The Global Kernel *k*-Means Clustering Algorithm

Grigorios Tzortzis and Aristidis Likas

***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, in this work 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 a modification is proposed to reduce the computational cost that does not significantly affect the solution quality. We test the proposed methods on artificial data and also for the first time we employ kernel *k*-means for MRI segmentation along with a novel kernel. The proposed methods compare favorably to kernel *k*-means with random restarts.**

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, where homogeneous groups are identified by minimizing the clustering error defined as the sum of the squared Euclidean distances between each dataset 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 to different initial positions and thus 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 good so as to obtain a near optimal minimum. To deal with this problem the global *k*-means algorithm has been proposed [1], 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, …, M clusters using *k*-means. The solution with M clusters is built deterministically, so there is no dependency on initial conditions, and near optimal minima is found as shown in [1].

Kernel *k*-means [2] is an extension of the standard *k*-means algorithm that maps data points from input space to a higher dimensional feature space through a nonlinear transformation and minimizes the clustering error in feature space. Thus nonlinearly separated clusters are obtained in input space overcoming the second limitation of *k*-means. An important property of kernel *k*-means is its close relation to spectral clustering which is discussed further in Section II.B. The soft version of kernel *k*-means with a geodesic kernel is also available [3].

In this work 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. The algorithm works in an incremental fashion by solving all problems with 1, …, M clusters, using kernel *k*-means, in order to solve the M-clustering problem. The idea behind the proposed method is that a near optimal solution with M clusters can be obtained by starting with a near optimal solution with M-1 clusters and initializing the M-th cluster appropriately based on a local search. During the local search the M-th cluster is initialized several times (specifically N times where N is the size of the dataset) and the solution with the lowest clustering error is kept as the solution with M 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 and so 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. In order to lower the complexity a speeding up scheme is proposed, called fast global kernel *k*-means, which requires running kernel *k*-means only M times.

We present experimental results that compare global kernel *k*-means, its fast version and kernel *k*-means with multiple restarts on artificial data and on MRI segmentation. The results back our claim that the global kernel *k*-means algorithm locates near optimal solutions as it outperforms kernel *k*-means with multiple restarts in terms of clustering error. The fast version in some cases proves equal to the original algorithm and its clustering error is always lower compared to the average clustering error achieved by kernel *k*-means during the restarts.

In the following section we formally define clustering error and describe briefly the *k*-means, global *k*-means and kernel *k*-means algorithms. In Section III we present the proposed global kernel *k*-means algorithm along with an analysis of its computational complexity. The speeding up scheme is also described in this section. Our experimental evaluation is presented in Section IV. Finally Section V concludes this work.

Grigorios Tzortzis and Aristidis Likas are with the Department of Computer Science, University of Ioannina, GR 45110, Ioannina, Greece (email: {gtzortzi, arly}@cs.uoi.gr).

978-1-4244-1821-3/08/$25.00©2008 IEEE

II. PRELIMINARIES

A. k-Means and Global k-Means

Suppose we are given a dataset $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}$, $\boldsymbol{x}_n \in \mathbb{R}^d$ and we aim to partition this dataset into M disjoint clusters $C_1, C_2, \dots, C_M$. The *k*-means algorithm finds local optimal solutions with respect to the clustering error defined as the sum of squared Euclidean distances between each data point $\boldsymbol{x}_n$ and the cluster center $\boldsymbol{m}_k$ that $\boldsymbol{x}_n$ belongs to. Analytically the clustering error is given by:

$$E(\boldsymbol{m}_1, \dots, \boldsymbol{m}_M) = \sum_{i=1}^{N}\sum_{k=1}^{M} I(\boldsymbol{x}_i \in C_k)\|\boldsymbol{x}_i - \boldsymbol{m}_k\|^2 \qquad (1)$$

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

The two main disadvantages of the *k*-means algorithm are first the dependence of the final solution on the initial position of the cluster centers and second that clusters must be linearly separable. To deal with the initialization problem the global *k*-means algorithm has been proposed [1], an incremental-deterministic algorithm that employs the *k*-means algorithm 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. We begin by solving the 1-clustering problem using *k*-means. The optimal solution to this problem is known and the cluster center corresponds to the dataset centroid. Then we solve 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 position resulting from the 1-clustering problem and the other during the n-th run is initially placed at data point $\boldsymbol{x}_n$. 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 $(\boldsymbol{m}_1^*, \dots, \boldsymbol{m}_{k-1}^*)$ denote the solution to the *k*-1-clustering problem. We perform N executions of the *k*-means algorithm, with $(\boldsymbol{m}_1^*, \dots, \boldsymbol{m}_{k-1}^*, \boldsymbol{x}_n)$ as initial cluster centers for the n-th run, and keep the one 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 intermediate solutions with 1, …, M clusters when solving the *M*-clustering problem without additional cost. The experiments performed in [1] 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 execution two variants of the global *k*-means algorithm are proposed in [1] that do not considerably degrade the performance of the algorithm.

