Multiple Kernel Clustering with Local Kernel Alignment Maximization

Miaomiao Li, Xinwang Liu
School of Computer
National University
of Defense Technology
Changsha, China, 410073

Lei Wang
School of Computer Science
and Software Engineering
University of Wollongong
NSW, Australia, 2522

Yong Dou, Jianping Yin, En Zhu
School of Computer
National University
of Defense Technology
Changsha, China, 410073

Abstract

Kernel alignment has recently been employed for multiple kernel clustering (MKC). However, we find that most of existing works implement this alignment in a global manner, which: i) *indiscriminately* forces all sample pairs to be equally aligned with the same ideal similarity; and ii) is inconsistent with a well-established concept that the similarity evaluated for two farther samples in a high dimensional space is less reliable. To address these issues, this paper proposes a novel MKC algorithm with a "local" kernel alignment, which only requires that the similarity of a sample to its k-nearest neighbours be aligned with the ideal similarity matrix. Such an alignment helps the clustering algorithm to focus on closer sample pairs that shall stay together and avoids involving unreliable similarity evaluation for farther sample pairs. We derive a new optimization problem to implement this idea, and design a two-step algorithm to efficiently solve it. As experimentally demonstrated on six challenging multiple kernel learning benchmark data sets, our algorithm significantly outperforms the state-of-the-art comparable methods in the recent literature, verifying the effectiveness and superiority of maximizing local kernel alignment.

Introduction

Multiple kernel clustering (MKC) aims to optimally integrate a group of pre-specified kernels to improve clustering performance [Zhao *et al.*, 2009; Lu *et al.*, 2014; Xia *et al.*, 2014; Zhou *et al.*, 2015; Kumar and Daumé, 2011]. Existing research in this regard can roughly be grouped into two categories. The first one learns a consensus matrix via low-rank optimization [Xia *et al.*, 2014; Zhou *et al.*, 2015; Kumar and Daumé, 2011]. The work in [Xia *et al.*, 2014] firstly constructs a transition probability matrix from each single view, and them uses them to recover a shared low-rank transition probability matrix as a crucial input to the standard Markov chain method for clustering. In [Zhou *et al.*, 2015], it is proposed to capture the structures of noises in each kernel and integrate them into a robust and consensus framework to learn a low-rank matrix. The algorithm [Kumar and Daumé, 2011] learns the clustering in one view and uses it to "label" the data in other views to modify a similarity matrix. By following multiple kernel learning (MKL) framework, the other category optimizes a group of kernel coefficients, and uses the combined kernel for clustering [Yu *et al.*, 2012; Gönen and Margolin, 2014; Du *et al.*, 2015; Lu *et al.*, 2014]. Along this line, the work in [Yu *et al.*, 2012] proposes a multiple kernel k-means clustering algorithm. In [Gönen and Margolin, 2014], the kernel combination weights are allowed to adaptively change with respect to samples to better capture their individual characteristics. By replacing the squared error in k-means with an $\ell_{2,1}$-norm based one, [Du *et al.*, 2015] presents a robust multiple kernel k-means algorithm that simultaneously finds the best clustering labels and the optimal combination of multiple kernels. In [Lu *et al.*, 2014], kernel alignment maximization is employed to jointly perform the k-means clustering and MKL. Our work in this paper belongs to the second category.

Among the above clustering algorithms in the second category, the kernel alignment maximization criterion for clustering [Lu *et al.*, 2014] has demonstrated promising performance. Actually, according to [Liu *et al.*, 2016], many classic MKC criteria such as [Yu *et al.*, 2012; Gönen and Margolin, 2014] can be interpreted from the perspective of kernel alignment maximization, suggesting the importance of this criterion for clustering. Nevertheless, although this alignment has the aforementioned good properties, we observe that it is implemented in a global manner, which: i) rigidly forces closer and farther sample pairs to be equally aligned to the same ideal similarity, and inappropriately neglects the intra-cluster variation of samples; and ii) is inconsistent with a well-established concept that the similarity evaluated for two farther samples in a high dimensional space is less reliable due to the presence of underlying manifold structure. As a result, maximizing global alignment could make these pre-specified kernels less effectively utilized, and in turn adversely affect the clustering performance.

To address these issues, we propose a novel MKC algorithm with a "local" kernel alignment. In specific, this local alignment only requires that the similarity of a sample to its k-nearest neighbours be aligned with the ideal similarity matrix. Such an alignment helps the clustering process to focus on closer sample pairs that shall stay together, avoids involv-

ing unreliable similarity evaluation for farther sample pairs, and also give the clustering process more flexibility to arrange the farther pairs according to the nature of data distribution. By this way, the local structure of the data can be well utilized to produce better alignment for clustering. After that, the optimization objective of the proposed local kernel alignment is carefully designed and an efficient algorithm with proved convergence is developed to solve the resultant optimization problem. Extensive experimental study has been conducted on six MKL benchmark data sets to evaluate clustering performance of the proposed algorithm. As indicated, our algorithm significantly outperforms the state-of-the-art ones, validating the effectiveness and advantage of the proposed local kernel alignment maximization.

Related Work

Kernel k-means clustering (KKM)

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 $\mathcal{H}$. The objective of kernel k-means clustering is to minimize the sum-of-squares loss over the cluster assignment matrix $\mathbf{Z} \in \{0,1\}^{n\times k}$, which can be formulated as the following optimization problem,

$$\min_{\mathbf{Z}\in\{0,1\}^{n\times k}} \sum_{i=1,c=1}^{n,k} Z_{ic}\|\phi(\mathbf{x}_i) - \boldsymbol{\mu}_c\|_2^2 \; s.t. \sum_{c=1}^{k} Z_{ic} = 1, \quad (1)$$

where $n_c = \sum_{i=1}^n Z_{ic}$ and $\boldsymbol{\mu}_c = \frac{1}{n_c}\sum_{i=1}^n Z_{ic}\phi(\mathbf{x}_i)$ are the number and centroid of the $c-\text{th}\,(1 \le c \le k)$ cluster.

