The Global Kernel k-Means Algorithm for Clustering in Feature Space

Grigorios F. Tzortzis and Aristidis C. Likas, *Senior Member, IEEE*

***Abstract*—Kernel k-means is an extension of the standard k-means clustering algorithm that identifies nonlinearly separable clusters. In order to overcome the cluster initialization problem associated with this method, we propose the global kernel k-means algorithm, a deterministic and incremental approach to kernel-based clustering. Our method adds one cluster at each stage, through a global search procedure consisting of several executions of kernel k-means from suitable initializations. This algorithm does not depend on cluster initialization, identifies nonlinearly separable clusters, and, due to its incremental nature and search procedure, locates near-optimal solutions avoiding poor local minima. Furthermore, two modifications are developed to reduce the computational cost that do not significantly affect the solution quality. The proposed methods are extended to handle weighted data points, which enables their application to graph partitioning. We experiment with several data sets and the proposed approach compares favorably to kernel k-means with random restarts.**

***Index Terms*—Clustering, graph partitioning, k-means, kernel k-means.**

I. Introduction

CLUSTERING, the goal of which is to partition data points into homogeneous groups, arises in a number of fields such as pattern recognition, machine learning, data mining, and image processing. One of the most popular clustering algorithms is k-means [1], where groups are identified by minimizing the clustering error defined as the sum of the squared Euclidean distances between each data set point and the corresponding cluster center. This algorithm suffers from two serious limitations. First, the solution depends heavily on the initial positions of the cluster centers, resulting in poor minima, and second it can only find linearly separable clusters.

A simple but very popular workaround for the first limitation is the use of multiple restarts, where the centers of the clusters are randomly placed at different initial positions, hence better local minima can be found. Still we have to decide on the number of restarts and also we are never sure if the initializations tried are sufficient so as to obtain a near-optimal minimum. To deal with this problem, the global k-means algorithm has been proposed [2], which employs the k-means algorithm as a local search procedure. This algorithm incrementally solves the M-clustering problem by solving all *intermediate problems* with $1, \ldots, M$ clusters using k-means. The solution with M clusters is built deterministically, so there is no dependency on initial conditions, and near-optimal solutions are found as shown in [2].

Kernel k-means [3] is an extension of the standard k-means algorithm that maps data points from the input space to a feature space through a nonlinear transformation and minimizes the clustering error in feature space. Thus, nonlinearly separated clusters in input space are obtained, overcoming the second limitation of k-means.

In this paper, we propose the global kernel k-means algorithm, a deterministic algorithm for optimizing the clustering error in feature space that employs kernel k-means as a local search procedure in order to solve the M-clustering problem. The algorithm works in an incremental fashion by solving all intermediate problems with $1, \ldots, M$ clusters, using kernel k-means. The idea behind the proposed method is that a near-optimal solution with k clusters can be obtained by starting with a near-optimal solution with $k-1$ clusters and initializing the kth cluster appropriately based on a local search. During the local search, N initializations are tried, where N is the size of the data set. The $k-1$ clusters are always initialized to the $(k-1)$-clustering problem solution, while the kth cluster for each initialization includes a single data set point. The solution with the lowest clustering error is kept as the solution with k clusters. Since the optimal solution for the 1-clustering problem is known, the above procedure can be applied iteratively to find a near-optimal solution to the M-clustering problem. This algorithm combines the advantages of both global k-means and kernel k-means, therefore it avoids both limitations of k-means. A drawback of global kernel k-means is its high computational complexity, inherited from the other two algorithms, as it requires running kernel k-means MN times when solving the M-clustering problem.

To lower the complexity, two speeding up schemes are proposed called fast global kernel k-means and global kernel k-means with convex mixture models. The first variant, for each intermediate problem locates the data set point that guarantees the greatest reduction in clustering error when the new cluster is initialized to include this point and executes kernel k-means only once from this initialization. The second variant locates a set of exemplars in the data set, by fitting a convex mixture model [4], and then, for each intermediate problem, tries only the exemplars as possible initializations for the new cluster instead of the whole data set.

Manuscript received October 16, 2008; revised January 20, 2009; accepted March 18, 2009. First published May 29, 2009; current version published July 09, 2009.

The authors are with the Department of Computer Science, University of Ioannina, GR 45110 Ioannina, Greece (e-mail: gtzortzi@cs.uoi.gr; arly@cs.uoi.gr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2009.2019722

Graph partitioning algorithms focus on clustering the nodes of a graph. Spectral methods have been effectively applied for optimizing many graph cut objectives, including ratio association and normalized cut. The main drawback of these methods is the need to calculate the eigenvectors of the graph affinity matrix which has substantial cost for large graphs. Weighted kernel k-means [5], a version of kernel k-means that assigns different weights to each data set point, is closely related to graph partitioning as its objective becomes equivalent to many graph cut criteria if the weights and kernel are set appropriately [5]–[8]. Hence, it can be used to locally optimize many graph cut objectives. Based on that, we have developed weighted versions of the proposed methods and use them to locate near-optimal solutions in graph partitioning problems without the need to compute eigenvectors.

We present experimental results that compare global kernel k-means, its two variants and kernel k-means with multiple restarts on artificial data, digits, face images, and graphs. The results back our claim that global kernel k-means locates near-optimal solutions as it outperforms kernel k-means with multiple restarts in terms of clustering error. There are cases where the speeding up schemes prove equal to the original algorithm, while in most of the experiments, they are superior to the best run of kernel k-means and their clustering error is always lower compared to the average of kernel k-means.

In the following section, we formally define clustering error and briefly describe k-means, global k-means, and kernel k-means. In Section III, we present the proposed global kernel k-means algorithm along with an analysis of its computational complexity, while the speeding up schemes are described in Section IV. The weighted versions of the proposed methods are presented in Section V and their application to graph partitioning is discussed next. Section VII contains the experimental evaluation, while Section VIII concludes this work. A preliminary version of this work has been presented in [9].

II. Preliminaries

A. k-Means and Global k-Means

Suppose we are given a data set $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, $\mathbf{x}_n \in \Re^d$, and we aim to partition this data set into M disjoint clusters $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_M$. The k-means algorithm [1] finds local optimal solutions with respect to the clustering error, defined as the sum of squared Euclidean distances between each data point $\mathbf{x}_n$ and the center $\mathbf{m}_k$ of the cluster that $\mathbf{x}_n$ belongs to. Analytically, the clustering error is given by

$$E(\mathbf{m}_1, \ldots, \mathbf{m}_M) = \sum_{n=1}^{N} \sum_{k=1}^{M} I(\mathbf{x}_n \in \mathcal{C}_k) \|\mathbf{x}_n - \mathbf{m}_k\|^2 \quad (1)$$

where $I(Y) = 1$ if Y is true and 0 otherwise.

The two main disadvantages of the k-means algorithm are the dependence of the final solution on the initial position of the cluster centers and that the clusters must be linearly separable. To deal with the initialization problem, the global k-means algorithm has been proposed [2], which is an incremental–deterministic algorithm that employs k-means as a local search procedure. This algorithm obtains near-optimal solutions in terms of clustering error.

In order to solve the M-clustering problem using global k-means, we proceed as follows. As the optimal solution to the 1-clustering problem corresponds to the data set centroid, we begin by solving the 2-clustering problem. We run k-means N times, each time starting with the following initial cluster centers: one cluster center is always placed at the optimal position of the 1-clustering problem and the other, during the nth run, is initially placed at data point $\mathbf{x}_n$. The solution with the lowest clustering error is kept as the solution of the 2-clustering problem. In general, for the k-clustering problem, let $(\mathbf{m}_1^*, \ldots, \mathbf{m}_{k-1}^*)$ denote the solution to the $(k-1)$-clustering problem. We perform N executions of the k-means algorithm, with $(\mathbf{m}_1^*, \ldots, \mathbf{m}_{k-1}^*, \mathbf{x}_n)$ as initial cluster centers for the nth run, and keep the solution resulting in the lowest clustering error. The above procedure is repeated until $k = M$.

It is obvious that the above algorithm does not suffer from the initialization of the cluster centers problem and computes a clustering of the data points in a deterministic way. Also, it provides all solutions with $1, \ldots, M$ clusters when solving the M-clustering problem without additional cost. The experiments reported in [2] verify that global k-means is better than k-means with multiple restarts. A drawback of global k-means is that it is computationally heavy as it requires running the k-means algorithm MN times. To speed up its execution, two variants are proposed in [2] that do not considerably degrade the performance of the algorithm. Another variant has been recently proposed in [10].

B. Kernel k-Means

Kernel k-means [3] is a generalization of the standard k-means algorithm where data points are mapped from input space to a higher dimensional feature space through a nonlinear transformation ϕ and then k-means is applied in feature space. This results in linear separators in feature space which correspond to nonlinear separators in input space. Thus, kernel k-means avoids the limitation of linearly separable clusters in input space that k-means suffers from.

The objective function that kernel k-means tries to minimize is the clustering error in feature space, shown in (2). We can define a *kernel matrix* $K \in \Re^{N \times N}$, where $K_{ij} = \phi(\mathbf{x}_i)^{\mathrm{T}} \phi(\mathbf{x}_j)$ and by taking advantage of the *kernel trick*, we can compute the squared Euclidean distances in (2) without explicit knowledge of the transformation ϕ using (3). Any positive-semidefinite matrix can be used as a kernel matrix. Notice that in this case cluster centers $\mathbf{m}_k$ in feature space cannot be calculated. Usually, a *kernel function* $K(\mathbf{x}_i, \mathbf{x}_j)$ is used to directly provide the inner products in feature space without explicitly defining transformation ϕ (for certain kernel functions the corresponding transformation is intractable), hence $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$. Some kernel function examples are given in Table I. Kernel k-means is described in Algorithm 1.