B. Kernel k-Means

Kernel *k*-means [2] 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 the feature space. This results in linear separators in feature space which correspond to nonlinear separators in input space. Thus kernel *k*-means avoids the problem of linearly separable clusters in input space that *k*-means suffers from.

TABLE I
EXAMPLES OF KERNEL FUNCTIONS

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

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

$$E(\boldsymbol{m}_1, \dots, \boldsymbol{m}_M) = \sum_{i=1}^{N}\sum_{k=1}^{M} I(\boldsymbol{x}_i \in C_k)\|\phi(\boldsymbol{x}_i) - \boldsymbol{m}_k\|^2 \qquad (2)$$

$$\text{where } \boldsymbol{m}_k = \frac{\sum_{i=1}^{N} I(\boldsymbol{x}_i \in C_k)\phi(\boldsymbol{x}_i)}{\sum_{i=1}^{N} I(\boldsymbol{x}_i \in C_k)}$$

$$\|\phi(\boldsymbol{x}_i) - \boldsymbol{m}_k\|^2 = K_{ii} - \frac{2\sum_{j=1}^{N} I(\boldsymbol{x}_j \in C_k)K_{ij}}{\sum_{j=1}^{N} I(\boldsymbol{x}_j \in C_k)} + \frac{\sum_{j=1}^{N}\sum_{l=1}^{N} I(\boldsymbol{x}_j \in C_k)I(\boldsymbol{x}_l \in C_k)K_{jl}}{\sum_{j=1}^{N}\sum_{l=1}^{N} I(\boldsymbol{x}_j \in C_k)I(\boldsymbol{x}_l \in C_k)} \qquad (3)$$

It must be noted that, by associating a weight with each data point, the weighted kernel *k*-means algorithm is derived [4] and it is proven that its objective function is equivalent to that of many graph partitioning problems such as ratio association, normalized cut etc if the weights and kernel are set appropriately [4] –[6]. Usually spectral methods, like the one in [7], are employed to solve these problems by calculating the eigenvectors of the kernel matrix. Spectral methods perform well because they compute globally optimal solutions of a *relaxation* of the problem considered. Calculating the eigenvectors of large matrices may prove problematic though. Kernel *k*-means avoids the need to calculate eigenvectors but cannot find the optimal solution because it depends on cluster initialization.

It has also been proved that performing *k*-means in the kernel pca space is equivalent to kernel *k*-means [8]. However this approach has two drawbacks: it is expensive since, like spectral clustering, it requires computation of the eigenvectors of the kernel matrix and it highly depends on the initialization of *k*-means.

Algorithm outline: Global kernel *k*-means

Input: Kernel matrix K, Total number of clusters M
Output: Final clustering of the points $C_1, C_2, \dots, C_M$
//There is no need to solve for 1 cluster as the solution is trivial and optimal. $C_1^* = X$

1. Solve all *k*–clustering problems for $k = 2$ to M
2. For each such problem run kernel *k*-means N times for $n = 1$ to N with input $(K,\ k,\ C_1^*, \dots, C_{k-1}^*, C_k = \{\boldsymbol{x}_n\})$ and output $(C_1^n, \dots, C_k^n, E_k^n)$
3. Find $E_k^* = \min_n (E_k^n)$ and set $C_1^*, \dots, C_k^*$ to the partitioning corresponding to E_k^* (this is the solution with k clusters).
4. Set $C_1 = C_1^*, \dots, C_M = C_M^*$ as output of the algorithm

Algorithm outline: Kernel *k*-Means

Input: Kernel matrix K, Number of clusters k, Initial clusters $C_1, \dots, C_k$
Output: Final clusters $C_1, \dots, C_k$, Clustering error E

1. For each point $\boldsymbol{x}_n$ and every cluster C_i compute $\|\phi(\boldsymbol{x}_n) - \boldsymbol{m}_i\|^2$ using (3)
2. Find $c^*(\boldsymbol{x}_n) = \text{argmin}_i(\|\phi(\boldsymbol{x}_n) - \boldsymbol{m}_i\|^2)$
3. Update clusters as $C_i = \{\boldsymbol{x}_n | c^*(\boldsymbol{x}_n) = i\}$
4. If not converged go to step 1 otherwise stop and return final clusters $C_1, \dots, C_k$ and E calculated using (2).

III. Global Kernel k-Means

In this paper we propose the *global kernel k-means* algorithm for minimizing the clustering error in feature space, defined in (2). Our method builds on the ideas of the global *k*-means and kernel *k*-means algorithms. Global kernel *k*-means maps the dataset points from input space to a higher dimensional feature space with the help of a kernel matrix $K \in \mathbb{R}^{N \times N}$ 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 center 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 *initializing the cluster centers* 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).