The optimization problem in Eq.(1) can be equivalently rewritten as the following matrix-vector form,

$$\min_{\mathbf{Z}\in\{0,1\}^{n\times k}} \text{Tr}(\mathbf{K}) - \text{Tr}(\mathbf{L}^{\frac{1}{2}}\mathbf{Z}^\top\mathbf{K}\mathbf{Z}\mathbf{L}^{\frac{1}{2}}) \; s.t. \; \mathbf{Z}\mathbf{1}_k = \mathbf{1}_n, \quad (2)$$

where $\mathbf{K}$ is a kernel matrix with $K_{ij} = \phi(\mathbf{x}_i)^\top\phi(\mathbf{x}_j)$, $\mathbf{L} = \text{diag}([n_1^{-1}, n_2^{-1}, \cdots, n_k^{-1}])$ and $\mathbf{1}_\ell \in \mathbb{R}^\ell$ is a column vector with all elements 1.

The variables $\mathbf{Z}$ in Eq.(2) is discrete, which makes the optimization problem very difficult to solve. However, this problem is usually approximated through relaxing $\mathbf{Z}$ to take arbitrary real values. Specifically, by defining $\mathbf{H} = \mathbf{Z}\mathbf{L}^{\frac{1}{2}}$ and letting $\mathbf{H}$ take real values, we obtain a relaxed version of the above problem.

$$\min_{\mathbf{H}\in\mathbb{R}^{n\times k}} \text{Tr}\left(\mathbf{K}(\mathbf{I}_n - \mathbf{H}\mathbf{H}^\top)\right) \; s.t. \; \mathbf{H}^\top\mathbf{H} = \mathbf{I}_k, \quad (3)$$

where $\mathbf{I}_k$ is an identity matrix with size $k \times k$. Noting that $\mathbf{Z}^\top\mathbf{Z} = \mathbf{L}^{-1}$, it can be obtained that $\mathbf{L}^{\frac{1}{2}}\mathbf{Z}^\top\mathbf{Z}\mathbf{L}^{\frac{1}{2}} = \mathbf{I}_k$, and this leads to the orthogonality constraint on $\mathbf{H}$. Finally, one can obtain the optimal $\mathbf{H}$ for Eq.(3) by taking the k eigenvectors that correspond to the k largest eigenvalues of $\mathbf{K}$.

Multiple kernel k-means clustering (MKKM)

In a 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 $\phi_{\boldsymbol{\mu}}(\mathbf{x}) = [\mu_1\phi_1(\mathbf{x})^\top, \mu_2\phi_2(\mathbf{x})^\top, \cdots, \mu_m\phi_m(\mathbf{x})^\top]^\top$, where $\boldsymbol{\mu} = [\mu_1, \mu_2, \cdots, \mu_m]^\top$ denotes the coefficients of each base kernel that needs to be optimized during learning. Correspondingly, the kernel function over the above mapping function can be calculated as

$$\kappa_{\boldsymbol{\mu}}(\mathbf{x}_i, \mathbf{x}_j) = \phi_{\boldsymbol{\mu}}(\mathbf{x}_i)^\top\phi_{\boldsymbol{\mu}}(\mathbf{x}_j) = \sum_{p=1}^m \mu_p^2\kappa_p(\mathbf{x}_i, \mathbf{x}_j). \quad (4)$$

By replacing the kernel matrix $\mathbf{K}$ in Eq.(3) with $\mathbf{K}_{\boldsymbol{\mu}}$ computed via Eq.(4), the following optimization objective is obtained for MKKM,

$$\min_{\mathbf{H}\in\mathbb{R}^{n\times k}, \boldsymbol{\mu}\in\mathbb{R}_+^m} \text{Tr}\left(\mathbf{K}_{\boldsymbol{\mu}}(\mathbf{I}_n - \mathbf{H}\mathbf{H}^\top)\right) \; s.t. \; \mathbf{H}^\top\mathbf{H} = \mathbf{I}_k, \; \boldsymbol{\mu}^\top\mathbf{1}_m = 1. \quad (5)$$

This problem can be solved by alternately updating $\mathbf{H}$ and $\boldsymbol{\mu}$: i) **Optimizing $\mathbf{H}$ given $\boldsymbol{\mu}$.** With the kernel coefficients $\boldsymbol{\mu}$ fixed, the $\mathbf{H}$ can be obtained by solving a kernel k-means clustering optimization problem in Eq.(3); ii) **Optimizing $\boldsymbol{\mu}$ given $\mathbf{H}$**. With $\mathbf{H}$ fixed, $\boldsymbol{\mu}$ can be optimized via solving the following quadratic programming with linear constraints,

$$\min_{\boldsymbol{\mu}\in\mathbb{R}_+^m} \sum_{p=1}^m \mu_p^2\text{Tr}\left(\mathbf{K}_p(\mathbf{I}_n - \mathbf{H}\mathbf{H}^\top)\right) \; s.t. \; \boldsymbol{\mu}^\top\mathbf{1}_m = 1. \quad (6)$$

As noted in [Yu *et al.*, 2012; Gönen and Margolin, 2014], using a convex combination of kernels $\sum_{p=1}^m \mu_p\mathbf{K}_p$ to replace $\mathbf{K}_{\boldsymbol{\mu}}$ in Eq.(5) is not a viable option, because this could make only one single kernel be activated and all the others assigned with zero weights. Also, other recent work using ℓ_2-norm combination can be found in [Kloft *et al.*, 2011; 2009; Cortes *et al.*, 2009].