Algorithm 1: Kernel k-Means

Input: Kernel matrix K, number of clusters k, initial clusters $\mathcal{C}_1, \ldots, \mathcal{C}_k$

Output: Final clusters $\mathcal{C}_1, \ldots, \mathcal{C}_k$, clustering error E

1: **for all** points $\mathbf{x}_n$ $n = 1, \ldots, N$ **do**

TABLE I
EXAMPLES OF KERNEL FUNCTIONS

Polynomial Kernel	$K(\mathbf{x}_i, \mathbf{x}_j) = \left(\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j + \gamma\right)^{\delta}$
Gaussian Kernel	$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2\right)$
Sigmoid Kernel	$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh\left(\gamma(\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j) + \theta\right)$

2: **for all** clusters $\mathcal{C}_i$ $i = 1$ to k **do**

3: Compute $||\phi(\mathbf{x}_n) - \mathbf{m}_i||^2$ using (3)

4: **end for**

5: Find $c^*(\mathbf{x}_n) = \arg\min_i(||\phi(\mathbf{x}_n) - \mathbf{m}_i||^2)$

6: **end for**

7: **for all** clusters $\mathcal{C}_i$ $i = 1$ to k **do**

8: Update cluster $\mathcal{C}_i = \{\mathbf{x}_n | c^*(\mathbf{x}_n) = i\}$

9: **end for**

10: **if** converged **then**

11: **return** final clusters $\mathcal{C}_1, \ldots, \mathcal{C}_k$ and E calculated using (2)

12: **else**

13: Go to step 1

14: **end if**

$$E(\mathbf{m}_1, \ldots, \mathbf{m}_M) = \sum_{n=1}^{N}\sum_{k=1}^{M} I(\mathbf{x}_n \in \mathcal{C}_k)\,||\phi(\mathbf{x}_n) - \mathbf{m}_k||^2 \quad (2)$$

where

$$\mathbf{m}_k = \frac{\sum_{n=1}^{N} I(\mathbf{x}_n \in \mathcal{C}_k)\phi(\mathbf{x}_n)}{\sum_{n=1}^{N} I(\mathbf{x}_n \in \mathcal{C}_k)}$$

$$||\phi(\mathbf{x}_n) - \mathbf{m}_k||^2 = K_{nn} - \frac{2\sum_{j=1}^{N} I(\mathbf{x}_j \in \mathcal{C}_k)K_{nj}}{\sum_{j=1}^{N} I(\mathbf{x}_j \in \mathcal{C}_k)} + \frac{\sum_{j=1}^{N}\sum_{l=1}^{N} I(\mathbf{x}_j \in \mathcal{C}_k)I(\mathbf{x}_l \in \mathcal{C}_k)K_{jl}}{\sum_{j=1}^{N}\sum_{l=1}^{N} I(\mathbf{x}_j \in \mathcal{C}_k)I(\mathbf{x}_l \in \mathcal{C}_k)}. \quad (3)$$

It can be shown that kernel k-means monotonically converges if the kernel matrix is positive semidefinite, i.e., is a *valid* kernel matrix. If the kernel matrix is not positive semidefinite, the algorithm may still converge, but this is not guaranteed. As for the computational complexity, in [6], it is shown that kernel k-means requires $O(N^2\tau)$ scalar operations, where τ is the number of iterations until convergence is achieved. If we also have to calculate the kernel matrix, $O(N^2 d)$ extra scalar operations are required.

It must be noted that, by associating a weight with each data point, the weighted kernel k-means algorithm is obtained [5]. It has been proved that its objective function is equivalent to that of many graph partitioning problems such as ratio association, normalized cut, etc., if the weights and the kernel matrix are set appropriately [5]–[7]. This subject is further analyzed in Sections V and VI.

It has also been proved that performing k-means in the kernel pca space is equivalent to kernel k-means [11]. However, this approach has two drawbacks: it requires computation of the eigenvectors of the kernel matrix and it highly depends on the initialization of k-means. Calculating the eigenvectors of large matrices is expensive and even prohibitive as it requires $O(N^3)$ operations.

III. GLOBAL KERNEL k-MEANS

In this paper, we propose the *global kernel k-means* algorithm for minimizing the clustering error in feature space (2). Our method builds on the ideas of global k-means and kernel k-means. Global kernel k-means maps the data set points from input space to a higher dimensional feature space with the help of a kernel matrix $K \in \Re^{N\times N}$, as kernel k-means does. In this way, *nonlinearly separable clusters* are found in input space. Also, global kernel k-means finds *near-optimal solutions* to the M-clustering problem by incrementally and deterministically adding a new cluster at each stage and by applying kernel k-means as a local search procedure instead of initializing all M clusters at the beginning of the execution. Thus, the problems of *cluster initialization* and getting trapped in *poor local minima* are also avoided. In a nutshell, global kernel k-means combines the advantages of both global k-means (near-optimal solutions) and kernel k-means (clustering in feature space).

A. Algorithm Description

Suppose we want to solve the M-clustering problem using global kernel k-means. Since the calculation of the cluster centers in feature space is intractable, for the same reason as for kernel k-means, we will represent a cluster in terms of the data points that belong to it instead of its center. We start by solving the 1-clustering problem using kernel k-means. The optimal solution to this problem is trivial as all data points are assigned to the same cluster. We continue with the 2-clustering problem where kernel k-means is executed N times. During the nth execution, the initialization is done by considering two clusters one of which contains only $\mathbf{x}_n$ and the other is $\mathcal{X} - \{\mathbf{x}_n\}$. Among the N solutions, the one with the lowest clustering error is kept as the solution with two clusters. In general, for the k-clustering problem, let $(\mathcal{C}_1^*, \ldots, \mathcal{C}_{k-1}^*)$ denote the solution with $k-1$ clusters and assume that $\mathbf{x}_n \in \mathcal{C}_r^*$. We perform N executions of the kernel k-means algorithm, with $(\mathcal{C}_1^*, \ldots, \mathcal{C}_r = \mathcal{C}_r^* - \{\mathbf{x}_n\}, \ldots, \mathcal{C}_{k-1}^*, \mathcal{C}_k = \{\mathbf{x}_n\})$ as initial clusters for the nth run, and keep the one resulting in the lowest clustering error. The above procedure is repeated until $k = M$. The global kernel k-means pseudocode is given in Algorithm 2.

Algorithm 2: Global Kernel k-Means

Input: Kernel matrix K, number of clusters M

Output: Final clustering of the points $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_M$

// There is no need to solve for one cluster as the solution is trivial and optimal. $\mathcal{C}_1^* = \mathcal{X}$

1: **for all** k-clustering problems $k = 2$ to M **do**

2: **for all** points $\mathbf{x}_n$ $n = 1, \ldots, N$ **do** // Suppose $\mathbf{x}_n \in \mathcal{C}_r^*$

3: Run Kernel k-Means with:

input $(K, k, \mathcal{C}_1^*, \ldots, \mathcal{C}_r = \mathcal{C}_r^* - \{\mathbf{x}_n\}, \ldots, \mathcal{C}_{k-1}^*, \mathcal{C}_k = \{\mathbf{x}_n\})$

output $(\mathcal{C}_1^n, \ldots, \mathcal{C}_k^n, E_k^n)$

4: **end for**

5: Find $E_k^* = \min_n(E_k^n)$ and set $(\mathcal{C}_1^*, \ldots, \mathcal{C}_k^*)$ to the partitioning corresponding to E_k^*
// This is the solution with k clusters

6: **end for**

7: **return** $\mathcal{C}_1 = \mathcal{C}_1^*, \ldots, \mathcal{C}_M = \mathcal{C}_M^*$ as output of the algorithm

The rationale behind the proposed method is based on the assumption that a near-optimal solution with k clusters can be obtained through local search, from an initial state with $k-1$ near-optimal clusters [solution of the $(k-1)$-clustering problem] and the kth cluster initialized appropriately. As we consider only one data point belonging to the kth cluster when it is initialized, this is equivalent to initializing, during the nth run, the kth cluster center at point $\phi(\mathbf{x}_n)$ in feature space. Limiting the set of possible positions for the kth center only to the data set points when mapped to feature space seems reasonable. Our experiments verify that the proposed algorithm computes near-optimal solutions, although it is difficult to prove theoretically. Note that during the execution of the algorithm, solutions for every k-clustering problem with $k < M$ are also obtained without additional cost, which may be desirable in case we want to decide on the number of clusters for our problem.

B. Computational Complexity

Due to its close relation to global k-means and kernel k-means, the global kernel k-means algorithm inherits their computational cost. Given the values of the kernel matrix, the complexity of kernel k-means is $O(N^2\tau)$ scalar operations, where τ is the number of iterations until convergence is achieved. In the global kernel k-means algorithm, in order to solve the M-clustering problem, we must run kernel k-means MN times. This makes the complexity of global kernel k-means $O(N^3M\tau)$. If we also have to calculate the kernel matrix, extra $O(N^2d)$ scalar operations are required, making the overall complexity $O(N^2(NM\tau + d))$. Storage of the matrix requires $O(N^2)$ memory and a scheme for dealing with insufficient memory has been proposed in [12], which can be readily applied to our algorithm. As the computational complexity is high for large data sets two speeding up schemes are proposed next.

IV. Speeding-Up Execution

A. Fast Global Kernel k-Means

The fast global kernel k-means algorithm is a simple method for lowering the complexity of the original algorithm. It is based on the same ideas as the fast global k-means variant proposed in [2]. We significantly reduce the complexity by overcoming the need to execute kernel k-means N times when solving the k-clustering problem given the solution for the $(k-1)$-clustering problem. Specifically, kernel k-means is employed *only once* and the kth cluster is initialized to include the point $\mathbf{x}_n$ that *guarantees* the greatest reduction in clustering error.

In more detail, we compute an *upper bound* $E_k^n \leq E_{k-1}^* - b_k^n$ of the final clustering error, when the kth cluster is initialized to include point $\mathbf{x}_n$. E_{k-1}^* is the clustering error corresponding to the $(k-1)$-clustering problem solution and b_k^n measures the guaranteed reduction of the error and is defined in (4). In this equation, d_{k-1}^i denotes the squared distance between $\mathbf{x}_i$ and its cluster center in feature space after solving the $(k-1)$-clustering problem. We select as initialization for the kth cluster the point $\mathbf{x}_n$ with the highest b_k^n value

$$b_k^n = \sum_{i=1}^{N} \max\left(d_{k-1}^i - \|\phi(\mathbf{x}_n) - \phi(\mathbf{x}_i)\|^2, 0\right)$$

where

$$\|\phi(\mathbf{x}_n) - \phi(\mathbf{x}_i)\|^2 = K_{nn} + K_{ii} - 2K_{ni}. \tag{4}$$

The above upper bound is derived from the following arguments. First, when the kth cluster is initialized to include point $\mathbf{x}_n$, it will allocate all points $\mathbf{x}_i$ that are closer to $\mathbf{x}_n$ in feature space than to their cluster center in the solution with $k-1$ clusters (distance d_{k-1}^i). Therefore, for each such point $\mathbf{x}_i$, the clustering error will decrease by $d_{k-1}^i - \|\phi(\mathbf{x}_n) - \phi(\mathbf{x}_i)\|^2$. The quantity b_k^n measures the reduction in error due to this reallocation. Second, since kernel k-means converges monotonically, we can be sure that the final error will never exceed our bound.