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 define 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 belong to the same cluster. We continue with the 2-clustering problem where kernel *k*-means is executed *N* times. During the *n*-th execution the initialization is done by considering two clusters one of which contains only $\boldsymbol{x}_n$. The solution with the lowest clustering error is kept as the solution with 2 clusters. In general for the *k*-clustering problem let $(C_1^*, \dots, C_{k-1}^*)$ denote the solution with k-1 clusters and assume that $\boldsymbol{x}_n \in C_l^*$. We perform N executions of the kernel *k*-means algorithm, with $(C_1^*, \dots, C_l = C_l^* - \{\boldsymbol{x}_n\}, \dots, C_{k-1}^*, C_k = \{\boldsymbol{x}_n\})$ as initial clusters for the *n*-th run, and keep the one resulting in the lowest clustering error. The above procedure is repeated until $k = M$.

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 starting from a state with *k*-1 near optimally defined clusters (solution of the *k*-1-clustering problem) and the *k*-th cluster initialized appropriately. It is quite reasonable to consider only one data point belonging to the *k*-th cluster when it is initialized as this is equivalent to initializing, during the *n*-th run, the *k*-th cluster center at point $\phi(\boldsymbol{x}_n)$ in feature space. Limiting the set of possible positions for the *k*-th center only to dataset 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 also solutions for every *k*-clustering problem with $k < M$ are obtained without additional cost which may be desirable in case we want to decide on the number of clusters for our problem.

A. Computational Complexity

Due to its close relation to global *k*-means and kernel *k*-means the global kernel *k*-means algorithm inherits their high computational cost. Given a kernel matrix the demanding step of the kernel *k*-means algorithm is the calculation of the distance between each point in feature space to every center, given by (3), in order to find the closest center. This is repeated for a number of iterations τ until convergence is achieved. As shown in [5] the complexity of kernel *k*-means is $O(N^2\tau)$ scalar operations. 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 an 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 was proposed in [9] which can be readily applied to our algorithm. As this is a very high complexity for large datasets a speeding up scheme is considered next.

B. 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 [1]. 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 *k*-th cluster is initialized to include the point $\boldsymbol{x}_n$ that *guarantees* the greatest reduction in clustering error. In more detail, we compute an *upper bound* $E_k^n \le E_{k-1}^* - b_k^n$ of the final clustering error when the *k*-th cluster is initialized to include point $\boldsymbol{x}_n$. E_{k-1}^* is the cluster-

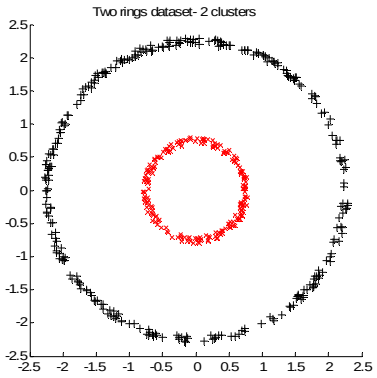

Fig. 1. Global kernel *k*-means, fast global kernel *k*-means and kernel *k*-means (run with minimum clustering error) on the two rings dataset.

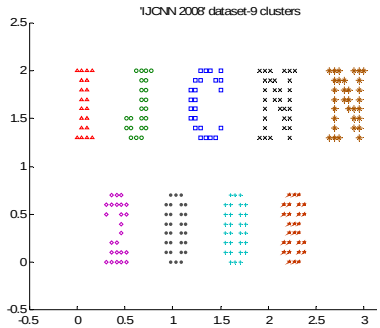

Fig. 2. Global kernel *k*-means, fast global kernel *k*-means and kernel *k*-means (run with minimum clustering error) on the 'IJCNN 2008' logo.

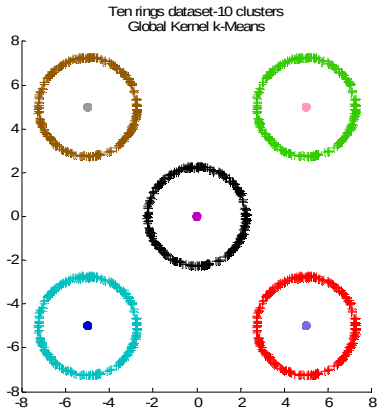

Fig. 3. Global kernel *k*-means on the ten rings dataset.

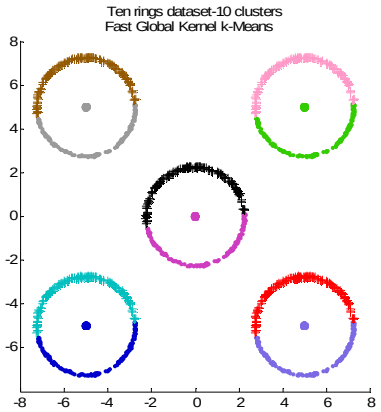

Fig. 4. Fast global kernel *k*-means on the ten rings dataset.

