flowers, ImageNet, and birds200. All the algorithms reported this paper are performed on a workstation with a 32-core Intel E5-2650 2.00 GHz processor and 256 GB memory.

Table 1: Datasets used in our experiments.

Datasets	#Samples	#Classes	#Views	#Kernels
102flowers	8196	102	2	6
mnist-10k	10000	10	2	6
cifar100-10k	10000	100	2	6
birds200	11788	200	2	6
Caltech256	30607	257	2	-
cifar100	50000	100	2	-
mnist	60000	10	2	-
ImageNet	1281167	1000	2	-

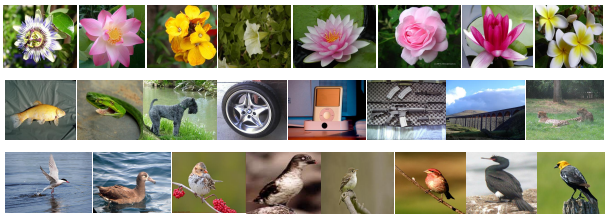

Figure 3: Samples from 102flowers, ImageNet and birds200.

4.2 Compared Algorithms

- **k-means**. This is the conventional k-means method based on Euclidean distance. It is the baseline method in our work.
- **PCA+k-means**. We perform a PCA over the samples and use the coefficient matrix as new features. We choose the first p principal components whose cumulative percent is greater than 0.99.
- **LSC-k** [Chen and Cai, 2011]. The landmark-based large scale spectral clustering method using k-means for landmark selection.
- **Robust multi-view k-means clustering (RMKMC** [Cai *et al.*, 2013]. RMKMC can be easily parallelized and performed on multi-core processors for big data clustering. Moreover, it is robust to data outliers.
- **Auto-weighted multiple graph learning (AMGL)** [Nie *et al.*, 2016]. AMGL learns a set of weights automatically for all views and does not need any parameters.
- **Multi-view learning with adaptive neighbors (MLAN)** [Nie *et al.*, 2017]. It performs multi-view clustering and local manifold structure learning simultaneously. Similar to AMGL, it has no explicit weight parameters.
- **CKM** [Shen *et al.*, 2017]. CKM aims to jointly learn binary codes and clusters. The advantages of CKM over conventional clustering methods is the low computational cost and storage.
- **MKKM** [Huang *et al.*, 2012]: The algorithm alternatively performs kernel k-means and updates the kernel coefficients, as introduced in the related work. It can be seen as the baseline of our algorithm. Note that this algorithm can only be performed on small datasets due to its heavy memory requirement.