Proposed Formulation

The Connection Between MKKM and Unsupervised Kernel Alignment Maximization

As a well-established criterion, kernel alignment maximization has been widely used to tune kernel parameters in supervised learning [Cortes *et al.*, 2012]. Nevertheless, this criterion is not readily applicable to clustering since the true labels in unsupervised learning is absent. A promising remedy is to update kernel coefficients by maximizing the alignment between the combined kernel $\mathbf{K}_{\boldsymbol{\mu}}$ and $\mathbf{H}\mathbf{H}^\top$, where $\mathbf{H}$ can be treated as pseudo-labels in the last iteration [Liu *et al.*, 2016]. In specific, the kernel alignment maximization for clustering can be fulfilled as,

$$\max_{\mathbf{H}\in\mathbb{R}^{n\times k}, \boldsymbol{\mu}\in\mathbb{R}_+^m} \frac{\langle\mathbf{K}_{\boldsymbol{\mu}}, \mathbf{H}\mathbf{H}^\top\rangle_{\text{F}}}{\sqrt{\langle\mathbf{K}_{\boldsymbol{\mu}}, \mathbf{K}_{\boldsymbol{\mu}}\rangle_{\text{F}}}} \; s.t. \; \mathbf{H}^\top\mathbf{H} = \mathbf{I}_k, \; \boldsymbol{\mu}^\top\mathbf{1}_m = 1, \quad (7)$$

where $\langle\mathbf{K}_{\boldsymbol{\mu}}, \mathbf{H}\mathbf{H}^\top\rangle_{\text{F}} = \text{Tr}(\mathbf{K}_{\boldsymbol{\mu}}\mathbf{H}\mathbf{H}^\top)$, $\langle\mathbf{K}_{\boldsymbol{\mu}}, \mathbf{K}_{\boldsymbol{\mu}}\rangle_{\text{F}} = \hat{\boldsymbol{\mu}}^\top\mathbf{M}\hat{\boldsymbol{\mu}}$ with $\hat{\boldsymbol{\mu}} = [\mu_1^2, \cdots, \mu_m^2]^\top$ and $\mathbf{M}$ is a positive semi-definite matrix with $M_{pq} = \text{Tr}(\mathbf{K}_p^\top\mathbf{K}_q)$ [Cortes *et al.*, 2012].

Although Eq.(7) is not difficult to understood, directly optimizing it is difficult since it is a fourth-order fractional optimization problem. In the following, we derive a new and related optimization problem from this kernel alignment maximization problem, based on the following two results.

Theorem 0.1 provides a second-order upper bound for the denominator in Eq.(7).

Theorem 0.1 *$\boldsymbol{\mu}^\top\mathbf{M}\boldsymbol{\mu}$ is an upper bound of $\hat{\boldsymbol{\mu}}^\top\mathbf{M}\hat{\boldsymbol{\mu}}$.*

Proof 1 *For any positive semi-definite matrices* $\mathbf{K}_p$ *and* $\mathbf{K}_q$*, there exists matrices* $\mathbf{U}_p$ *and* $\mathbf{U}_q$ *such that* $\mathbf{K}_p = \mathbf{U}_p\mathbf{U}_p^\top$ *and* $\mathbf{K}_q = \mathbf{U}_q\mathbf{U}_q^\top$*. Consequently,* $M_{pq} = \mathrm{Tr}(\mathbf{K}_p^\top\mathbf{K}_q) = \mathrm{Tr}(\mathbf{U}_p\mathbf{U}_p^\top\mathbf{U}_q\mathbf{U}_q^\top) = \mathrm{Tr}((\mathbf{U}_p^\top\mathbf{U}_q)(\mathbf{U}_p^\top\mathbf{U}_q)^\top) = \|\mathbf{U}_p^\top\mathbf{U}_q\|_{\mathrm{F}}^2 \geq 0$*, where* $\|\cdot\|_{\mathrm{F}}$ *denotes Frobenius norm. Also, we have* $\boldsymbol{\mu}^\top\mathbf{M}\boldsymbol{\mu} = \sum_{p,q=1}^m M_{pq}\mu_p\mu_q \geq \sum_{p,q=1}^m M_{pq}\mu_p^2\mu_q^2 = \hat{\boldsymbol{\mu}}^\top\mathbf{M}\hat{\boldsymbol{\mu}}$ *since* $\boldsymbol{\mu} \in \mathbb{R}_+^m$, $\boldsymbol{\mu}^\top\mathbf{1}_m = 1$ *and* $M_{pq} \geq 0$*. This completes the proof.*

Compared with $\hat{\boldsymbol{\mu}}^\top\mathbf{M}\hat{\boldsymbol{\mu}}$, $\boldsymbol{\mu}^\top\mathbf{M}\boldsymbol{\mu}$ is much easier to handle since it leads to a well studied quadratic programming. Moreover, this term can be treated as a regularization on the kernel coefficients to prevent μ_p and μ_q from being jointly assigned to a large weight if M_{pq} is relatively high.

In addition, we find that minimizing the negative of numerator, i.e., $-\mathrm{Tr}(\mathbf{K}_{\boldsymbol{\mu}}\mathbf{H}\mathbf{H}^\top)$, together with $\boldsymbol{\mu}^\top\mathbf{M}\boldsymbol{\mu}$ simultaneously cannot guarantee that the whole objective is convex w.r.t $\boldsymbol{\mu}$ with fixed $\mathbf{H}$. This would affect the quality of solution at each iteration, leading to unsatisfying performance. Fortunately, the following Theorem gives a good substitute of $-\mathrm{Tr}(\mathbf{K}_{\boldsymbol{\mu}}\mathbf{H}\mathbf{H}^\top)$ while with convexity.