When using this variant of the global kernel k-means algorithm to solve the M-clustering problem, we must execute kernel k-means M times instead of MN times. Given the kernel matrix, calculation of b_k^n requires $O(N)$ scalar operations as d_{k-1}^i is calculated when executing kernel k-means for the $(k-1)$-clustering problem. Each time we have to calculate N quantities b_k^n and this must be repeated M times in order to solve the problem with M clusters. Thus, the cost incurred by the need to estimate the upper bound is $O(N^2M)$. Overall, the fast global kernel k-means algorithm has $O(N^2(M\tau + d + M)) = O(N^2(M\tau + d))$ complexity, which is considerably lower than that of global kernel k-means and is comparable to the complexity of kernel k-means when M is small. In general, this reduction in complexity comes at the cost of finding solutions with higher clustering error than the original algorithm. However, as our experiments indicate, in several problems, the performance of the fast version is comparable to that of global kernel k-means, which makes it a good fast alternative.

B. Global Kernel k-Means With Convex Mixture Models

Another way, apart from fast global kernel k-means, to lower the complexity of the global kernel k-means algorithm when solving the M-clustering problem, is to select a set of good exemplars ($\mathcal{X}_E \subset \mathcal{X}$) in feature space, with an exemplar-based

method, and then apply global kernel k-means with the restriction that the points tried as initializations for the newly added cluster come from set $\mathcal{X}_E$ only.

In order to solve the k clustering problem, given the solution with $k-1$ clusters, we must run kernel k-means as many times as the number of selected exemplars instead of N times. In each of these runs, we initialize the kth cluster to include an exemplar of those selected. Many algorithms that perform exemplar-based clustering can be used for this purpose, such as affinity propagation [13]. In our work, we use the method proposed by Lashkari *et al.* [4], which is based on the use of exemplar-based mixture models also called convex mixture models (CMMs).

Clustering with a convex mixture model results in soft assignments of data points to clusters in such a way that the likelihood is maximized, as with standard mixture models. Given a data set $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, $\mathbf{x}_i \in \Re^d$, Lashkari *et al.* [4] seek to maximize the following log-likelihood function:

$$\begin{aligned} L\left(\{q_j\}_{j=1}^N; \mathcal{X}\right) &= \frac{1}{N}\sum_{i=1}^{N} \log\left[\sum_{j=1}^{N} q_j f_j(\mathbf{x}_i)\right] \\ &= \frac{1}{N}\sum_{i=1}^{N} \log\left[\sum_{j=1}^{N} q_j e^{-\beta d_\varphi(\mathbf{x}_i, \mathbf{x}_j)}\right] \\ &\quad + \text{const.} \end{aligned} \tag{5}$$

where q_j denotes the prior probability of the jth component satisfying the constraint $\sum_{j=1}^{N} q_j = 1$, $f_j(\mathbf{x})$ is an exponential family distribution on random variable $\mathbf{X}$ with its expectation parameter equal to the jth data point and, taking into account the bijection between regular exponential families, and Bregman divergences [14], d_φ is the Bregman divergence corresponding to the components' distribution, i.e., $f_j(\mathbf{x}) = C(\mathbf{x})\exp(-\beta d_\varphi(\mathbf{x}, \mathbf{x}_j))$. Note that the exponential family distributions used in the components are centered at the data set points and have no adjustable parameters. Moreover, a convex mixture model has as many components as the number of data set points, hence it represents all data points as cluster center candidates (candidate exemplars). The constant β controls the sharpness of the components and $0 < \beta < \infty$. Also, this parameter *controls the number of clusters* identified by the algorithm when the soft assignments are turned into hard ones. Higher β values result in more clusters in the final solution. To solve the above optimization problem, we maximize $L(\{q_j\}_{j=1}^N; \mathcal{X})$ over $\{q_j\}_{j=1}^N$ s.t. $\sum_{j=1}^{N} q_j = 1$, which are the only unknown parameters of the simplified likelihood function.

The log-likelihood function (5) can be expressed in terms of the Kullback–Leibler (KL) divergence if we define $\hat{P}(\mathbf{x}) = 1/N$, $\mathbf{x} \in \mathcal{X}$, to be the empirical distribution of the data set, $Q(\mathbf{x}) = \sum_{j=1}^{N} q_j f_j(\mathbf{x})$ the mixture model distribution, and by noting that

$$\begin{aligned} D(\hat{P}\|Q) &= -\sum_{\mathbf{x}\in\mathcal{X}} \hat{P}(\mathbf{x}) \log Q(\mathbf{x}) - \mathbb{H}(\hat{P}) \\ &= -L\left(\{q_j\}_{j=1}^N; \mathcal{X}\right) + \text{const.} \end{aligned} \tag{6}$$

where $\mathbb{H}(\hat{P})$ is the entropy of the empirical distribution that does not depend on the parameters of the mixture model. Now the maximization of (5) is equivalent to the minimization of (6). This minimization problem is *convex* and can be solved with an efficient-iterative algorithm. As proved in [15], the updates on the components' prior probabilities are given by

$$q_j^{(t+1)} = q_j^{(t)} \sum_{\mathbf{x}\in\mathcal{X}} \frac{\hat{P}(\mathbf{x}) f_j(\mathbf{x})}{\sum_{j'=1}^{N} q_{j'}^{(t)} f_{j'}(\mathbf{x})} \tag{7}$$

and the algorithm is guaranteed to converge to the global minimum as long as $q_j^{(0)} > 0\ \forall j$. The prior q_j associated with data point $\mathbf{x}_j$ is a measure of *how likely this point is to be an exemplar*. This is an important fact for our integration of convex mixture models with global kernel k-means.

The convex mixture model as presented in [4] performs clustering in input space. In this work, we propose a slight modification so as to perform clustering in feature space. For this reason, we use the same nonlinear transformation ϕ as in kernel k-means to map the data points to feature space and the log-likelihood (5) becomes

$$\begin{aligned} L\left(\{q_j\}_{j=1}^N; \mathcal{X}\right) &= \frac{1}{N}\sum_{i=1}^{N} \log\left[\sum_{j=1}^{N} q_j f_j\left(\phi(\mathbf{x}_i)\right)\right] \\ &= \frac{1}{N}\sum_{i=1}^{N} \log\left[\sum_{j=1}^{N} q_j e^{-\beta d_\varphi(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j))}\right] \\ &\quad + \text{const.} \end{aligned} \tag{8}$$

where $f_j(\phi(\mathbf{x}))$ is an exponential family distribution with its expectation parameter equal to the mapping of the jth data point in feature space. If we select Euclidean distance as the Bregman divergence, as in our experiments, then we obtain a Gaussian convex mixture model in feature space and $d_\varphi(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j))$ can be expressed in terms of a kernel function as

$$d_\varphi\left(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\right) = \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2 = K_{ii} + K_{jj} - 2K_{ij}. \tag{9}$$

Given the convex mixture model corresponding to the likelihood defined in (8) we now proceed to its integration with global kernel k-means. Let us note that to get the exemplars in feature space the likelihood is optimized only once. For the implementation of the algorithm that optimizes (8), we follow the same steps as in [4]. Letting $s_{ij} = \exp(-\beta d_\varphi(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)))$ and using two auxiliary vectors z and n, we update the prior probabilities q_j as follows:

$$z_i^{(t)} = \sum_{j=1}^{N} s_{ij} q_j^{(t)} \quad n_j^{(t)} = \frac{1}{N}\sum_{i=1}^{N} \frac{s_{ij}}{z_i^{(t)}} \quad q_j^{(t+1)} = n_j^{(t)} q_j^{(t)} \tag{10}$$

where $q_j^{(0)} > 0$ for all data points we want to consider as possible exemplars. As our target is to identify a number of good exemplars, say P exemplars, we run the algorithm until the P highest q_j values correspond to the same data points for a number of consecutive iterations. Moreover, we require that the order among the P highest q_j values remains the same during these iterations. Note that this convergence criterion differs from that proposed in [4]. The points of the data set that correspond to the P highest q_j values are those considered as good exemplars and are included in set $\mathcal{X}_E$.

Clustering with the convex mixture model corresponding to likelihood (8) requires to select a value for the parameter β. It is possible to identify a reasonable range of β values by determining a reference value β_0. For this purpose, we choose the empirical value proposed in [4], but appropriately modified for clustering in feature space, defined in (11). In our experiments, we set $\beta = \beta_0$ and obtain good solutions in most cases

$$\beta_0 = N^2 \log N / \sum_{i,j} d_\varphi \left(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\right). \tag{11}$$

After we have determined the set of exemplars $\mathcal{X}_E$, we run global kernel k-means. We will refer to the data points belonging to $\mathcal{X}_E$ as $\mathbf{x}_j^E$, $j = 1, \ldots, P$. Now suppose we want to solve the M-clustering problem. We must solve all intermediate problems with $1, \ldots, M$ clusters but now instead of trying N different initializations for the newly added cluster, one for each point in $\mathcal{X}$, we try only P initializations, one for each point in $\mathcal{X}_E$. In more detail, for the k-clustering problem, let $(\mathcal{C}_1^*, \ldots, \mathcal{C}_{k-1}^*)$ denote the solution with $k-1$ clusters and assume that $\mathbf{x}_p^E \in \mathcal{C}_r^*$. We perform P executions of the kernel k-means algorithm with $(\mathcal{C}_1^*, \ldots, \mathcal{C}_r = \mathcal{C}_r^* - \{\mathbf{x}_p^E\}, \ldots, \mathcal{C}_{k-1}^*, \mathcal{C}_k = \{\mathbf{x}_p^E\})$ as initial clusters for the pth run, and keep the one resulting in the lowest clustering error as the solution with k clusters. Remember that initializing cluster $\mathcal{C}_k$ to contain point $\mathbf{x}_p^E$ during the pth run is the same as placing the kth center at point $\phi(\mathbf{x}_p^E)$ in feature space. The above procedure is repeated for $k = 2, \ldots, M$.

When using this variant of global kernel k-means, we select P exemplars in order to solve the M-clustering problem. The idea behind this is that the exemplars identified by the convex mixture model contain most of the information required by global kernel k-means to partition the data set. So by trying only the P exemplars as possible initializations for the newly added cluster, we hope to approximate the solution of the original global kernel k-means algorithm. Usually one should choose $P > M$, but $P \ll N$ because in most cases only a small subset of the data set contains useful (nonredundant) information for the partitioning.