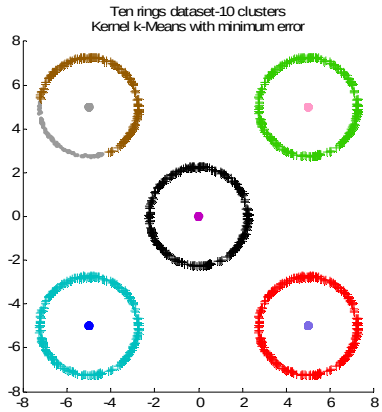

Fig. 5. Kernel k-means (the run with minimum clustering error) on the ten rings dataset.

ing 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) where d_{k-1}^i denotes the squared distance between $\boldsymbol{x}_i$ and its cluster center in feature space after solving the k-1-clustering problem. We select as initialization for the k-th cluster the point $\boldsymbol{x}_n$ that maximizes b_k^n:

$$b_k^n = \sum_{i=1}^{N} \max\left(d_{k-1}^i - \|\phi(\boldsymbol{x}_n) - \phi(\boldsymbol{x}_i)\|^2, 0\right) \quad (4)$$

where $\|\phi(\boldsymbol{x}_n) - \phi(\boldsymbol{x}_i)\|^2 = K_{nn} + K_{ii} - 2K_{ni}$

The correctness of the above upper bound is derived from the two following facts. First when the k-th cluster is initialized at point $\boldsymbol{x}_n$ it will allocate all points that are closer to $\boldsymbol{x}_n$ in feature space than to their cluster center in the solution with k-1 clusters (distance d_{k-1}^i). Quantity b_k^n measures the reduction in error due to this reallocation. Second, since kernel *k*-means monotonically converges as long as the kernel matrix is positive semi-definite, we are sure that the error will never exceed our bound.

When using this variant of the global kernel *k*-means algorithm in order 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 b_k^ns and this must be repeated M times in order to solve the problem with M clusters. Thus the overall cost incurred by the need to estimate the upper bound is $O(N^2M)$. Overall the fast global kernel *k*-means algorithm has $O\big(N^2(M\tau + d + M)\big) = O\big(N^2(M\tau + d)\big)$ 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 sufficiently small. This reduction in complexity comes at the cost of finding solutions with higher clustering error than the original algorithm in some cases. Our experiments indicate that the performance of the fast version is similar to that of global kernel *k*-means in many cases. This lower computational cost could make our algorithm a very good alternative to spectral methods for graph cut optimization, if weights are specified in the same way as in [4]-[6], as it also computes near optimal solutions.

IV. Experimental Evaluation

In this section we study the performance of global kernel *k*-means, its fast version and simple kernel *k*-means on a number of artificial datasets and also on segmentation of simulated MRI images. Our aim is to discover if indeed global kernel *k*-means avoids getting trapped in poor local minima and also how close the solutions of the fast global kernel *k*-means algorithm are to those of the original algorithm. The experiments were conducted on a computer run-

TABLE II
ARTIFICIAL DATASETS RESULTS IN TERMS OF CLUSTERING ERROR. THE KERNEL PARAMETER σ VALUE IS ALSO SHOWN

Method/Dataset		Two Rings $\sigma = 1$	Ten Rings $\sigma = 1.8$	'IJCNN 2008' $\sigma = 0.7$
Global kernel *k*-means		**320.17**	**966.87**	**27.97**
Fast global kernel *k*-means		**320.17**	1073.18	**27.97**
Kernel *k*-means (100 runs)	**Mean**	334.4	1107.97	37.72
	Std	6.4	177.24	6.16
	Min	**320.17**	981.53	**27.97**
	Max	351.05	1765.29	63.03

ning Windows XP with 3.5GB ram and 2.4GHz Intel Core 2 Duo processor. The code was implemented in MATLAB[1].

A. Artificial Datasets

We compared the three clustering algorithms mentioned above on three different artificial datasets. In all our experiments we used the Gaussian kernel, defined in Table I, which has a parameter σ whose value needs to be determined a priori. When comparing the algorithms the same σ was used for all three so as to obtain a meaningful comparison. The quality of the solutions produced by the three algorithms was evaluated in terms of the clustering error defined in equation (2). Given the value of σ, in order to measure the performance of the global kernel *k*-means and fast global kernel *k*-means algorithms we ran each algorithm once since they find deterministic solutions. For the kernel *k*-means algorithm we ran the algorithm 100 times, because the solution depends on the initialization of the clusters, and report the average clustering error, its standard deviation and minimum-maximum values during the 100 runs.

The first dataset (Figure 1) contains two rings with a total of 500 points. It is obvious that this problem is not linearly separable which makes *k*-means inappropriate for it. Running global kernel *k*-means and its fast version for $M = 2$ we manage to identify the two rings. Kernel *k*-means finds the two ring solution only in 12 out of 100 runs. In Table II we can see that the solution with two rings corresponds to the lowest clustering error. This solution is depicted in Figure 1.