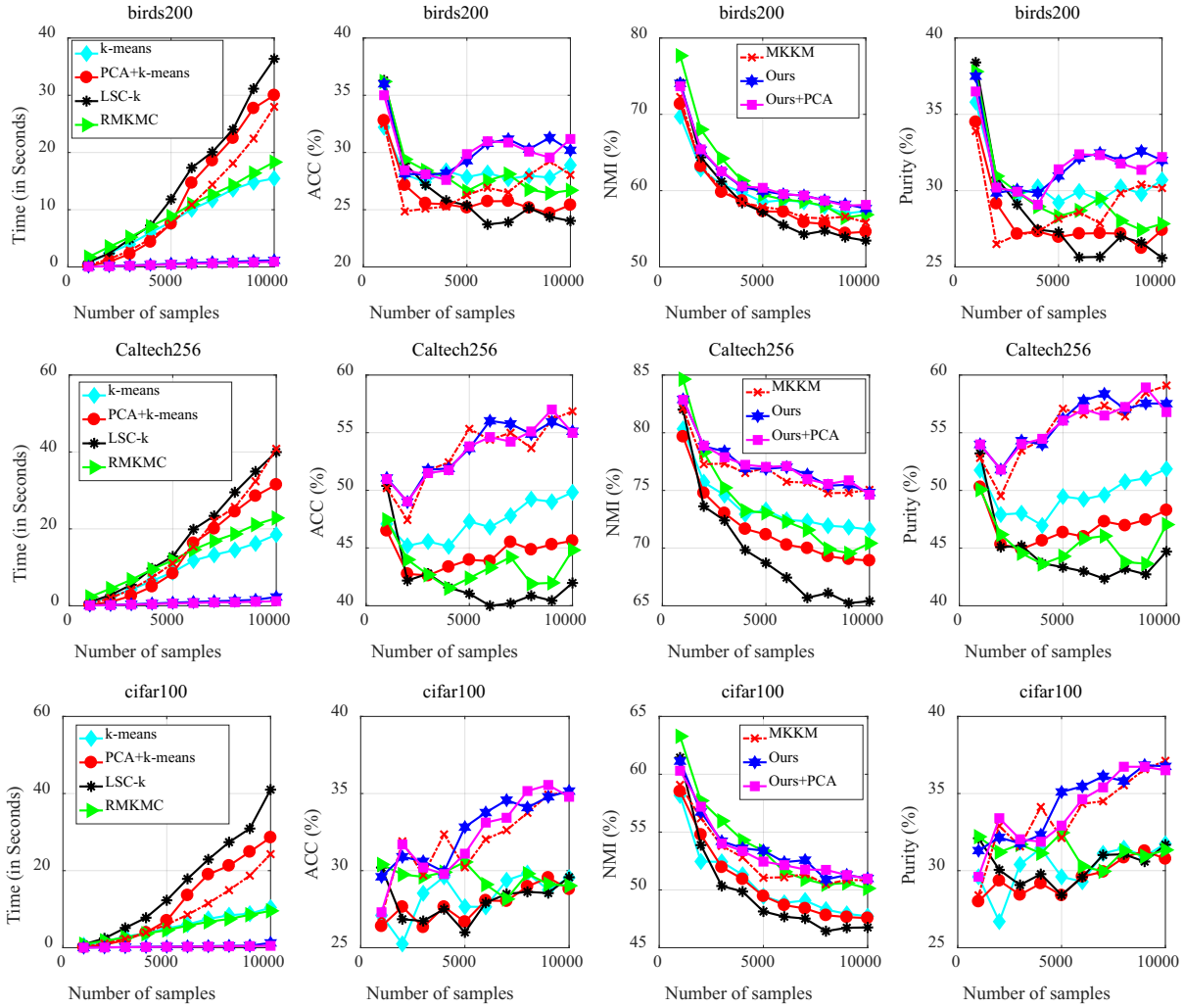

Figure 4: Clustering and timing performance with different sample sizes on birds200, Caltech256 and cifar100.

4.3 Experimental Results

Performance on small datasets. Compared with alternative clustering algorithms, our algorithm consumes 1.26s, 14.3s, 0.76s, and 0.49s for clustering, and reports 35.2%, 31.34%, 55.99%, and 70.78% clustering accuracy on cifar100-10k, birds200, 102flowers and mnist-10k, respectively. By contrast, MKKM takes 24.2s, 61.3s, 16s, and 10.1s, and achieves comparative clustering accuracy. We also contrast our algorithm with six other algorithms: k-means, PCA+k-means, LSC-k, RMKMC, AMGL and MLAN. Figure 5 displays the performance on small datasets of all algorithms. Although AMGL and MLAN achieve the best accuracy on mnist-10k (91.99% and 88.12%, respectively), they consume too much time, which means they should not be applied to large-scale datasets. Furthermore, they only show the best performance on this dataset; however, on other three datasets, our two algorithms show better performance while costing less time.

Performance on large datasets. Table 2 reports the clustering performance and timing performance of all the aforementioned algorithms. It can be observed that our algorithms achieve the best performance on all data sets, while needing relatively less running time. On large-scale dataset, such as ImageNet, our algorithms achieve an accuracy of 38.84% and 38.92%, and the speed up compared to k-means are 3.3 and 11.05, which means our algorithms are more suitable for large-scale datasets. This indicates that our approximate algorithms can be applied to estimate the performance of MKKM on large-scale datasets. In addition, our algorithms would not lose too much performance after reducing the dimension; for example, by using PCA, the performance of our algorithm decreases by an average of less than 1%. These observations show that the proposed clustering algorithms are effective and efficient.