Theorem 0.2 $\mathrm{Tr}\left(\mathbf{K}_{\boldsymbol{\mu}}(\mathbf{I}_n - \mathbf{H}\mathbf{H}^\top)\right)$ *is convex w.r.t* $\boldsymbol{\mu}$ *with fixed* $\mathbf{H}$.

Proof 2 *We have* $\mathbf{H}\mathbf{H}^\top\mathbf{H} = \mathbf{H}$ *since* $\mathbf{H}^\top\mathbf{H} = \mathbf{I}_k$*. By denoting* $\mathbf{H} = [\mathbf{h}_1, \cdots, \mathbf{h}_k]$*, we can see that* $\mathbf{H}\mathbf{H}^\top\mathbf{h}_c = \mathbf{h}_c$*,* $\forall\, 1 \leq c \leq k$*. This means* $\mathbf{H}\mathbf{H}^\top$ *has* k *eigenvalue with* 1*. Meanwhile, its rank is no more than* k*, which implies its has* $n-k$ *eigenvalue with* 0*. Correspondingly,* $\mathbf{I}_n - \mathbf{H}\mathbf{H}^\top$ *has* $n-k$ *and* k *eigenvalue with* 1 *and* 0*. As a result,* $\mathrm{Tr}\left(\mathbf{K}_p(\mathbf{I}_n - \mathbf{H}\mathbf{H}^\top)\right) \geq 0$*.* $\mathrm{Tr}\left(\mathbf{K}_{\boldsymbol{\mu}}(\mathbf{I}_n - \mathbf{H}\mathbf{H}^\top)\right) = \sum_{p=1}^m \mu_p^2\mathrm{Tr}\left(\mathbf{K}_p(\mathbf{I}_n - \mathbf{H}\mathbf{H}^\top)\right)$*, which is therefore convex w.r.t* $\boldsymbol{\mu}$*. This completes the proof.*

Actually, $\mathrm{Tr}\left(\mathbf{K}_{\boldsymbol{\mu}}(\mathbf{I}_n - \mathbf{H}\mathbf{H}^\top)\right)$ can be intuitively understood by adding a prior $\sum_{p=1}^m \mu_p^2\mathrm{Tr}\left(\mathbf{K}_p\right)$ on $-\mathrm{Tr}(\mathbf{K}_{\boldsymbol{\mu}}\mathbf{H}\mathbf{H}^\top)$ to guarantee its convexity with $\boldsymbol{\mu}$.

Based on the above-mentioned observations, instead of maximizing the kernel alignment by solving a fractional optimization in Eq.(7), we turn to minimize the following,

$$\begin{aligned}\min_{\mathbf{H}\in\mathbb{R}^{n\times k},\boldsymbol{\mu}\in\mathbb{R}_+^m}\ & \mathrm{Tr}\left(\mathbf{K}_{\boldsymbol{\mu}}(\mathbf{I}_n - \mathbf{H}\mathbf{H}^\top)\right) + \frac{\lambda}{2}\boldsymbol{\mu}^\top\mathbf{M}\boldsymbol{\mu} \\ s.t.\ & \mathbf{H}^\top\mathbf{H} = \mathbf{I}_k,\ \boldsymbol{\mu}^\top\mathbf{1}_m = 1,\end{aligned} \tag{8}$$

where λ is introduced to trade off the two terms.

The above results will be used in the next section to develop our optimization problem for the proposed local kernel alignment.

Multiple Kernel Clustering with Local Kernel Alignment Maximization

As seen, Eq.(7) (or Eq.(8)) maximizes the alignment between the combined kernel matrices $\mathbf{K}_{\boldsymbol{\mu}}$ and the ideal kernel matrix $\mathbf{H}\mathbf{H}^\top$ globally. By following the analysis in introduction, such a global criterion: i) *indiscriminately* forces closer and farther sample pairs to be equally aligned to the same ideal similarity, and inappropriately neglects the intra-cluster variation of samples; and ii) is inconsistent with a well-established concept that the similarity evaluated for two farther samples in a high dimensional space is less reliable. These two drawbacks could adversely impact the clustering performance. In this paper, instead of enforcing the global alignment of all samples, we propose to locally align the similarity of each sample to its k-nearest neighbours with corresponding ideal kernel matrix, which is flexible and able to well handle the intra-cluster variations. In specific, the local kernel alignment for the i-th sample can be calculated as,

$$\max_{\mathbf{H}\in\mathbb{R}^{n\times k},\boldsymbol{\mu}\in\mathbb{R}_+^m} \frac{\langle\mathbf{K}_{\boldsymbol{\mu}}^{(i)}, \mathbf{H}^{(i)}\mathbf{H}^{(i)\top}\rangle_{\mathrm{F}}}{\sqrt{\langle\mathbf{K}_{\boldsymbol{\mu}}^{(i)}, \mathbf{K}_{\boldsymbol{\mu}}^{(i)}\rangle_{\mathrm{F}}}}\ s.t.\ \mathbf{H}^\top\mathbf{H} = \mathbf{I}_k,\ \boldsymbol{\mu}^\top\mathbf{1}_m = 1, \tag{9}$$

where $\mathbf{K}_{\boldsymbol{\mu}}^{(i)}$ and $\mathbf{H}^{(i)}$ are sub-matrix of $\mathbf{K}_{\boldsymbol{\mu}}$ and $\mathbf{H}$ whose indices are specified by the τ-nearest neighbors of the i-th sample, and $\mathbf{M}^{(i)}$ is a matrix with $M_{pq}^{(i)} = \mathrm{Tr}(\mathbf{K}_p^{(i)\top}\mathbf{K}_q^{(i)})$.