The second dataset (Figure 3) 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 dataset has 5000 points and 10 clusters. For this difficult clustering problem, the global kernel *k*-means algorithm manages to find ten rings as shown in Figure 3 which is also the solution with the lowest clustering error in Table II. The fast global kernel *k*-means algorithm fails to identify the rings correctly: it splits the outer ring into two parts and merges one of them with the inner ring as shown in Figure 4. This problem is quite hard to solve and the greedy decisions that fast global kernel *k*-means makes result in suboptimal solutions with higher clustering error. Finally kernel *k*-means never identifies the ten rings solution during the 100 runs and also its average clustering error is higher than that of fast global kernel *k*-means. Figure 5 depicts the clustering result corresponding to the lowest clustering error during the 100 runs of kernel *k*-means.

The third dataset is the 'IJCNN 2008' logo and contains 9 clusters, one for each letter, with a total of 243 points. Global kernel *k*-means as well as its fast version split the logo correctly to its letters which is also the solution with lowest clustering error in Table II. Kernel *k*-means separates the logo to its letters only in 5 out of 100 runs. Figure 2 shows the clustering result with the lowest clustering error.

Despite the fact that kernel *k*-means was initialized 100 times for all datasets it never achieved a clustering error lower than that of global kernel *k*-means. This observation together with the fact that global kernel *k*-means correctly locates the rings in the second dataset, which is a difficult problem to solve, justify our statement that it finds near optimal solutions. Moreover the fast version of the algorithm in some cases, such as the first and third datasets, proves equal to the original algorithm but when the problem is hard, as with the second dataset, it is inferior to the original algorithm but still superior over kernel *k*-means as its error is lower than the average error of kernel *k*-means. Finally kernel *k*-means is very sensitive to the initialization of the clusters and finds from near optimal solutions to very bad ones during the 100 runs which make it difficult to decide on the number of restarts. Also we can never be sure if we managed to locate a near optimal solution during those repetitions.

B. MRI Segmentation

We have tested global kernel *k*-means, fast global kernel *k*-means and kernel *k*-means with multiple restarts on simulated MRI images downloaded from the *BrainWeb* site [10]. Specifically we used the normal brain database, which contains a single 3-d brain image, with the following simulation parameters: T1 modality, 1mm slice thickness, 3% noise and 20% intensity non-uniformity. The corresponding ground truth (discrete version) is also available and assigns to each voxel the label of the tissue that contributes most to that voxel. The tissues are divided in ten categories: background, cerebrospinal fluid (CSF), grey matter, white matter, muscle/skin, skin, skull, fat, glial matter and connective. For our experiments we focused on brain slices along the z-axis and in particular on those around the middle of the 3-d volume where the first seven tissue categories prevail, thus we considered clustering into *seven clusters*.

We segmented the slices 60, 80 and 100, shown in Figures 6-8, into seven clusters based on information derived from pixel intensities and considered only the pixels of the seven prevalent tissue categories. The ground truth for each of the above slices is depicted in Figures 9-11. Typical approaches consider only pixel intensities as features and employ *k*-means or Gaussian mixture models. In this work we suggest the use of kernel *k*-means for MRI segmentation. Moreover each pixel was represented with a vector containing not only the pixel intensity, but also the intensity histogram of a window around the pixel. The histogram was normalized so as bin quantities to represent probabilities. Based on this representation, we defined a kernel (5) that calculates the similari-

[1] For better understanding the experimental results please view the figures in color.

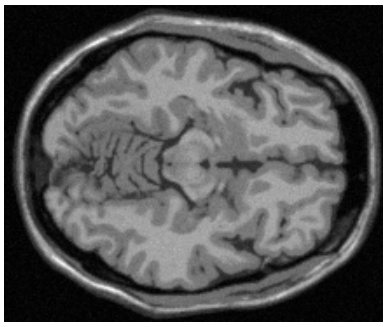

Fig. 6. MRI slice 60.

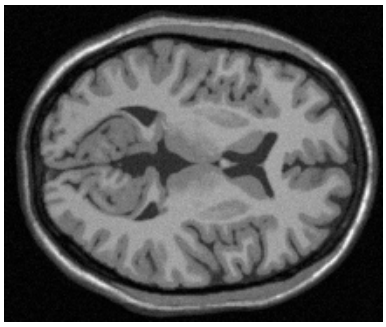

Fig. 7. MRI slice 80.

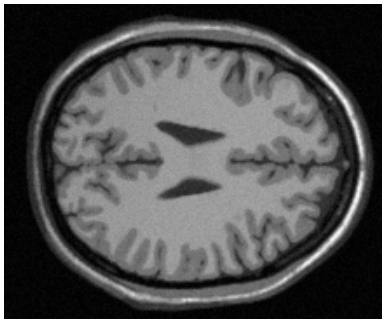

Fig. 8. MRI slice 100.