Performance with different numbers of training samples. As mentioned before, our algorithm has a training stage and testing stage. In the training stage, we sample a subset from the whole data set. In this section, we report on the clustering performance using different sizes of training set. In order to verify the effect of the size of the training set on the final clustering performance, we change the size of the training set and train different networks, then test these networks

Table 2: ACC, NMI, purity, and time comparison of different clustering algorithms on all datasets.

	mnist	cifar100	102flowers	birds200	caltech256	ImageNet
ACC (%)						
k-means	64.48	31.32	50.71	29.54	48.63	36.28
PCA+k-means	54.38	28.82	48.61	25.15	44.52	34.80
LSC-k	**78.69**	30.54	49.13	24.52	41.28	30.37
RMKMC	-	-	36.05	20.80	41.50	-
AMGL	-	-	53.65	22.32	41.76	-
MLAN	-	-	52.50	30.81	36.98	-
CKM	60.57	30.14	49.28	27.56	48.02	33.21
MKKM	-	-	51.72	29.37	-	-
Ours	70.22	**37.37**	55.79	**31.34**	54.30	38.84
Ours+PCA	70.18	37.15	**55.99**	31.33	**54.45**	**38.92**
NMI (%)						
	mnist	cifar100	102flowers	birds200	caltech256	ImageNet
k-means	62.68	44.86	67.38	57.07	70.18	67.93
PCA+k-means	49.64	43.62	66.14	53.70	66.81	65.92
LSC-k	**79.80**	44.84	68.14	53.05	63.54	58.41
RMKMC	-	-	57.33	50.10	62.12	-
AMGL	-	-	65.65	42.18	63.10	-
MLAN	-	-	62.15	56.82	61.73	-
CKM	61.43	44.14	66.73	55.24	69.83	64.15
MKKM	-	-	65.34	55.50	-	-
Ours	66.61	**48.09**	**69.34**	57.43	72.09	**68.20**
Ours+PCA	66.49	47.99	68.18	**57.44**	**72.14**	67.71
Purity (%)						
	mnist	cifar100	102flowers	birds200	caltech256	ImageNet
k-means	69.92	32.72	56.94	30.81	57.95	36.68
PCA+k-means	55.41	30.37	55.72	27.04	53.80	36.49
LSC-k	**83.83**	32.97	58.32	26.15	49.08	29.03
RMKMC	-	-	43.01	21.98	48.96	-
AMGL	-	-	59.83	25.14	47.35	-
MLAN	-	-	58.22	31.01	47.09	-
CKM	67.85	30.18	54.91	28.89	56.87	33.83
MKKM	-	-	56.99	30.85	-	-
Ours	75.64	**38.64**	**62.44**	**33.16**	61.26	39.44
Ours+PCA	75.58	38.37	60.85	32.72	**61.31**	**40.40**
Time (s)						
	mnist	cifar100	102flowers	birds200	caltech256	ImageNet
k-means	229.22	281.55	52.30	211.31	332.74	9762.00
PCA+k-means	1575.01	552.30	257.86	394.13	427.43	14459.00
LSC-k	5309.12	3684.37	345.08	497.01	1670.74	32321.80
RMKMC	-	-	2694.88	9242.36	37359	-
AMGL	-	-	1377.23	5734.74	~2 days	-
MLAN	-	-	1312.25	2727.96	77897	-
CKM	13.25	15.36	4.96	18.05	95.7	508
MKKM	-	-	160.66	613.12	-	-
Ours	41.91	47.43	7.64	143.84	696	2957.90
Ours+PCA	17.70	28.63	6.74	32.69	620.9	883.81

Table 3: ACC, NMI, purity, and time comparison of different clustering algorithms on all datasets.

	mnist	cifar100	102flowers	birds200	Caltech256	ILSVRC2012
k-means	47.60	20.78	44.24	19.38	33.83	22.98
LSC-k	**72.16**	24.49	47.27	19.30	32.87	16.47
PCA+k-means	48.63	22.86	45.16	19.99	37.56	25.32
CKM	50.12	20.87	44.11	19.29	32.96	23.58
Ours	68.05	**27.92**	**54.46**	**20.17**	41.00	28.90
Ours+PCA	67.57	26.74	49.70	19.55	**41.45**	**29.06**

on three datasets. Figure 6 presents the results. From this figure, we can see that the accuracy, NMI, and purity have no significant changes when the size of the training set changes, which indicates that our algorithm is stable.

Performance with different sizes of datasets. Figure 4 displays the performance of our algorithm on subsets with different number of samples. It is observed that i) the running time of the compared algorithms dramatically increases when the numbers of samples is varied, while ours increase slightly; ii) there are no fixed rules between clustering performance and the number of samples; and iii) MKKM and our algorithm always achieve the best performance with different numbers of samples.