By following the aforementioned analysis in subsection , Eq.(9) can be conceptually rewritten as,

$$\begin{aligned}\min_{\mathbf{H}\in\mathbb{R}^{n\times k},\boldsymbol{\mu}\in\mathbb{R}_+^m}\ & \mathrm{Tr}\left(\mathbf{K}_{\boldsymbol{\mu}}^{(i)}(\mathbf{I}_\tau - \mathbf{H}^{(i)}\mathbf{H}^{(i)\top})\right) + \frac{\lambda}{2}\boldsymbol{\mu}^\top\mathbf{M}^{(i)}\boldsymbol{\mu} \\ & s.t.\ \mathbf{H}^\top\mathbf{H} = \mathbf{I}_k,\ \boldsymbol{\mu}^\top\mathbf{1}_m = 1\end{aligned} \tag{10}$$

where $\mathbf{K}_{\boldsymbol{\mu}}^{(i)} = \mathbf{S}^{(i)\top}\mathbf{K}_{\boldsymbol{\mu}}\mathbf{S}^{(i)}$, $\mathbf{H}^{(i)} = \mathbf{S}^{(i)\top}\mathbf{H}$, $\mathbf{S}^{(i)} \in \{0,1\}^{n\times\tau}$ is a matrix indicating the τ-nearest neighbors of the i-th sample and $\mathbf{I}_\tau$ is an identity matrix with size τ.

By taking over the local kernel alignment in Eq.(10) for each sample and defining $\mathbf{A}^{(i)} = \mathbf{S}^{(i)}\mathbf{S}^{(i)\top}$, we obtain the objective function of our proposed algorithm as follows,

$$\begin{aligned}\min_{\mathbf{H}\in\mathbb{R}^{n\times k},\boldsymbol{\mu}\in\mathbb{R}_+^m}\ & \sum_{i=1}^n\Big[\mathrm{Tr}\left(\mathbf{K}_{\boldsymbol{\mu}}\left(\mathbf{A}^{(i)} - \mathbf{A}^{(i)}\mathbf{H}\mathbf{H}^\top\mathbf{A}^{(i)}\right)\right) \\ & + \frac{\lambda}{2}\boldsymbol{\mu}^\top\mathbf{M}^{(i)}\boldsymbol{\mu}\Big]\ s.t.\ \mathbf{H}^\top\mathbf{H} = \mathbf{I}_k,\ \boldsymbol{\mu}^\top\mathbf{1}_m = 1\end{aligned} \tag{11}$$

Algorithm 1 Multiple Kernel Clustering with Local Kernel Alignment Maximization

Input: $\{\mathbf{K}_p\}_{p=1}^m$, k, λ and ϵ_0.
Output: $\mathbf{H}$ and $\boldsymbol{\mu}$.
Initialize $\boldsymbol{\mu}^{(1)} = \mathbf{1}_m/m$ and $t = 1$.
Generating $\mathbf{S}^{(i)}$ for i-th samples $(1 \leq i \leq n)$ by $\mathbf{K}_{\boldsymbol{\mu}^{(1)}}$.
repeat
  $\mathbf{K}_{\boldsymbol{\mu}}^{(t)} = \sum_{p=1}^m \left(\mu_p^{(t)}\right)^2\mathbf{K}_p$.
  Update $\mathbf{H}^{(t)}$ by solving Eq.(12) with given $\mathbf{K}_{\boldsymbol{\mu}}^{(t)}$.
  Update $\boldsymbol{\mu}^{(t)}$ by solving Eq.(13) with given $\mathbf{H}^{(t)}$.
  $t = t + 1$.
until $\left(\mathrm{obj}^{(t-1)} - \mathrm{obj}^{(t)}\right)/\mathrm{obj}^{(t)} \leq \epsilon_0$

Alternate optimization

Although the kernel alignment is maximized in a localized way, the resultant optimization problem in Eq.(11) is not altered significantly and still easy to solve by existing off-the-shelf packages. In specific, we design a two-step algorithm to

solve this problem alternately. (i) **Optimizing $\mathbf{H}$ with fixed $\boldsymbol{\mu}$.** Given $\boldsymbol{\mu}$, $\mathbf{H}$ can be obtained by solving the following optimization problem,

$$\max_{\mathbf{H}\in\mathbb{R}^{n\times k}} \operatorname{Tr}\left(\mathbf{H}^\top \sum\nolimits_{i=1}^{n}\left(\mathbf{A}^{(i)}\mathbf{K}_{\boldsymbol{\mu}}\mathbf{A}^{(i)}\right)\mathbf{H}\right) \ s.t.\ \mathbf{H}^\top\mathbf{H}=\mathbf{I}_k, \tag{12}$$

which is a standard kernel k-means clustering problem and can be efficiently solved; (ii) **Optimizing $\boldsymbol{\mu}$ with fixed $\mathbf{H}$.** Given $\mathbf{H}$, the optimization in Eq.(11) w.r.t $\boldsymbol{\mu}$ is a quadratic programming with linear constraints, which essentially solves the following problem,

$$\min_{\boldsymbol{\mu}\in\mathbb{R}_+^m} \ \frac{1}{2}\boldsymbol{\mu}^\top\left(2\mathbf{Z}+\lambda\mathbf{M}\right)\boldsymbol{\mu} \quad s.t.\ \boldsymbol{\mu}^\top\mathbf{1}_m=1, \tag{13}$$

where $\mathbf{Z}=\operatorname{diag}\left(\left[\operatorname{Tr}\left(\mathbf{K}_1\mathbf{V}\right),\cdots,\operatorname{Tr}\left(\mathbf{K}_m\mathbf{V}\right)\right]\right)$, $\mathbf{V}=\sum\limits_{i=1}^{n}\Big(\mathbf{A}^{(i)}-\mathbf{A}^{(i)}\mathbf{H}\mathbf{H}^\top\mathbf{A}^{(i)}\Big)$ and $M_{pq}=\sum\limits_{i=1}^{n}\operatorname{Tr}\Big(\mathbf{K}_p\mathbf{A}^{(i)}\mathbf{K}_q\mathbf{A}^{(i)}\Big)$.