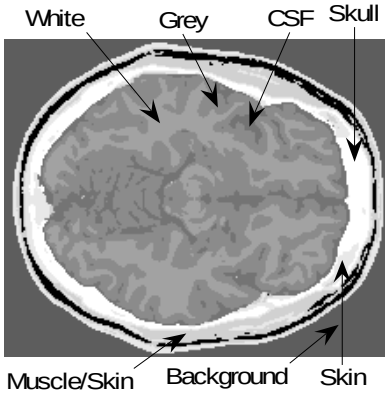

Fig. 9. Ground truth for slice 60. In black are the 3 tissues we ignore in our experiments.

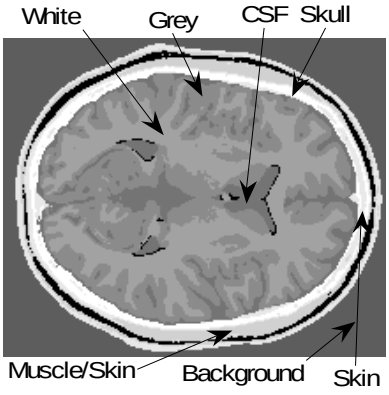

Fig. 10. Ground truth for slice 80. In black are the 3 tissues we ignore in our experiments.

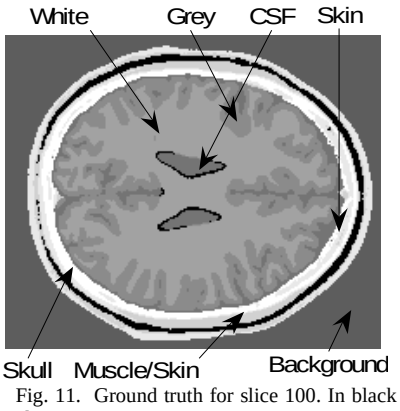

Fig. 11. Ground truth for slice 100. In black are the 3 tissues we ignore in our experiments.

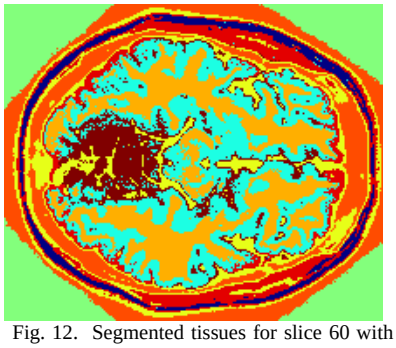

Fig. 12. Segmented tissues for slice 60 with fast global kernel *k*-means. In dark blue are the 3 tissues we ignore in our experiments.

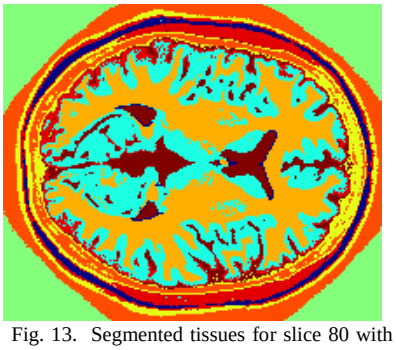

Fig. 13. Segmented tissues for slice 80 with fast global kernel *k*-means. In dark blue are the 3 tissues we ignore in our experiments.

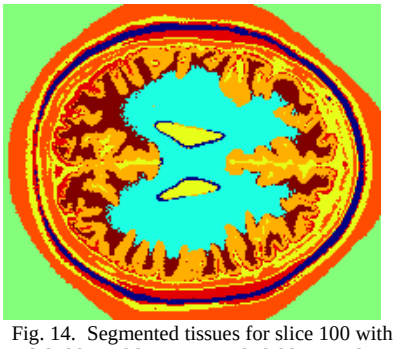

Fig. 14. Segmented tissues for slice 100 with fast global kernel *k*-means. In dark blue are the 3 tissues we ignore in our experiments.

ty between two pixels based on the similarity between their intensities and the similarity between the corresponding histograms. More specifically, if $I(i)$ is the intensity of pixel i and $P_z(i)$ is the probability associated with the z-th bin of pixel's i histogram, the kernel element K_{ij} is computed as:

$$K_{ij} = \exp\left(\frac{-\|I(i) - I(j)\|^2}{2\sigma^2}\right)\left[\sum_{z=1}^{Bins}\sqrt{P_z(i)P_z(j)}\right] \qquad (5)$$

The above kernel definition implies a positive semi-definite (i.e. valid) kernel matrix as it can be interpreted as a Gram matrix. The validity of the kernel is easily proven based on the property that if $\boldsymbol{x} = (\boldsymbol{x}_a, \boldsymbol{x}_b)$ and K_a, K_b are valid kernels then also $K(\boldsymbol{x}, \boldsymbol{x}') = K_a(\boldsymbol{x}_a, \boldsymbol{x}'_a)K_b(\boldsymbol{x}_b, \boldsymbol{x}'_b)$ is a valid kernel. In our case K_a is the Gaussian kernel among the intensities and K_b is simply the dot product between the square roots of the histogram vectors. Note that $K_{ii} = 1$ which is the maximum value for this kernel.