Performance of single-view data. To display the advantage of multi-view, we report the clustering accuracy of single-view on all datasets in Table 3.

From Table 3, we can conclude that i) our algorithms achieve better performance on single view datasets; and i) the performance of multi-view data is higher than that of single-view.

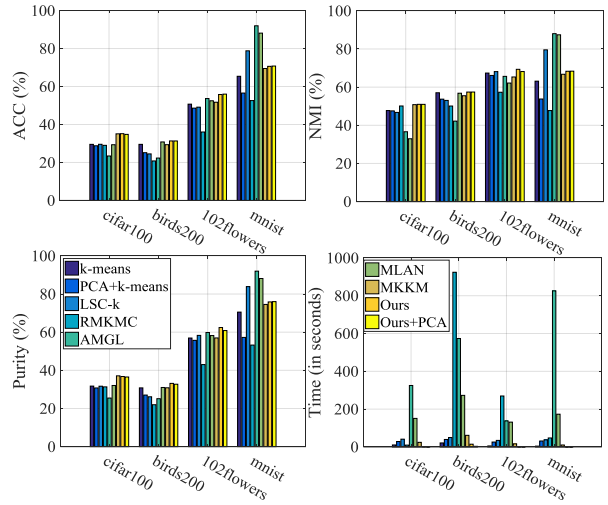

Figure 5: Performance on several small datasets.

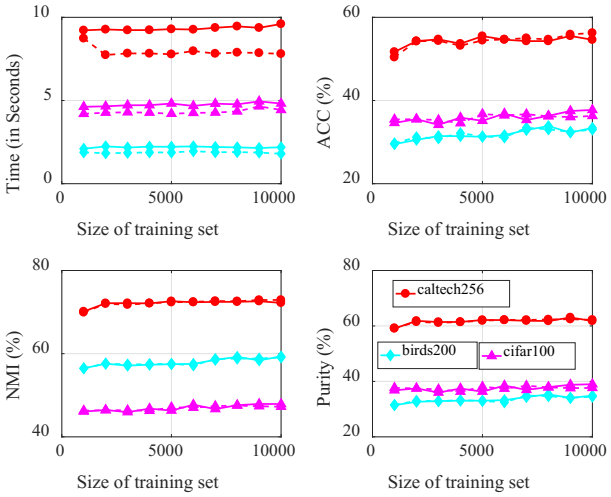

Figure 6: Clustering performance variance with changes to the size of the training set.

5 Conclusion

In this paper, we propose a novel algorithm to overcome timing and memory issues. It achieves comparative performance with MKC algorithms. In particular, our algorithm trains a deep neural network to regress the indicating matrix generated by MKC algorithms on a small subset, then obtains the approximate indicating matrix of the whole dataset using the trained network, and performs the k-means on the output of our network. To further reduce the execution time of our algorithm, we add the PCA to our network to descend the dimension of output. Extensive experiments show that our algorithm is effective and efficient.

Acknowledgments

This work was supported in part by the Natural Science Foundation of China (61403405, U1435219, 61402507).

References

[Cai *et al.*, 2013] Xiao Cai, Feiping Nie, and Heng Huang. Multi-view k-means clustering on big data. In *IJCAI*, pages 2598–2604, 2013.

[Cao *et al.*, 2015] Xiaochun Cao, Changqing Zhang, Huazhu Fu, and Si Liu. Diversity-induced multi-view subspace clustering. In *CVPR*, pages 586–594, 2015.

[Chen and Cai, 2011] Xinlei Chen and Deng Cai. Large scale spectral clustering with landmark-based representation. In *AAAI*, pages 313–318, 2011.

[Cortes *et al.*, 2012] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research*, 13(1):795–828, 2012.

[Du *et al.*, 2015] Liang Du, Peng Zhou, Lei Shi, Hanmo Wang, Mingyu Fan, Wenjian Wang, and Yi Dong Shen. Robust multiple kernel k-means using $\ell_{2,1}$ -norm. In *International Conference on Artificial Intelligence*, pages 3476–3482, 2015.

[Gönen and Margolin, 2014] Mehmet Gönen and Adam A Margolin. Localized data fusion for kernel k-means clustering with application to cancer biology. *NIPS*, 2:1305–1313, 2014.