As for the complexity of this variant, we must run kernel k-means MP times instead of MN times to solve the M-clustering problem. This is a great reduction, especially for large data sets, if we consider that $P \ll N$ for the aforementioned reasons. Of course, there is the additional cost of identifying the exemplars. The algorithm (10) considered here has a complexity of $O(N^2)$ per iteration [4]. Moreover, the calculation of the quantities s_{ij} requires an additional $O(N^2)$ operations if the kernel matrix is given. If the algorithm requires τ' iterations to converge, the overall cost becomes $O(N^2\tau')$. The global kernel k-means algorithm has a cost of $O(N^2(PM\tau + d))$ in this case (since we consider P exemplars) including the kernel calculation. So the overall cost of the proposed variant is $O(N^2(PM\tau + d + \tau'))$. Its complexity is considerably lower to that of the original algorithm when $P \ll N$. Our experiments indicate that despite the reduction to the complexity, the quality of the solutions is satisfactory, and in many cases, very similar to that of the original algorithm. Finally, its complexity is higher than that of fast global kernel k-means, but it finds better solutions that are closer to those of the original algorithm as shown in our experiments. This happens because this method is less greedy, as it tries P initializations when solving the k-clustering problem instead of one.

V. Adding Weights to Global Kernel k-Means and Its Speeding-Up Schemes

A. Weighted Kernel k-Means

If we associate a positive weight with each data point the weighted kernel k-means algorithm is derived [5]. The weights play a crucial role in proving an equivalence of clustering to graph partitioning, which is the reason we are interested in this version of kernel k-means. Again, suppose we want to solve the M-clustering problem. The objective function is expressed as follows, where $w_i > 0$ is the weight associated with data point $\mathbf{x}_i$:

$$E(\mathbf{m}_1, \ldots, \mathbf{m}_M) = \sum_{i=1}^{N} \sum_{k=1}^{M} I(\mathbf{x}_i \in \mathcal{C}_k) w_i \|\phi(\mathbf{x}_i) - \mathbf{m}_k\|^2$$

where

$$\mathbf{m}_k = \frac{\sum_{i=1}^{N} I(\mathbf{x}_i \in \mathcal{C}_k) w_i \phi(\mathbf{x}_i)}{\sum_{i=1}^{N} I(\mathbf{x}_i \in \mathcal{C}_k) w_i}. \tag{12}$$

Note that the center of the cluster in feature space is the weighted average of the points that belong to the cluster. Once again, we can take advantage of the kernel trick and calculate the squared Euclidean distances in (12) without explicitly defining transformation ϕ, using (13). Algorithm 1 can be applied for this version of kernel k-means by only modifying lines 3 and 11, where the point to cluster center distances must be calculated using (13) and the clustering error using (12)

$$\|\phi(\mathbf{x}_i) - \mathbf{m}_k\|^2 = K_{ii} - \frac{2\sum_{j=1}^{N} I(\mathbf{x}_j \in \mathcal{C}_k) w_j K_{ij}}{\sum_{j=1}^{N} I(\mathbf{x}_j \in \mathcal{C}_k) w_j} + \frac{\sum_{j=1}^{N}\sum_{l=1}^{N} I(\mathbf{x}_j \in \mathcal{C}_k) I(\mathbf{x}_l \in \mathcal{C}_k) w_j w_l K_{jl}}{\sum_{j=1}^{N}\sum_{l=1}^{N} I(\mathbf{x}_j \in \mathcal{C}_k) I(\mathbf{x}_l \in \mathcal{C}_k) w_j w_l}. \tag{13}$$

B. Weighted Global Kernel k-Means

It is straightforward that we can use the global kernel k-means algorithm to optimize the objective function defined in (12) by associating a weight with each data point. Algorithm 2 can be applied with the slightest modification to get the weighted global kernel k-means algorithm. Specifically, we must include on the input the weights and run weighted kernel k-means instead of kernel k-means in line 3. All other steps remain the same.

C. Weighted Fast Global Kernel k-Means

Fast global kernel k-means can also be extended to handle weights for each data point. Again, we must run weighted kernel k-means instead of kernel k-means, but the issue that changes is the way we select the point that guarantees the greatest reduction in clustering error when solving the k-clustering problem given the solution with $k-1$ clusters. This point will be used to initialize the kth cluster for the one and only run of weighted kernel

k-means. Now the weights must be taken into account, resulting in a modified definition of the quantity b_k^n (14) that measures the reduction in clustering error when the kth cluster is initialized to include point $\mathbf{x}_n$. Again, d_{k-1}^i denotes the squared distance between $\mathbf{x}_i$ and its cluster center in feature space after solving the $(k-1)$-clustering problem

$$b_k^n = \sum_{i=1}^{N} w_i \max\left(d_{k-1}^i - \|\phi(\mathbf{x}_n) - \phi(\mathbf{x}_i)\|^2, 0\right). \quad (14)$$

The above definition measures the reduction in clustering error by identifying the points that will be allocated by the new cluster center $\phi(\mathbf{x}_n)$. For each such point $\mathbf{x}_i$, the clustering error will decrease by $w_i(d_{k-1}^i - \|\phi(\mathbf{x}_n) - \phi(\mathbf{x}_i)\|^2)$. Apparently, we select as initialization for the kth cluster the point $\mathbf{x}_n$ that maximizes b_k^n. No other modifications are required to obtain the weighted fast global kernel k-means algorithm.

D. Weighted Global Kernel k-Means With Convex Mixture Models

The global kernel k-means with CMM algorithm is divided into two parts. The first is the identification of a set $\mathcal{X}_E$ of good exemplars and the second the application of global kernel k-means, where for the newly added cluster, we try only the exemplars in $\mathcal{X}_E$ as possible initializations. It is obvious that weights can be applied to the second part, as shown in Section V-B. Now we focus our description on the first part and propose *a modification to the convex mixture model algorithm in order to accommodate weights*. We can incorporate the weights into the convex mixture model objective through the empirical distribution of the data set. More specifically, instead of a uniform distribution, we use

$$\hat{P}(\mathbf{x}) = \begin{cases} \dfrac{w_i}{\sum_{i'=1}^{N} w_{i'}}, & \mathbf{x} \in \mathcal{X} \\ 0, & \text{otherwise.} \end{cases} \quad (15)$$

If $Q(\mathbf{x}) = \sum_{j=1}^{N} q_j f_j(\phi(\mathbf{x}))$ is the convex mixture model distribution in feature space, we can write the convex mixture model objective in terms of the KL divergence as

$$D(\hat{P}\|Q) = -\sum_{\mathbf{x}\in\mathcal{X}} \hat{P}(\mathbf{x}) \log Q(\mathbf{x}) - \mathbb{H}(\hat{P}). \quad (16)$$

This is the same objective as for the unweighted case but with (15) as the empirical distribution. It can be proved that the minimization of (16) can be performed in the same way as in the unweighted case, thus the updates on the components' priors probabilities are given by

$$q_j^{(t+1)} = q_j^{(t)} \sum_{\mathbf{x}\in\mathcal{X}} \frac{\hat{P}(\mathbf{x}) f_j\left(\phi(\mathbf{x})\right)}{\sum_{j'=1}^{N} q_{j'}^{(t)} f_{j'}\left(\phi(\mathbf{x})\right)} \quad (17)$$

with $\hat{P}(\mathbf{x})$ defined in (15). We can use the algorithm described in (10) to update the priors with a slight modification on $n_j^{(t)}$, now defined as

$$n_j^{(t)} = \sum_{i=1}^{N} \hat{P}(\mathbf{x}_i) \frac{s_{ij}}{z_i^{(t)}}. \quad (18)$$

The change on the empirical data set distribution results in a change on the formula for the empirical value of the β parameter, now given by (19). The derivation can be found in the Appendix

$$\beta_0 = N \frac{-\sum_{i=1}^{N} \hat{P}(\mathbf{x}_i) \log \hat{P}(\mathbf{x}_i)}{\sum_{i,j=1}^{N} \hat{P}(\mathbf{x}_i) d_\varphi\left(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\right)}. \quad (19)$$

In summary, to run weighted global kernel k-means with CMM, one must define an empirical distribution of the form (15), in order to incorporate the weights into the convex mixture model objective, and find the mixture components prior probabilities q_j using the updates described in (17). A good range of values for the β parameter is determined through β_0, defined in (19). After identifying the set of exemplars $\mathcal{X}_E$, we run weighted global kernel k-means as described in Section V-B.

VI. Graph Partitioning

Graph partitioning is another approach to clustering data. In this problem, we are given a graph $G = (\mathcal{V}, \mathcal{E}, A)$, where $\mathcal{V}$ denotes the set of vertices, $\mathcal{E}$ is the set of edges, and A is an $|\mathcal{V}| \times |\mathcal{V}|$ affinity matrix which contains the pairwise similarities of the vertices, i.e., the weights of the edges and we aim to partition the graph into M disjoint clusters such that $\mathcal{V}_1 \cup \mathcal{V}_2 \cup \ldots \cup \mathcal{V}_M = \mathcal{V}$.

A number of different graph partitioning objectives have been proposed such as ratio association, ratio cut, normalized cut, etc. Usually spectral methods, like the one in [16], are employed to solve these problems by calculating the eigenvectors of the affinity matrix. Spectral methods perform well because they compute globally optimal solutions of a *relaxation* of the problem considered. Calculating the eigenvectors of large matrices, i.e., for graphs with many vertices, is computationally expensive, as it requires $O(N^3)$ operations, and may also be infeasible. In [5]–[7], it is proved that the weighted kernel k-means objective is equivalent to that of many graph cut problems if the weights and kernel are set appropriately. The proof is based on formulating the problems as trace maximizations following a similar approach to [17], where the k-means objective is formulated as a trace maximization. Weighted kernel k-means avoids the need to calculate eigenvectors but cannot find the optimal solution because it depends on cluster initialization. Even when calculation of eigenvectors is possible, the experiments in [6] show that weighted kernel k-means can further improve the clustering result produced by spectral methods.

As discussed in Section V, weighted global kernel k-means and its two variants can also be used to optimize (12). Hence, these algorithms can also be applied to graph partitioning and incorporate the advantages they bring over kernel k-means to this task. As shown in our experiments, these algorithms outperform weighted kernel k-means on graph partitioning in most cases and thus can be considered as a good alternative to spectral methods, especially the two variants which demonstrate good solution quality with low computational cost. In the following, we focus on the ratio association and normalized cut problems, which we experimented with, and describe how the weights and kernel must be set to obtain the equivalence. The reader is referred to [6] and [7] for a thorough analysis.

Let $\text{links}(\mathcal{A}, \mathcal{B})$ denote the sum of the edge weights between nodes in $\mathcal{A}$ and $\mathcal{B}$, i.e., $\text{links}(\mathcal{A}, \mathcal{B}) = \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{B}} A_{ij}$ and let $\text{degree}(\mathcal{A})$ denote the sum of the edge weights between nodes in $\mathcal{A}$ and all vertices, i.e., $\text{degree}(\mathcal{A}) = \text{links}(\mathcal{A}, \mathcal{V})$. Also let D be the diagonal $|\mathcal{V}| \times |\mathcal{V}|$ degree matrix where $D_{ii} = \sum_{j=1}^{|\mathcal{V}|} A_{ij}$.