When comparing the clustering algorithms, the quality of the solution was once again measured in terms of clustering error. To decide on the kernel parameter values clustering error is not a valid measure though. Instead we define the *misclassification error* using the ground truth information. Specifically, each cluster on the final solution is assigned the label of the majority tissue class present on the cluster. The pixels of the other classes in the cluster are counted as misclassifications. Summing the misclassified pixels over all clusters and dividing by the total number of pixels gives the misclassification error. By trying different combinations for the σ value, window size and number of bins we decided to use $\sigma = 0.7$, a 31-by-31 window and split the intensity range of the whole slice into 70 equally spaced bins. The above parameters were discovered using slice 80 and were applied directly on the other two slices so as to see if they are effective in nearby slices. Note that tissue distribution changes across slices.

TABLE III

COMPARISON OF THE KERNEL IN (5) WITH $K_{ij} = \exp\left(\frac{-\|I(i)-I(j)\|^2}{2\sigma^2}\right)$ IN TERMS OF MISCLASSIFICATION ERROR USING FAST GLOBAL KERNEL *K*-MEANS

Kernel/Slice		**Slice 60**	**Slice 80**	**Slice 100**
Kernel given in (5) $\sigma = 0.7$ Win=31x31 Bins=70		**19.89%**	**14.1%**	**15.82%**
Gaussian kernel on pixel intensities	$\sigma = 0.2$	26.24%	22.37%	18.53%
	$\sigma = 0.4$	23.24%	22.11%	18.08%
	$\sigma = 0.7$	23.53%	22.22%	18.09%
	$\sigma = 1.0$	23.77%	22.27%	18.09%

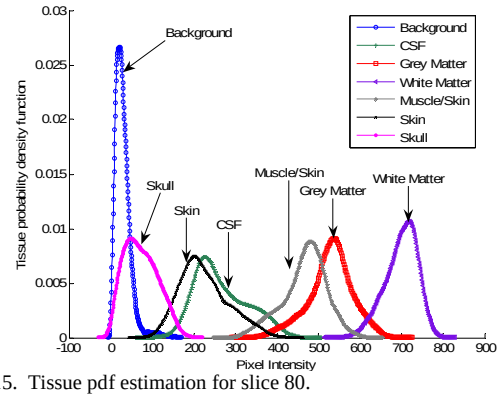

Fig. 15. Tissue pdf estimation for slice 80.

TABLE IV

CLUSTERING ERROR (CE) FOR FAST GLOBAL KERNEL *K*-MEANS AND KERNEL *K*-MEANS WITH MULTIPLE RESTARTS ON MRI SEGMENTATION. THE MISCLASSIFICATION ERROR (ME) FOR FAST GLOBAL KERNEL *K*-MEANS AND THE RUN OF KERNEL *K*-MEANS WITH MINIMUM CE IS ALSO SHOWN

Method/Slice $\sigma = 0.7$ Win=31x31 Bins=70		**Slice 60**		**Slice 80**		**Slice 100**	
		CE	ME	CE	ME	CE	ME
Fast global kernel *k*-means		5208.32	19.89%	5064.99	14.1%	5010.15	15.82%
Kernel *k*-means (100 runs)	**Mean**	5286.95	19.89%	5244.39	14.01%	5094.85	15.97%
	Std	66.29		127.63		141.7	
	Min	5207.65		5064.27		5009.75	
	Max	5364.68		5477.84		5808.77	

Our experimental evaluation focuses only on the comparison of fast global kernel *k*-means and kernel *k*-means with multiple restarts because global kernel *k*-means incurs a very high computational complexity, due to the large dataset size (each slice is of size 181-by-217), which makes its application impractical (requires a couple of days even when not considering all pixels as possible initializations). The global kernel *k*-means algorithm was run once, using the above parameter values, for slice 80 and the result was almost identical to that of the fast version. This is very encouraging because it seems that the fast version results equal those of the original algorithm for the MRI segmentation problem. The original algorithm was not further considered in our experiments. We also experimented with the simpler approach of using $K_{ij} = \exp\left(\frac{-\|I(i)-I(j)\|^2}{2\sigma^2}\right)$ as a kernel i.e. a Gaussian kernel on pixel intensities and no histogram. We compare the two approaches in Table III using fast global kernel *k*-means. It is evident that the histogram information does help to reduce the misclassification error. This backs our decision to adopt a more complex kernel.