[Gong *et al.*, 2015] Yunchao Gong, Marcin Pawlowski, Fei Yang, Louis Brandy, Lubomir Boundev, and Rob Fergus. Web scale photo hash clustering on a single machine. In *CVPR*, pages 19–27, 2015.

[Huang *et al.*, 2012] Hsin Chien Huang, Yung Yu Chuang, and Chu Song Chen. Multiple kernel fuzzy clustering. *IEEE Transactions on Fuzzy Systems*, 20(1):120–134, 2012.

[Jinglin *et al.*, 2016] Xu Jinglin, Han Junwei, and Nie Feiping. Discriminatively embedded k-means for multi-view clustering. In *CVPR*, pages 5356–5364, 2016.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25(2), 2012.

[Kumar *et al.*, 2011] Abhishek Kumar, Piyush Rai, and Hal Daume. Co-regularized multi-view spectral clustering. In *NIPS*, pages 1413–1421, 2011.

[Li *et al.*, 2015] Yeqing Li, Feiping Nie, Heng Huang, and Junzhou Huang. Large-scale multi-view spectral clustering via bipartite graph. pages 2750–2756, 2015.

[Liu *et al.*, 2016] Xinwang Liu, Yong Dou, Jianping Yin, Lei Wang, and En Zhu. Multiple kernel k-means clustering with matrix-induced regularization. In *AAAI*, pages 1888–1894, 2016.

[Lu *et al.*, 2014] Yanting Lu, Liantao Wang, Jianfeng Lu, Jingyu Yang, and Chunhua Shen. Multiple kernel clustering based on centered kernel alignment. *Pattern Recognition*, 47(11):3656–3664, 2014.

[Nie *et al.*, 2016] Feiping Nie, Jing Li, and Xuelong Li. Parameter-free auto-weighted multiple graph learning: A framework for multiview clustering and semi-supervised classification. In *IJCAI*, pages 1881–1887, 2016.

[Nie *et al.*, 2017] Feiping Nie, Guohao Cai, and Xuelong Li. Multi-view clustering and semi-supervised classification with adaptive neighbours. In *AAAI*, 2017.

[Shen *et al.*, 2017] Xiaobo Shen, Weiwei Liu, Ivor Tsang, Fuming Shen, and Quan-sen Sun. Compressed k-means for large-scale clustering. In *AAAI*, 2017.

[Simonyan and Zisserman, 2014] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[Wang *et al.*, 2016] De Wang, Feiping Nie, and Heng Huang. Fast robust non-negative matrix factorization for large-scale human action data clustering. In *IJCAI*, pages 2104–2110, 2016.

[Weiran *et al.*, 2015] Wang Weiran, Arora Raman, Livescu Karen, and Bilmes Jeff. On deep multi-view representation learning. In *ICML*, pages 1083–1092, 2015.

[Xia *et al.*, 2014] Rongkai Xia, Yan Pan, Lei Du, and Jian Yin. Robust multi-view spectral clustering via low-rank and sparse decomposition. In *AAAI*, pages 2149–2155, 2014.

[Yu *et al.*, 2012] Shi Yu, Lon Charles Tranchevent, Bart De Moor, and Yves Moreau. Optimized data fusion for kernel k-means clustering. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 34(5):1031–1039, 2012.

[Zhang and Lu, 2016] Ruqi Zhang and Zhiwu Lu. Large scale sparse clustering. In *IJCAI*, pages 2336–2342, 2016.

[Zhang *et al.*, 2015] Changqing Zhang, Huazhu Fu, Si Liu, Guangcan Liu, and Xiaochun Cao. Low-rank tensor constrained multiview subspace clustering. In *ICCV*, pages 1582–1590, 2015.

[Zhao *et al.*, 2009] Bin Zhao, James T. Kwok, and Changshui Zhang. Multiple kernel clustering. In *ICDM*, pages 638–649, 2009.

[Zhou and Burges, 2007] Dengyong Zhou and Christopher J. C. Burges. Spectral clustering and transductive learning with multiple views. In *ICML*, pages 1159–1166, 2007.

[Zhou *et al.*, 2015] Peng Zhou, Liang Du, Lei Shi, Hanmo Wang, and Yi-Dong Shen. Recovery of corrupted multiple kernels for clustering. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 4105–4111, 2015.