1) Ratio Association: The ratio association problem tries to maximize within cluster association relative to the cluster size. The objective function is

$$RA(G) = \max_{\mathcal{V}_1, \ldots, \mathcal{V}_M} \sum_{i=1}^{M} \frac{\text{links}(\mathcal{V}_i, \mathcal{V}_i)}{|\mathcal{V}_i|}. \tag{20}$$

As proved in [6] and [7], to make the objective function of weighted kernel k-means equivalent to that of ratio association, we must set $w_i = 1$ and $K = A$. As $w_i = 1$, this is practically reduced to the unweighted version of the algorithms where the affinity matrix is in place of the kernel matrix. Obviously data set $\mathcal{X}$ contains the nodes and $N = |\mathcal{V}|$.

2) Normalized Cut: The normalized cut problem aims to minimize the cut between clusters and the remaining vertices, relative to the degree of the cluster. This objective is very popular in graph partitioning [18], [19] and is defined as

$$NC(G) = \min_{\mathcal{V}_1, \ldots, \mathcal{V}_M} \sum_{i=1}^{M} \frac{\text{links}(\mathcal{V}_i, \mathcal{V} \setminus \mathcal{V}_i)}{\text{degree}(\mathcal{V}_i)}. \tag{21}$$

As shown in [5]–[7], to make the objective function of weighted kernel k-means equivalent to that of normalized cut, we must set $w_i = D_{ii}$ and $K = D^{-1}AD^{-1}$. Obviously data set $\mathcal{X}$ contains the nodes and $N = |\mathcal{V}|$.

The above definitions of the kernel matrix do not guarantee that it will be positive semidefinite (i.e., valid), a condition that is sufficient but not necessary to guarantee convergence of the discussed algorithms. A workaround for this problem is proposed in [6] which relies on a diagonal shift of the kernel matrix. More specifically, for the ratio association problem, we set $K = \lambda I + A$, where I is the identity matrix and λ is a constant large enough to ensure that K is positive semidefinite. For the normalized cut problem, we set $K = \lambda D^{-1} + D^{-1}AD^{-1}$. A good issue is that this modification of the kernel matrix does not change the objective of the optimization problem we solve, although it can degrade the performance of the algorithms, as shown in [6], when the shift is too large.

VII. Experimental Evaluation

In this section, we extensively study the performance of global kernel k-means, its two variants, and kernel k-means with multiple restarts by applying these algorithms on many clustering problems.[1] These problems range from artificial data sets to handwritten digits, face images, and graphs. Our aim is to test the effectiveness of the proposed global kernel k-means algorithm and its two variants on different tasks, examine if indeed the original algorithm avoids getting trapped in poor local minima, and also discover how close the solutions of the speeding-up schemes are to those of the original algorithm. In all the experiments, except for graph partitioning, the Gaussian kernel is used and its parameter σ value is empirically determined. Note that appropriate kernel selection is out of the scope of this work and that is the reason we only try the Gaussian kernel with prespecified σ value. The experiments were conducted on a computer with 3.5-GB RAM and 2.4-GHz Intel Core 2 Duo processor and the code[2] was implemented in MATLAB.

A. Artificial Data Sets

We compare the four clustering algorithms on two artificial data sets that are not linearly separable, hence any algorithm, such as k-means, which identifies linearly separable clusters in input space is inappropriate. To obtain a meaningful comparison, the same σ value is used for the Gaussian kernel in all four methods. For the global kernel k-means with CMM, we set $\beta = \beta_0$ [see (11)], $s_{ij} = \exp(-\beta \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2)$, and select twice as many exemplars as the number of clusters ($P = 2M$). The quality of the solutions produced by the algorithms is evaluated in terms of clustering error. As global kernel k-means and its two variants perform clustering deterministically they are run once. On the other hand, kernel k-means is restarted 100 times and we report the average and minimum clustering error for the 100 runs.

The first data set (Fig. 1) consists of five copies of two rings where the inner ring is dense and has 700 points while the outer ring has 300 points. The whole data set has 5000 points and ten clusters. For this difficult clustering problem, global kernel k-means manages correctly to discriminate ten rings as shown in Fig. 1(a). Also, global kernel k-means with CMM identifies the ten rings correctly. This demonstrates that the exemplars determined through the CMM algorithm capture most of the information required to partition the data set. The solution corresponding to those two algorithms is the one with the lowest clustering error in Table II. Fast global kernel k-means fails to correctly identify all the rings: it splits the outer ring into two parts and merges one of them with the inner ring as illustrated in Fig. 1(b). As the problem is quite difficult to solve, the greedy decisions made by this algorithm result in suboptimal solutions with higher clustering error. Finally, kernel k-means never splits correctly the ten rings during the 100 runs and also note that its average clustering error is higher than that of fast global kernel k-means. Fig. 1(c) depicts the clustering result corresponding to the lowest clustering error during the 100 runs of kernel k-means.

The second data set (Fig. 2) contains three rings with a total of 550 points and we perform clustering into three clusters. Global kernel k-means identifies the three rings as shown in Fig. 2(a), which is also the solution with the lowest clustering error in Table II. This problem is even more difficult than the previous one, as it is proved by the fact that global kernel k-means with CMM fails to discriminate the three rings. This algorithm splits the middle ring into two parts and merges one of them with the outer ring as illustrated in Fig. 2(b). Fast global kernel k-means provides a similar "optical" result with global kernel k-means with CMM [Fig. 2(c)], but with higher clustering error indicating that the exemplars provide higher quality information for

[1]For better understanding of the experimental results, it is suggested to view the figures in color.

[2]The code is available from http://www.cs.uoi.gr/~gtzortzi

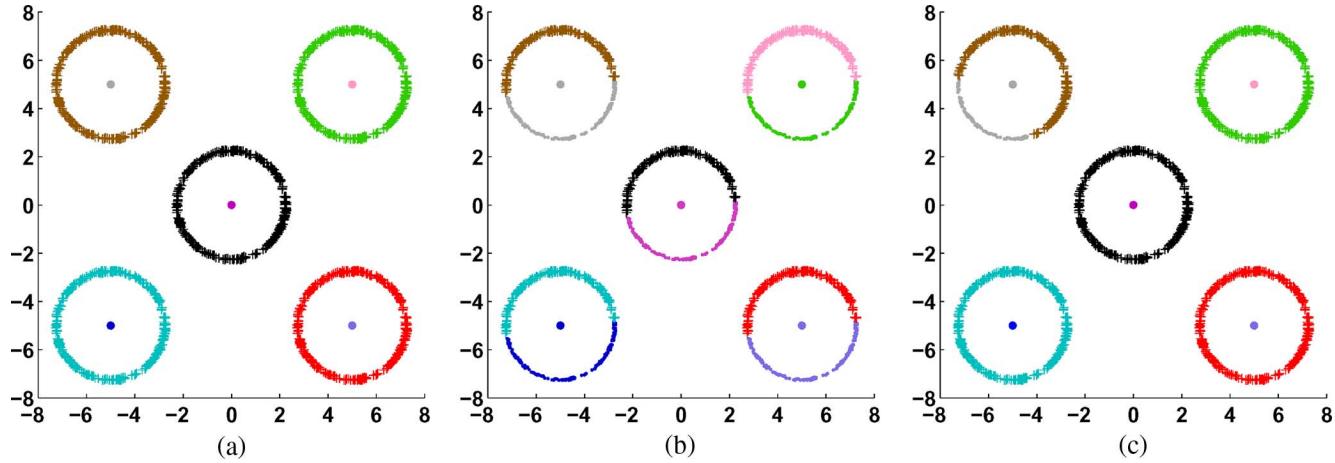

Fig. 1. Clustering results for the ten rings data set. (a) Global kernel k-means and global kernel k-means with CMM. (b) Fast global kernel k-means. (c) Kernel k-means (the run with minimum clustering error).

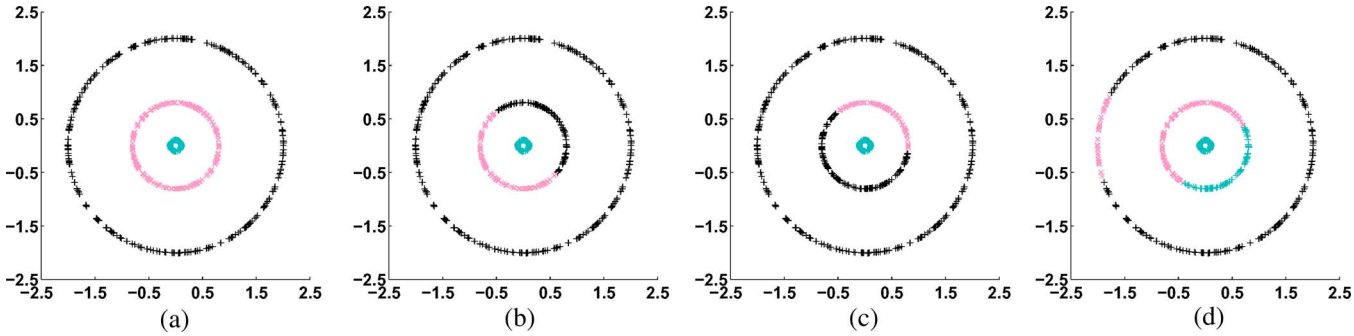

Fig. 2. Clustering results for the three rings data set. (a) Global kernel k-means. (b) Global kernel k-means with CMM. (c) Fast global kernel k-means. (d) Kernel k-means (the run with minimum clustering error).

TABLE II
ARTIFICIAL DATA SETS RESULTS IN TERMS OF CLUSTERING ERROR

Method/Dataset		Three Rings $\sigma = 0.55$	Ten Rings $\sigma = 1.8$
Global Kernel *k*-Means		**365.88**	**966.87**
Global Kernel *k*-Means with CMM		368.74	**966.87**
Fast Global Kernel *k*-Means		370.4	1073.18
Kernel *k*-Means (100 runs)	**Average**	378.68	1107.97
	Min	373.09	981.53

splitting the data set. Finally, kernel k-means fails to locate the rings correctly during the 100 runs and even its best run, depicted in Fig. 2(d), has a clustering error which is worse than that of all the other methods.