To compare the two algorithms using the kernel given in (5) we ran fast global kernel *k*-means only once while kernel *k*-means was restarted 100 times for each slice and report here the average clustering error, its standard deviation and minimum-maximum values during the 100 runs. The results for the three slices are summarized in Table IV where also the misclassification error is shown. Note that for kernel *k*-means the misclassification error reported corresponds to the run with minimum clustering error. As we can see the best solutions identified by kernel *k*-means during the 100 runs are almost identical for slices 60, 80 and 100 to that of fast global kernel *k*-means. For slice 60, only 3 out of 100 runs are better for slice 80, 12 out of 100 runs and for slice 100, 28 out of 100 runs. The solution with minimum clustering error is found 1, 5 and 3 times for slices 60, 80 and 100 respectively. A more fair and correct comparison though is the one between the average clustering error during the 100 runs and the corresponding error of the fast global kernel *k*-means algorithm where clearly kernel *k*-means is outperformed for all slices. This is because during the restarts very bad solutions are also discovered. Moreover consider that executing those restarts is time consuming due to the large dataset size (requires in our implementation around 16 hours) while fast global kernel *k*-means is almost 21 times faster (takes around 45 minutes to run). The misclassification error for the two algorithms is identical for slice 60 while for slice 80 kernel *k*-means is slightly better and for slice 100 fast global kernel *k*-means is slightly better. The above suggest that choosing the fast global kernel *k*-means algorithm for MRI segmentation is a wiser solution. By examining the results in Table IV for slice 100 we observe that lower clustering error does not guarantee and lower misclassification error. We focus though primarily on clustering error when comparing the two algorithms as this is the quantity optimized by both algorithms.

Figures 12-14 depict the clustering result of fast global kernel *k*-means on the three slices. Since different tissues share the same pixel intensities (e.g. background and skull), as shown in Figure 15, it is difficult to place them on different clusters even when working with a kernel that takes advantage of the window information. For slice 60 we can see that grey and white matter form pure clusters. The background is split into two clusters and one of them is mixed with the skull while CSF is mixed with skin. For slice 80 grey matter, white matter, CSF and skin form quite pure

clusters. Again the background is split and the one part is mixed with the skull. Finally for slice 100 the result is similar to that of slice 60. Note that although some tissues are broken to two clusters, e.g. the white matter in slice 100, these clusters are pure something that is considered a good solution. Overall the clustering solution found by fast global kernel *k*-means is satisfactory especially for the inner part of the brain.

V. Conclusions

We have presented the global kernel *k*-means clustering algorithm, an algorithm 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 minima. 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, …, M clusters, are solved. This may prove useful in problems where we seek the actual number of clusters. Moreover we developed the fast global kernel *k*-means algorithm which considerably reduces the computational cost of the original algorithm without degrading significantly the quality of the solution.

We tested the above algorithms on artificial data and observed that they compare favorably to kernel *k*-means with multiple restarts. The original algorithm always found the best solution and in some cases the fast version proved equal. We proposed the use of kernel *k*-means for MRI segmentation and suggested the use of a *composite kernel* including not only pixel intensity but also local histogram information. For this task the fast global kernel *k*-means algorithm finds solutions that are almost equal to the best one identified by kernel *k*-means during the restarts and is a lot faster to execute. These facts make our method a viable clustering scheme that identifies near optimal solutions.

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 foundations behind the assumptions of the method. As already mentioned kernel *k*-means is closely related to spectral clustering. So extending the proposed algorithm by associating weights with each data point, following the ideas in [4]-[6], and using it to solve graph cut problems and comparing it to spectral methods is another possible research direction. Finally we plan to use the global kernel *k*-means in conjunction with criteria and techniques for estimating the optimal number of clusters.

Acknowledgment

This work has been supported by Interreg IIIA (Greece-Italy) grant I2101005.

References

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

[2] B. Scholkopf, A. Smola, and K. –R. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299-1319, July 1998.

[3] J. Kim, K. –H. Shim, and S. Choi, "Soft geodesic kernel *k*-means," in *Proc. 32nd IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 2007, pp. 429-432.

[4] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel *k*-means, spectral clustering and normalized cuts," in *Proc. 10th ACM Knowledge Discover and Data Mining Conf.*, 2004, pp. 551-556.

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

[6] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors: a multilevel approach," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 11, pp. 1944-1957, Nov. 2007.

[7] A. Y. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. of Neural Information Processing Systems*, 2001.

[8] J. Li, D. Tao, W. Hu, and X. Li, "Kernel principle component analysis in pixels clustering," in *Proc. Int. Conf. on Web Intelligence*, 2005, pp. 786-789.

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

[10] C. A. Cocosco, V. Kollokian, R. K. –S. Kwan, and A. C. Evans, "BrainWeb: Online interface to a 3d MRI simulated brain database," *NeuroImage*, vol.5, no.4, 1997.
Available: http://www.bic.mni.mcgill.ca/brainweb/