In sum, our algorithm for solving Eq.(11) is outlined in Algorithm 1, where $\text{obj}^{(t)}$ denotes the objective value at the t-th iterations. It is worth pointing out that the neighborhood of each sample is kept unchanged during the optimization. In specific, the τ-nearest neighbors of each samples are measured by $\mathbf{K}_{\boldsymbol{\mu}^{(1)}}$. By doing so, the objective of Algorithm 1 is guaranteed to be monotonically decreased when optimizing one variable with the other fixed at each iteration. At the same time, the whole optimization problem is lower-bounded. As a result, the proposed algorithm can be guaranteed to be convergent. We also record the objective at each iteration and the results validate the convergence. The algorithm usually converges in less than ten iterations in all of our experiments.

Experiments

Data sets

The proposed algorithm is experimentally evaluated on six widely used MKL benchmark data sets shown in Table 1. They are UCI-Digital[1], Oxford Flower17[2], Protein fold prediction[3], YALE[4], Oxford Flower102[5] and Caltech102[6].

Table 1: Datasets used in our experiments.

Dataset	#Samples	#Views	#Classes
Digital	2000	3	10
Flower17	1360	7	17
ProteinFold	694	12	27
YALE	165	5	15
Flower102	8189	4	102
Caltech102	1530	25	102

For ProteinFold, we generate 12 base kernel matrices by following [Damoulas and Girolami, 2008], where the second order polynomial kernel and inner product (cosine) kernel are applied to the first ten feature sets and the last two feature sets, respectively. For YALE, five base kernel matrices are constructed according to [Zhou *et al.*, 2015]. For the rest of the other data sets, all kernel matrices are pre-computed and publicly downloaded from the above websites.

[1] http://ss.sysu.edu.cn/~py/

[2] http://www.robots.ox.ac.uk/~vgg/data/flowers/17/

[3] http://mkl.ucsd.edu/dataset/protein-fold-prediction

[4] http://vismod.media.mit.edu/vismod/classes/mas622-00/datasets/

[5] http://www.robots.ox.ac.uk/~vgg/data/flowers/102/

[6] http://mkl.ucsd.edu/dataset/ucsd-mit-caltech-101-mkl-dataset

Compared algorithms

Many recently proposed method are compared, including

- **Average multiple kernel k-means (A-MKKM)**: All kernels are uniformly weighted to generate a new kernel, which is taken as the input of kernel k-means.
- **Single best kernel k-means (SB-KKM)**: Kernel k-means is performed on each single kernel and the best result is reported.
- **Multiple kernel k-means (MKKM)** [Huang *et al.*, 2012]: The algorithm alternately performs kernel k-means and updates kernel coefficients.
- **Localized multiple kernel k-means (LMKKM)** [Gönen and Margolin, 2014]: LMMKM combines the kernels by sample-adaptive weights.
- **Robust multiple kernel k-means (RMKKM)** [Du *et al.*, 2015]: RMKKM improves the robustness of MKKM by replacing the sum-of-squared loss with an $\ell_{2,1}$-norm one.
- **Co-regularized spectral clustering (CRSC)** [Kumar and Daumé, 2011]: CRSC provides a co-regularization way to perform spectral clustering.
- **Robust multiview spectral clustering (RMSC)** [Xia *et al.*, 2014]: RMSC constructs a transition probability matrix from each single view, and uses them to recover a shared low-rank transition matrix for clustering.
- **Robust Multiple Kernel Clustering (RMKC)** [Zhou *et al.*, 2015]: RMKC learns a robust yet low-rank kernel for clustering by capturing the structure of noises in multiple kernels.

The Matlab codes of KKM, MKKM and LMKKM are publicly available at `https://github.com/mehmetgonen/lmkkmeans`. For RMKKM, CRSC, RMSC and RCE, we use their matlab implementations from authors' websites in our experiments.

Experimental settings

In all our experiments, all base kernels are first centered and then scaled so that for all i and p we have $K_p(\mathbf{x}_i,\mathbf{x}_i)=1$ by following [Cortes *et al.*, 2012; 2013]. For all data sets, it is assumed that the true number of clusters is known and set as the true number of classes. The parameters of RMKKM, RMSC and RMKC are selected by grid search according to the suggestions in their papers. For the proposed algorithm, its regularization parameters λ and τ are chosen from $[2^{-15},2^{-13},\cdots,2^{15}]$ and $[0.05,0.1,\cdots,0.95]*n$ by grid search, where n is the number of samples.

The widely used clustering accuracy (ACC), normalized mutual information (NMI) and purity are applied to evaluate the clustering performance. For all algorithms, we repeat

Table 2: ACC, NMI and purity comparison of different clustering algorithms on six benchmark data sets.