Overall, global kernel k-means achieves the lowest clustering error for both data sets and none of the 100 runs of kernel k-means provides a better result. This observation together with the fact that it correctly locates the rings in both data sets, which are difficult clustering problems, justify our claim that it provides near-optimal solutions. Also, the global kernel k-means with CMM algorithm provides identical results to the original algorithm for the first data set and is the second best method for the other data set, thus making it a good alternative with lower computational cost. To further demonstrate the potential of this method, we reran the experiment for the ten rings by selecting fewer exemplars (equal to the number of clusters, i.e., $P = M$) and again the rings were correctly identified. This further supports the fact that the exemplars identified are of very good quality. Fast global kernel k-means may be inferior to the original algorithm and the other variant for these hard problems, but is still superior over kernel k-means. As expected, kernel k-means is very sensitive to initialization and during the 100 runs finds from near-optimal solutions to very bad ones for both data sets. Therefore, it makes it difficult to decide on the sufficient number of restarts. As a final comment, we would like to stress that data sets consisting of concentric rings, like the above, can be clustered more effectively by using kernels based on geodesic distance instead of the Gaussian kernel, as in [20].

B. Handwritten Digits

For handwritten digits recognition, we selected the Pendigits and the Multiple Features data sets, which are available from the University of California at Irvine (UCI) repository [21]. The Pendigits data set contains 7494 training digits and 3498 testing digits represented as vectors in 16-dimensional space. The Multiple Features data set consists of 200 digits per class that are represented in terms of various feature sets. We have selected two of them, the first describing the digits using profile correlations (216 features) and the other using Zernike moments (47

TABLE III
HANDWRITTEN DIGITS RESULTS IN TERMS OF CLUSTERING ERROR (CE) AND NORMALIZED MUTUAL INFORMATION (NMI)

Method/Dataset		Pendigits Full $\sigma = 2.1$		Pendigits.tes $\sigma = 2.8$		Profile Correlations $\sigma = 9$		Zernike Moments $\sigma = 6.5$	
		CE	NMI	CE	NMI	CE	NMI	CE	NMI
Global Kernel *k*-Means		-	-	-	-	**1310.77**	**0.789**	**790.43**	0.507
Global Kernel *k*-Means with CMM		6514.95	0.776	1490.44	0.749	1311.5	0.773	790.48	0.506
Fast Global Kernel *k*-Means		6514.95	0.776	1504.81	0.75	1325.82	0.752	791.06	**0.531**
Kernel *k*-Means (100 runs)	**Average**	6668.34	0.739	1537.69	0.713	1330.81	0.729	797.58	0.502
	Min	**6514.94**	**0.777**	**1485.2**	**0.754**	1310.79	0.787	790.45	0.503

features). For all data sets, the class label of the digits is also available (0-9). We test the global kernel k-means with CMM and fast global kernel k-means algorithms on the Multiple Features, the whole Pendigits, and the Pendigits.tes (testing digits only) data sets. We do not test the global kernel k-means algorithm on Pendigits as its size (10 992 digits) makes the algorithm's application time consuming. In all cases, we perform clustering into ten clusters (equal to the number of class labels). The algorithms are compared to kernel k-means with multiple restarts.

The performance of the above algorithms is primarily measured in terms of clustering error, but as the labels of the digits are available, we can also form the confusion matrix between clusters and classes and calculate normalized mutual information (NMI) [5]. This measure indicates how much the clustering and true class labels match and is defined as

$$\mathrm{NMI} = \frac{2\sum_{l=1}^{M}\sum_{h=1}^{c}\frac{n_l^h}{N}\log\frac{n_l^h N}{\sum_{i=1}^{M} n_i^h \sum_{i=1}^{c} n_l^i}}{\mathbb{H}(\pi)+\mathbb{H}(\zeta)} \tag{22}$$

where N is the data set size, M is the number of clusters, c is the number of classes, n_l^h is the number of points in cluster l from class h, $\mathbb{H}(\pi) = -\sum_{i=1}^{M}(n_i/N)\log(n_i/N)$ is the entropy of the clusters, and $\mathbb{H}(\zeta) = -\sum_{i=1}^{c}(n^i/N)\log(n^i/N)$ is the entropy of the classes. High NMI values indicate that the clustering solution and the partitioning based on the true class labels match well.

For the experiments, the same σ value is used for the Gaussian kernel in all algorithms in order to obtain a meaningful comparison. For the global kernel k-means with CMM algorithm, we set $\beta = \beta_0$ [see (11)], $s_{ij} = \exp(-\beta\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2)$, and select twice as many exemplars as the number of clusters ($P = 2M$), hence we select 20 exemplars. This algorithm, global kernel k-means, and fast global kernel k-means are run once. Kernel k-means is run 100 times and we report here the average and minimum performance values. The results are shown in Table III. Note that the NMI values of the "**Min**" row are those achieved by the restart with minimum clustering error.

When clustering the whole Pendigits data set, we observe that fast global kernel k-means, global kernel k-means with CMM, and the run of kernel k-means with minimum clustering error are evenly matched. The solution with minimum clustering error is identified in only seven out of 100 runs by kernel k-means. The average clustering error and NMI values are considerably worse though, as very bad solutions are identified during the restarts. When considering only the testing digits, kernel k-means achieves lower clustering error on its best run, but on average, it is worse than the other two algorithms. In more detail, 15 out of 100 runs achieve lower clustering error than fast global kernel k-means, but only three out of 100 achieve lower than global kernel k-means with CMM. As global kernel k-means with CMM has lower clustering error than fast global kernel k-means and is very close to the best run of kernel k-means, we can state that the exemplars identified are of good quality. As for the NMI values, they are similar for the three algorithms, despite the differences in clustering error, except for the average NMI of kernel k-means, which is quite lower due to bad solutions occurring during the restarts.

For the profile correlations case of the Multiple Features data set, global kernel k-means is the best in terms of both clustering error and NMI. Kernel k-means exhibits a similar behavior to the Pendigits data set when compared to the two variants, i.e., its best run is superior, but on average, it is worse. In particular, 40 out of 100 runs achieve lower clustering error than fast global kernel k-means, but only eight out of 100 are better than global kernel k-means with CMM. Note that global kernel k-means with CMM is very close to the original algorithm, whereas fast global kernel k-means is not. The NMI values are closely related to clustering error for this data set, i.e., lower clustering error results in higher NMI values. Clustering with the Zernike moments features leads to a similar behavior to profile correlations in terms of clustering error. The two variants are now closer to the best of kernel k-means as nine out of 100 runs and only one out of 100 are better than fast global kernel k-means and global kernel k-means with CMM, respectively. The NMI scores are low and do not follow the clustering error trend. Fast global kernel k-means is considerably better than the other methods which in turn are close to each other.

C. Olivetti Data Set

The Olivetti face database [22] contains ten 64 $\times$ 64 gray scale images of each of 40 individuals. We selected the first 100 images, belonging to ten individuals, and applied the same preprocessing as in [13], resulting in 900 images belonging to ten individuals (classes). We test the global kernel k-means, fast global kernel k-means, and global kernel k-means with CMM algorithms and compare them to kernel k-means with multiple restarts as well as affinity propagation [13]. Affinity propagation was tested on this data set in [13] and [23], achieving good results, and thus serves as a good measure for the performance of our methods. For our experiments, each image is represented with a vector containing the pixel intensities. Again the same σ is used for the Gaussian kernel for all algorithms. For the global kernel k-means with CMM algorithm, we set $\beta = \beta_0$ [see

TABLE IV
OLIVETTI RESULTS IN TERMS OF CLUSTERING ERROR

Method/Clusters $\sigma = 6.5$		10 Clusters	20 Clusters	30 Clusters	40 Clusters	50 Clusters	70 Clusters	100 Clusters	150 Clusters
Global Kernel *k*-Means		**220.76**	**192.31**	**174.3**	**161.23**	**150.39**	**134.54**	**117.54**	**96.94**
Global Kernel *k*-Means with CMM		221.43	192.91	175.42	162.27	152.67	137.75	120.32	101.26
Fast Global Kernel *k*-Means		222.6	195.81	178.57	165.11	155.47	139.14	122.99	104.96
Kernel *k*-Means (100 runs)	**Average**	223.43	196.52	179.77	167.89	158.35	143.33	127.15	107.19
	Min	221.4	193.78	177.58	165.03	154.74	140.44	124.39	104.45

TABLE V
OLIVETTI RESULTS IN TERMS OF MISCLASSIFICATION ERROR

Method/Clusters $\sigma = 6.5$		10 Clusters	20 Clusters	30 Clusters	40 Clusters	50 Clusters	70 Clusters	100 Clusters	150 Clusters
Global Kernel *k*-Means		56.78%	41.22%	32.89%	25.33%	**18.78%**	**13.89%**	12.22%	7.33%
Global Kernel *k*-Means with CMM		57.66%	45.89%	32.67%	**22.22%**	19.89%	18.33%	11.78%	9.11%
Fast Global Kernel *k*-Means		58.56%	47.67%	37.33%	28.78%	25.44%	17%	12.56%	9%
Kernel *k*-Means (100 runs)	**Average**	57.95%	42.2%	34.58%	29.6%	25.83%	19.84%	14.96%	10.14%
	Min	**55.67%**	**38.44%**	**31.44%**	28.67%	21%	19.56%	14.33%	6.89%
Affinity Propagation		60.22%	47.67%	38.56%	29%	23.11%	14.89%	**9.67%**	**5.11%**

(11)], $s_{ij} = \exp(-\beta||\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)||^2)$, and the number of selected exemplars equals three times the number of clusters ($P = 3M$). For affinity propagation, we set the preferences equal to a common value that yields the desired number of clusters and the pairwise similarities equal to $s(i,j) = -||\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)||^2$. Finally, kernel k-means is restarted 100 times.

We partition the data set in different number of clusters using all the above algorithms and for every experiment we report the clustering error and the *misclassification error*. For the misclassification error, each cluster is associated with the class (individual) whose images are the majority in that cluster. The images of the other classes are counted as misclassifications. Note that as the number of clusters increases the misclassification error naturally drops.

Table IV shows the clustering error achieved by the algorithms for different number of clusters. We do not report the clustering error for affinity propagation as it is naturally much higher relative to the other algorithms, since affinity propagation identifies exemplars and not centers as representatives of the clusters. As we can see, the performance of the algorithms is very similar for ten clusters, but as the number of clusters increases (the problem gets harder), the differences in performance become clear. Global kernel k-means is the best method achieving constantly the lowest clustering error. Global kernel k-means with CMM is the second best algorithm for 20 or more clusters. This supports our claim that good exemplars are identified and then fine tuned with global kernel k-means. Fast global kernel k-means is very close to the best run of kernel k-means, a little worse for 50 or less clusters and better for more. Finally, the average clustering error of kernel k-means is the highest in all cases, demonstrating that bad solutions are identified during the restarts.