Datasets	A-MKKM	SB-KKM	MKKM	LMKKM	RMKKM	CRSC	RMSC	RMKC	Proposed
ACC									
Digital	88.75	75.40	47.00	47.00	40.45	84.80	90.40	88.90	**96.25**
Flower17	51.03	42.06	45.37	42.94	48.38	52.72	53.90	52.35	**63.75**
ProteinFold	28.10	33.86	27.23	23.49	30.98	34.87	33.00	28.82	**37.90**
YALE	52.12	56.97	52.12	53.33	58.79	55.15	56.36	56.97	**64.24**
Flower102	27.29	33.13	21.96	22.57	28.17	37.26	32.97	33.54	**40.84**
Caltech102	35.56	33.14	34.77	27.97	29.67	33.33	31.50	35.56	**39.48**
NMI									
Digital	80.59	68.38	48.16	48.16	46.87	73.51	81.80	80.88	**91.63**
Flower17	50.19	45.14	45.35	44.12	50.73	52.13	53.89	50.42	**59.61**
ProteinFold	38.53	42.03	37.16	34.92	38.78	43.34	43.91	39.46	**44.46**
YALE	57.72	58.42	54.16	55.59	59.70	56.89	59.11	57.69	**65.10**
Flower102	46.32	48.99	42.30	43.24	48.17	54.18	53.36	49.73	**57.60**
Caltech102	59.90	59.07	59.64	55.17	55.86	58.20	58.40	59.90	**62.67**
purity									
Digital	88.75	76.10	49.70	49.70	44.20	77.75	82.90	88.90	**96.25**
Flower17	51.99	44.63	46.84	45.81	51.54	56.47	53.24	53.01	**63.82**
ProteinFold	36.17	41.21	33.86	32.71	36.60	40.78	42.36	36.46	**43.95**
YALE	53.94	57.58	52.73	54.55	59.39	56.36	56.97	57.58	**64.85**
Flower102	32.28	38.78	27.61	28.79	33.86	44.08	40.24	38.87	**48.21**
Caltech102	37.12	35.10	37.25	29.41	31.70	35.75	33.27	37.12	**41.83**

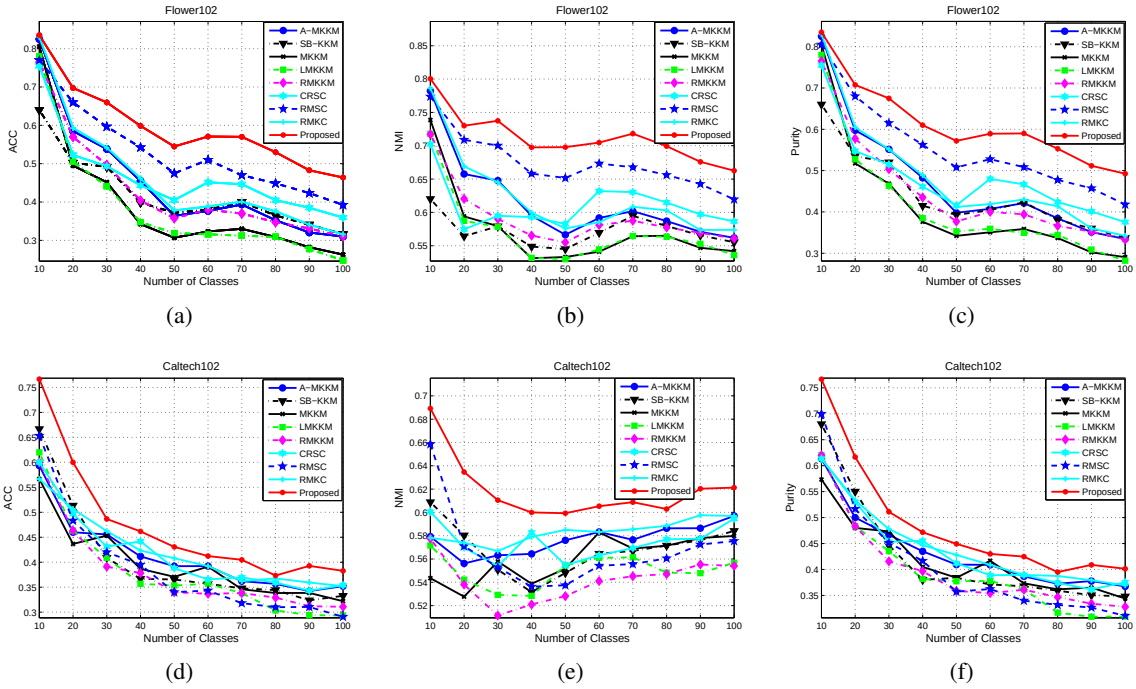


Figure 1: Clustering accuracy, NMI and purity comparison with different number of classes on Flower102 and Caltech102.

each experiment for 50 times with random initialization to reduce the affect of randomness caused by k-means, and report the best result.

Experimental results

The ACC, NMI and purity of the above-mentioned algorithms on the six data sets are reported in Table 2. We have the following observations from these results:

- Our algorithm demonstrates the best clustering performance in terms of clustering accuracy, NMI and purity on all data sets. Taking the results in Table 2 as an example, it exceeds the second best one by 5.85%, 9.85%, 3.03%, 5.45%, 3.58% and 3.92% on Digital, Flower17, ProteinFold, YALE, Flower102 and

Table 3: Aggregated ACC, NMI and purity comparison of different clustering algorithms on Flower102 and Caltech102.

Datasets	A-MKKM	SB-KKM	MKKM	LMKKM	RMKKM	CRSC	RMSC	RMKC	Proposed
Flower102	45.21	42.08	39.08	38.56	43.33	46.67	52.89	46.25	**59.54**
	61.62	57.22	57.35	57.06	59.24	61.08	67.51	62.20	**71.24**
	47.54	44.28	41.48	41.53	45.04	48.42	55.59	48.53	**61.36**
Caltech102	41.18	40.26	39.50	37.88	37.99	41.49	38.64	42.00	**47.11**
	57.68	56.83	56.01	55.01	54.19	57.43	56.74	58.35	**61.92**
	43.38	42.37	41.77	39.72	39.98	43.53	41.13	44.34	**48.75**

Caltech102, respectively. Also, its superiority is also confirmed from the NMI and purity reported in Table 2.

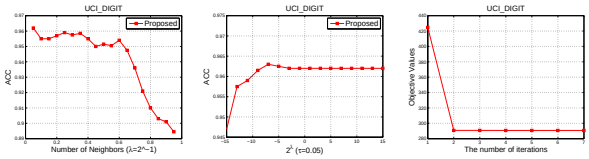
- The proposed algorithm significantly outperforms exiting MKKM, which can be seen as a special case of global kernel alignment. In specific, the clustering accuracy of MKKM is only 47% on Digital, which implies that it may even not work on this data set. In contrast, our algorithm achieves 96.25%, which is the best result among all the compared ones.
- As a strong baseline, A-MKKM usually demonstrates comparable or even better performance than most of algorithms in comparison. However, our algorithm clearly outperforms this baseline on all data sets, which indicates its superiority in clustering performance.

Table 2 also reports the comparison of NMI and purity. Again, we observe that the proposed algorithm significantly outperforms the compared ones. In all, these results have well verified the effectiveness of maximizing the kernel alignment in a local way.

To investigate the clustering performance with respect to the number of classes, we select samples from the first $10, 20, \cdots, 100$ classes on both Flower102 and Caltech102, where 20 and 15 samples are randomly selected for each class. By this way, we generate ten data sets on Flower102 and Caltech102, respectively.

The ACC, NMI and purity of the above-mentioned algorithms with different number of classes on Flower102 are plotted in sub-figures 1(a), 1(b) and 1(c). As shown, our algorithm (in red) consistently keeps on the top of all sub-figures, indicating the best performance. Meanwhile, we also find similar results from the results on Caltech102 in sub-figures 1(d), 1(e) and 1(f). Moreover, we report the aggregated ACC, NMI and purity of each algorithm, which is defined as the mean of ACC (NMI, purity) on Flower102 or Caltech102 with the number of classes varied in the range of $10, 20, \cdots, 100$, as shown in Table 3. Again, our algorithm is superior to other compared algorithms in terms of the aggregated ACC, NMI and purity.

From the above experiments, we can conclude that the proposed algorithm: i) effectively addresses the issues of indiscriminately forcing all sample pairs to be equally aligned to the same ideal similarity; and ii) well utilizes the local structure of data to significantly improve clustering performance. Our local kernel alignment is flexible and allows the pre-specified kernels to be aligned for better clustering, bringing the significant improvements on clustering performance.



(a) (b) (c)

Figure 2: (a) The effect of the number of neighbors τ, (b) The regularization parameter λ on clustering accuracy, (c) The objective value of our algorithm at each iteration.

Parameter selection and Convergence

As can be seen in Eq.(11), our algorithm introduces the number of neighbors τ and regularization parameter λ. We then experimentally show the effect of each parameter on the performance of our algorithm by fixing the other on Digital.

Figure 2(a) plots the ACC of our algorithm by varying τ in a large range $[0.05, 0.1, \cdots, 0.95] * n$ with $\lambda = 2^{-1}$. From this figure, we observe: i) with the increase of τ, the ACC first maintains on a high value and then decreases, validating the effectiveness of maximizing the local kernel alignment; and ii) our algorithm shows stable performance across a wide range of τ. Similarly, Figure 2(b) presents the ACC of our algorithm by varying λ from 2^{-15} to 2^{15} with $\tau = 0.05$. Again, our algorithm demonstrates stable performance across a wide range of λ. These results indicate that the performance of our method is stable across a wide range of parameters.

An example of the objective value of our algorithm at each iteration is plotted in Figure 2(c). As observed from this example, the objective value is monotonically decreased and the algorithm quickly converges in less than ten iterations.

Conclusions

This work proposes the multiple kernel clustering algorithm by maximizing the kernel alignment locally—a more flexible and effective algorithm which well handles the issue of indiscriminately forcing all sample pairs to be equally aligned to the same ideal similarity and integrates the underlying local structure of data to achieve better alignment for clustering. A two-step algorithm with proved convergence is designed to solve the resultant optimization problem. Experimental results clearly demonstrates the superiority of our algorithm. In the future, we plan to extend our algorithm to a more general framework, and use it as a platform to revisit existing multiple kernel clustering algorithms and uncover their relationship. Meanwhile, designing an sample-specific penalty for each sample is also worth exploring.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (project No. 61403405 and U1435219).

References

[Cortes *et al.*, 2009] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. L2 regularization for learning kernels. In *UAI*, pages 109–116, 2009.

[Cortes *et al.*, 2012] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *JMLR*, 13:795–828, 2012.

[Cortes *et al.*, 2013] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Multi-class classification with maximum margin multiple kernel. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 46–54, 2013.

[Damoulas and Girolami, 2008] Theodoros Damoulas and Mark A. Girolami. Probabilistic multi-class multi-kernel learning: on protein fold recognition and remote homology detection. *Bioinformatics*, 24(10):1264–1270, 2008.

[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 clustering using ℓ_{21}-norm. In *IJCAI*, 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. In *NIPS*, pages 1305–1313, 2014.

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

[Kloft *et al.*, 2009] Marius Kloft, Ulf Brefeld, Sören Sonnenburg, Pavel Laskov, Klaus-Robert Müller, and Alexander Zien. Efficient and accurate l_p-norm multiple kernel learning. In *NIPS*, pages 997–1005, 2009.

[Kloft *et al.*, 2011] Marius Kloft, Ulf Brefeld, Sören Sonnenburg, and Alexander Zien. l_p-norm multiple kernel learning. *JMLR*, 12:953–997, 2011.

[Kumar and Daumé, 2011] Abhishek Kumar and Hal Daumé. A co-training approach for multi-view spectral clustering. In *ICML*, pages 393–400, 2011.

[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*, 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.

[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, Léon-Charles Tranchevent, Xinhai Liu, Wolfgang Glänzel, Johan A. K. Suykens, Bart De Moor, and Yves Moreau. Optimized data fusion for kernel k-means clustering. *IEEE TPAMI*, 34(5):1031–1039, 2012.

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

[Zhou *et al.*, 2015] Peng Zhou, Liang Du, Lei Shi, Hanmo Wang, and Yi-Dong Shen. Recovery of corrupted multiple kernels for clustering. In *IJCAI*, pages 4105–4111, 2015.