In Table V, the algorithms are compared in terms of misclassification error. The results do not follow the clustering error trend, leading to the conclusion that lower clustering error does not necessarily imply and lower misclassification error. There are some similarities though, as kernel k-means gradually becomes the worst performer as the number of clusters increases, except for some fluctuations for 50 and 150 clusters. Also, global kernel k-means is among the best algorithms in all cases. Affinity propagation starts as the worst algorithm and ends as the best. In general, the two variants of global kernel k-means are inferior to the original algorithm, but both are better than the best run of kernel k-means for 70 and 100 clusters and global kernel k-means with CMM is also better for 40 and 50 clusters.

Overall, we can state that global kernel k-means is the best algorithm followed by the CMM variant, the fast global kernel k-means algorithm, and finally kernel k-means. This observation is primarily based on clustering error performance as this is the objective optimized by the algorithms. As for affinity propagation, its performance appears to be close to that of global kernel k-means when the number of clusters is large (70 or more), but worse for fewer clusters (Table V).

D. Graph Partitioning

As already discussed, the weighted versions of kernel k-means, global kernel k-means, and the two variants can be used to optimize many graph cut objectives. In our experiments, we test the weighted version of global kernel k-means and its two variants against weighted kernel k-means with multiple restarts, on the three graphs listed in Table VI that are available through the graph partitioning archive [24]. These graphs are undirected and all edge weights are equal to one, so the affinity matrix is symmetric and its entries are either zero or one. Each graph is partitioned into 32, 64, and 128 clusters and for each number of clusters we maximize the ratio association objective and we also minimize the normalized cut objective. The kernel and the weights are defined as discussed in Section VI. A problem that usually occurs is that the kernel matrix might not be positive semidefinite, thus the algorithms may not converge. For this reason, we diagonally shift the kernel matrix by a small amount as mentioned in Section VI. Note that we perform a diagonal shift that does not necessarily make the kernel matrix positive semidefinite, but is sufficient for the algorithms to converge. This is done in order to avoid adding a big diagonal shift which consequently will decrease

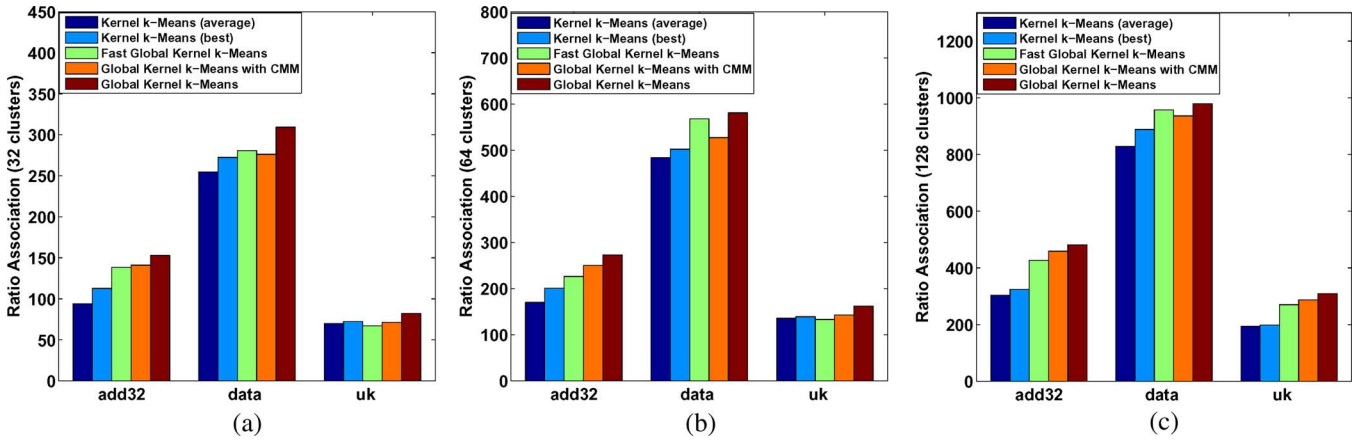

Fig. 3. Ratio association values achieved by global kernel k-means and its variants as well as by 50 restarts of kernel k-means: (a) 32 clusters; (b) 64 clusters; and (c) 128 clusters.

TABLE VI
TEST GRAPHS USED FOR GRAPH PARTITIONING

Graph Name	Number of Nodes	Number of Edges
add32	4960	9462
data	2851	15093
uk	4824	6837

the algorithms' performance as shown in [6]. For the weighted global kernel k-means with CMM algorithm, we set $\beta = \beta_0$ [see (19)], $s_{ij} = \exp(-\beta(K_{ii} + K_{jj} - 2K_{ij}))$, and select twice as many exemplars as the number of clusters ($P = 2M$). Finally, weighted kernel k-means is restarted 50 times.

Fig. 3 depicts the ratio association values achieved by the algorithms for 32, 64, and 128 clusters. For this task, we use the unweighted version of the algorithms since $w_i = 1$. Clearly global kernel k-means is the best algorithm as it achieves the highest ratio association for all graphs and all number of clusters. For the *add32* and *data* graphs, its two variants are also better than the best run of kernel k-means. Surprisingly, for the *data* graph, fast global kernel k-means is superior to global kernel k-means with CMM. For the *uk* graph and 32 clusters, the two variants are inferior to the best run of kernel k-means, but as the number of clusters increases and the problem gets harder, the global kernel k-means with CMM jumps ahead for 64 clusters and then fast global kernel k-means follows for 128 clusters. For this graph and 128 clusters, the ratio association achieved by the two variants is 38% higher than that of the best run of kernel k-means while for *add32* and 128 clusters it is 36% higher. This shows that the speeding-up schemes are considerably better than kernel k-means in many cases. Note that the two variants are very close to the original algorithm in some experiments, such as the *data* graph for 64 and 128 clusters where the original algorithm achieves only 2% higher ratio association than the best variant and the *add32* graph for 128 clusters where it achieves 5% higher ratio association than the best variant. This demonstrates that there are cases where the two speeding-up schemes lower the computational complexity without degrading the quality of the solution of the original algorithm. Between the two variants, there is no clear winner as global kernel k-means with CMM is better for the *add32* and *uk* graphs, while fast global kernel k-means is superior for the *data* graph.

Results on normalized cut values for 32, 64, and 128 clusters are shown in Fig. 4. For this task, we use the weighted version of the algorithms described in Section V. We do not set $\beta = \beta_0$ for the global kernel k-means with CMM algorithm as the β_0 value is huge, resulting in bad exemplars. Instead, we set $\beta = 10^{-3}\beta_0$, $\beta = 10^{-5}\beta_0$, and $\beta = 10^{-2}\beta_0$ for the *add32*, *data*, and *uk* graphs, respectively. Once again, global kernel k-means is the best performer for all graphs and all number of clusters as it achieves the lowest normalized cut, which in some cases is two to three times less than that of the second best algorithm. Global kernel k-means with CMM, although this time not close to the original algorithm, is clearly the second best method. In general, fast global kernel k-means has a similar performance to the best run of kernel k-means. There are cases where fast global kernel k-means is better, such as the *uk* graph for 32 and 128 clusters, and others where kernel k-means gets ahead, such as the *data* graph for 64 clusters and the *add32* graph for 128 clusters.

Overall, we conclude that global kernel k-means can be effectively applied to graph partitioning as it clearly outperforms kernel k-means with multiple restarts on this task and also is the best algorithm of those considered in all experiments. The two variants are very good alternatives to the original algorithm when the ratio association criterion is considered. For the normalized cut criterion, although global kernel k-means with CMM is inferior to the original algorithm, we may still consider its application in place of global kernel k-means, if lower computation time is important. Unfortunately, the same cannot be said for fast global kernel k-means as it is very close to kernel k-means with multiple restarts.

VIII. CONCLUSION

We have proposed the global kernel k-means clustering algorithm, a method that maps data points from input space to a higher dimensional feature space through the use of a kernel function and optimizes the clustering error in the feature space by locating near-optimal solutions. The main advantages of this method are its deterministic nature, which makes it independent of cluster initialization, and the ability to identify nonlinearly separable clusters in input space. Another important feature of the proposed algorithm is that in order to solve the M-clustering problem, all intermediate clustering problems with $1, \ldots, M$

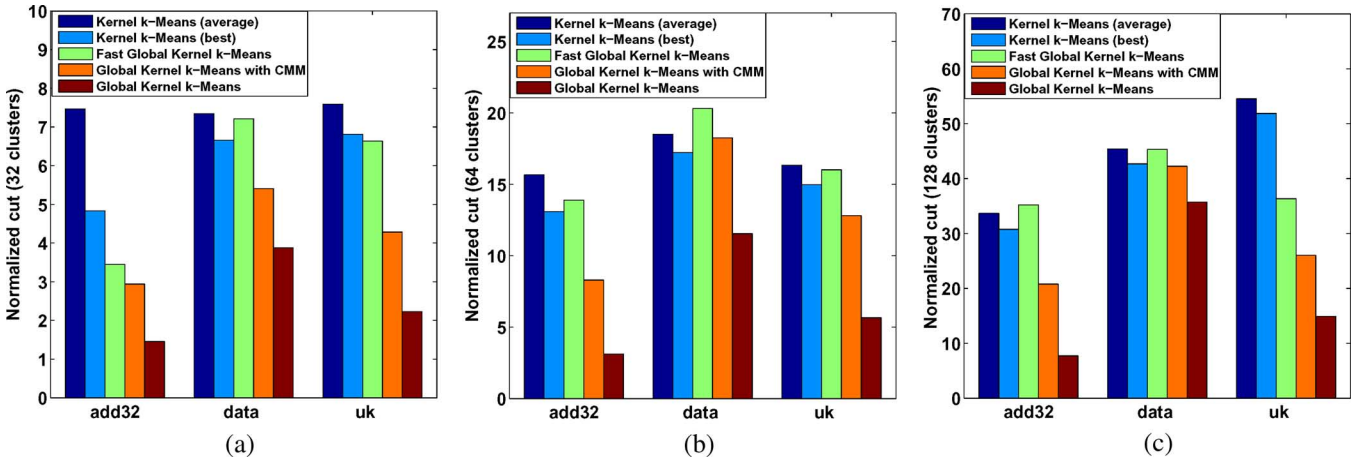

Fig. 4. Normalized cut values achieved by global kernel k-means and its variants as well as by 50 restarts of kernel k-means: (a) 32 clusters; (b) 64 clusters; (c) 128 clusters.

clusters are solved. This may prove useful in problems where we seek the true number of clusters.

Moreover, we developed two variants of the algorithm to accelerate its execution. The fast global kernel k-means variant considerably reduces the computational cost by requiring one run of kernel k-means for each intermediate clustering problem. The global kernel k-means with CMM variant first identifies a set of good exemplars, by fitting a convex mixture model to the data, and then tries only these exemplars as possible initializations for the newly added cluster in each of the intermediate problems. This variant has less computational savings, but it provides solutions closer to those of the original algorithm as shown by our experiments.

We also extended the above algorithms to handle weighted data points based on the weighted kernel k-means algorithm. The use of weights makes possible the application of these methods to graph partitioning, as their objective function can become equivalent to several graph cut criteria, if the weights and kernel matrix are set appropriately.

The aforementioned methods have been tested on several diverse data sets in order to ensure their broad applicability and draw reliable conclusions. In general, we could state that global kernel k-means and its two variants outperform kernel k-means with multiple restarts. Global kernel k-means, whenever applied, is the best performer. For the large Pendigits data set, where the computational cost makes the application of this algorithm time consuming, the two variants are good alternatives, particularly global kernel k-means with CMM. For the Olivetti data set, global kernel k-means is close to affinity propagation when the number of clusters is large and is clearly better for fewer clusters. As for the two variants, global kernel k-means with CMM is better. For the graph partitioning problem, where the weighted versions are used, the global kernel k-means algorithm is by far the best. The two variants are good alternatives when the ratio association is optimized, but for normalized cut only the global kernel k-means with CMM variant provides solutions similar to the original algorithm. Overall, we conclude that the exact global kernel k-means method is the best choice whenever the data set size allows its use. If the data set is large or time is a critical factor then the global kernel k-means with CMM variant could be applied and if further acceleration is required the fast global kernel k-means variant could be employed.

As for future work, a possible direction is the use of parallel processing to accelerate the global kernel k-means algorithm, since the local search performed when solving the k-clustering problem requires running kernel k-means N times and these executions are independent of each other. Another important issue is the development of theoretical results concerning the near-optimality of the obtained solutions. Also, we plan to use global kernel k-means in conjunction with criteria and techniques for estimating the optimal number of clusters and integrate it with other exemplar-based techniques. Finally, the application of this algorithm to graph partitioning needs further investigation and a comparison with spectral methods and other graph clustering techniques is required.

Appendix
Determining β_0 for Weighted Global Kernel k-Means With Convex Mixture Models

We follow the same approach as in [4] and reformulate our problem as a rate-distortion problem. By making use of [4, Proposition 1] and defining

$$Q'(j,\mathbf{x}) = q_j f_j\left(\phi(\mathbf{x})\right) = q_j C(\mathbf{x}) e^{-\beta d_\varphi(\phi(\mathbf{x}),\phi(\mathbf{x}_j))} \tag{23}$$

$$P'(j,\mathbf{x}) = \hat{P}(\mathbf{x})P'(j|\mathbf{x}) = \begin{cases} \dfrac{w_i}{\sum_{i'=1}^{N} w_{i'}} r_{ij}, & \mathbf{x} \in \mathcal{X} \\ 0, & \text{otherwise} \end{cases} \tag{24}$$

where $Q'(j,\mathbf{x})$ is the joint distribution of the convex mixture model, $\hat{P}(\mathbf{x})$ is the data set empirical distribution given by (15), $P'(j,\mathbf{x})$ is a distribution which has $\hat{P}(\mathbf{x})$ as its marginal, and $r_{ij} = P'(j|\mathbf{x}_i)$, the minimization of (16) becomes equivalent to minimizing the following objective:

$$D(P'\|Q') = \sum_{i,j=1}^{N} \hat{P}(\mathbf{x}_i) r_{ij} \left[\log \frac{r_{ij}}{q_j} + \beta d_\varphi\left(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\right)\right] + \text{const.} \tag{25}$$

Actually, since $P'(j,\mathbf{x}) \neq 0$ only for $\mathbf{x} \in \mathcal{X}$, the objective function is expressed only in terms of variables q_j and $P'(j|\mathbf{x}_i)$, therefore our goal is to minimize (25) in the space of distribu-

tions of random variable $(J, I) \in \{1, \ldots, N\} \times \{1, \ldots, N\}$, namely, in the product space of mixture model component indices $\times$ data point indices. For any set of values r_{ij}, setting $q_j = \sum_{i=1}^{N} \hat{P}(\mathbf{x}_i) r_{ij}$ minimizes (25). Substituting this on the above equation, we obtain

$$\begin{aligned} & D\left(P' \| Q'(P')\right) \\ & = \sum_{i,j=1}^{N} \hat{P}(\mathbf{x}_i) r_{ij} \left[\log \frac{r_{ij}}{\sum_{i'=1}^{N} \hat{P}(\mathbf{x}_{i'}) r_{i'j}} \right. \\ & \qquad \left. + \beta d_{\varphi}\left(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\right) \right] + \text{const.} \\ & = \mathbb{I}(J; I) + \beta \mathbb{E}_{J,I} d_{\varphi}\left(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\right) + \text{const.} \end{aligned} \tag{26}$$

where the first term is the mutual information under distribution $P'(j, \mathbf{x})$ and the second term is the expected value of the pairwise distances in feature space under the same distribution. Note that (26) is almost identical to [4, eq. (8)]. The only difference is on the empirical distribution of the data set.

If we interpret the above problem in the framework of rate distortion theory, we can think of r_{ij} as a probabilistic encoding, in feature space, of the data set onto itself with the corresponding average distortion $D = \mathbb{E}_{J,I} d_{\varphi}(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j))$ and the rate $\mathbb{I}(J; I)$. For the derivative of the rate-distortion function $R(D)$ [25], we have $\partial R / \partial D = -\beta$. In order to identify a good reference value for β, we follow the same approach as in [4] and define β_0 to be the slope of a line connecting the following two points on the rate distortion graph: the case of sending any point to itself

$$r_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

with zero distortion and rate equal to the entropy of the data set $R(0) = -\sum_{i=1}^{N} \hat{P}(\mathbf{x}_i) \log \hat{P}(\mathbf{x}_i)$ and the case of a random code $r_{ij} = 1/N$ with zero rate and average distortion $D = (1/N) \sum_{i,j=1}^{N} \hat{P}(\mathbf{x}_i) d_{\varphi}(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j))$. Thus, the empirical value of the β parameter for the weighted convex mixture model becomes

$$\beta_0 = N \frac{-\sum_{i=1}^{N} \hat{P}(\mathbf{x}_i) \log \hat{P}(\mathbf{x}_i)}{\sum_{i,j=1}^{N} \hat{P}(\mathbf{x}_i) d_{\varphi}\left(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\right)}$$

with $\hat{P}(\mathbf{x})$ given by (15).

REFERENCES

[1] R. Xu and D. Wunsch, II, "Survey of clustering algorithms," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 645–678, May 2005.

[2] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognit.*, vol. 36, no. 2, pp. 451–461, Feb. 2003.

[3] B. Scholkopf, A. J. Smola, and K.-R. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998.

[4] D. Lashkari and P. Golland, "Convex clustering with exemplar-based models," *Adv. Neural Inf. Process. Syst.*, vol. 20, pp. 825–832, 2008.

[5] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means, spectral clustering and normalized cuts," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Disc. Data Mining*, 2004, pp. 551–556.

[6] I. S. Dhillon, Y. Guan, and B. Kulis, "A unified view of kernel k-means, spectral clustering and graph cuts," Univ. Texas, Austin, TX, Tech. Rep. TR-04-25, 2004.

[7] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors: A multilevel approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 11, pp. 1944–1957, Nov. 2007.

[8] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta, "A survey of kernel and spectral methods for clustering," *Pattern Recognit.*, vol. 41, no. 1, pp. 176–190, Jan. 2008.

[9] G. Tzortzis and A. Likas, "The global kernel k-means clustering algorithm," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jun. 2008, pp. 1977–1984.

[10] A. M. Bagirov, "Modified global k-means algorithm for sum-of-squares clustering problems," *Pattern Recognit.*, vol. 41, no. 10, pp. 3192–3199, Oct. 2008.

[11] J. Li, D. Tao, W. Hu, and X. Li, "Kernel principle component analysis in pixels clustering," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell.*, Sep. 2005, pp. 786–789.

[12] R. Zhang and A. I. Rudnicky, "A large scale clustering scheme for kernel k-means," in *Proc. 16th Int. Conf. Pattern Recognit.*, 2002, pp. 289–292.

[13] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, pp. 972–976, 2007.

[14] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh, "Clustering with Bregman divergences," *J. Mach. Learn. Res.*, vol. 6, pp. 1705–1749, 2005.

[15] I. Csiszár and P. C. Shields, "Information theory and statistics: A tutorial," *Commun. Inf. Theory*, vol. 1, no. 4, pp. 417–528, 2004.

[16] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Adv. Neural Inf. Process. Syst.*, vol. 14, pp. 849–856, 2002.

[17] H. Zha, X. He, C. H. Q. Ding, M. Gu, and H. D. Simon, "Spectral relaxation for k-means clustering," *Adv. Neural Inf. Process. Syst.*, vol. 14, pp. 1057–1064, 2002.

[18] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[19] S. X. Yu and J. Shi, "Multiclass spectral clustering," in *Proc. 9th IEEE Int. Conf. Comput. Vis.*, 2003, pp. 313–319.

[20] M. Gong, L. Jiao, L. Bo, L. Wang, and X. Zhang, "Image texture classification using a manifold distance based evolutionary clustering method," *Opt. Eng.*, vol. 47, no. 7, p. 077201, Jul. 2008.

[21] A. Asuncion and D. Newman, UCI Machine Learning Repository, Univ. California at Irvine, Irvine, CA, 2007 [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[22] Olivetti and Oracle Research Laboratory, The Olivetti & Oracle Research Laboratory Database of Faces, Cambridge, U.K., 1994 [Online]. Available: http://mambo.ucse.edu/psl/olivetti.html

[23] D. Dueck and B. Frey, "Non-metric affinity propagation for unsupervised image categorization," in *Proc. 11th IEEE Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–8.

[24] C. Walshaw, The Graph Partitioning Archive, 2007 [Online]. Available: http://staffweb.cms.gre.ac.uk/~c.walshaw/partition/

[25] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

Grigorios F. Tzortzis received the B.Sc. and M.Sc. degrees in computer science from the University of Ioannina, Ioannina, Greece, in 2006 and 2008, respectively, where he is currently working towards the Ph.D. degree at the Department of Computer Science.

His research interests include machine learning, neural networks, multiview learning, and data mining.

Aristidis C. Likas (SM'03) received the Diploma degree in electrical engineering and the Ph.D. degree in electrical and computer engineering from the National Technical University of Athens, Athens, Greece, in 1990 and 1994, respectively.

Since 1996, he has been with the Department of Computer Science, University of Ioannina, Ioannina, Greece, where he is currently an Associate Professor. His research interests include neural networks, machine learning, statistical signal processing, and bioinformatics.