Kernel k-means, Spectral Clustering and Normalized Cuts

Inderjit S. Dhillon
Dept. of Computer Sciences
University of Texas at Austin
Austin, TX 78712
inderjit@cs.utexas.edu

Yuqiang Guan
Dept. of Computer Sciences
University of Texas at Austin
Austin, TX 78712
yguan@cs.utexas.edu

Brian Kulis
Dept. of Computer Sciences
University of Texas at Austin
Austin, TX 78712
kulis@cs.utexas.edu

ABSTRACT

Kernel k-means and spectral clustering have both been used to identify clusters that are non-linearly separable in input space. Despite significant research, these methods have remained only loosely related. In this paper, we give an explicit theoretical connection between them. We show the generality of the weighted kernel k-means objective function, and derive the spectral clustering objective of normalized cut as a special case. Given a positive definite similarity matrix, our results lead to a novel weighted kernel k-means algorithm that monotonically decreases the normalized cut. This has important implications: a) eigenvector-based algorithms, which can be computationally prohibitive, are not essential for minimizing normalized cuts, b) various techniques, such as local search and acceleration schemes, may be used to improve the quality as well as speed of kernel k-means. Finally, we present results on several interesting data sets, including diametrical clustering of large gene-expression matrices and a handwriting recognition data set.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Search and Retrieval; I.5.3 [**Pattern Recognition**]: Clustering

General Terms

Algorithms, Theory

Keywords

Spectral Clustering, Kernel k-means, Graph Partitioning

1. INTRODUCTION

Clustering has received a significant amount of attention in the last few years as one of the fundamental problems in data mining. k-means is one of the most popular clustering algorithms. Recent research has generalized the algorithm in many ways; for example, similar algorithms for clustering can be obtained using arbitrary Bregman divergences as the distortion measure [2]. Other advances include using local search to improve the clustering results [5] and using the triangle inequality to speed up the computation [4].

A major drawback to k-means is that it cannot separate clusters that are non-linearly separable in input space. Two recent approaches have emerged for tackling such a problem. One is kernel k-means, where, before clustering, points are mapped to a higher-dimensional feature space using a nonlinear function, and then kernel k-means partitions the points by linear separators in the new space. The other approach is spectral clustering algorithms, which use the eigenvectors of an affinity matrix to obtain a clustering of the data. A popular objective function used in spectral clustering is to minimize the normalized cut [12].

On the surface, kernel k-means and spectral clustering appear to be completely different approaches. In this paper we first unite these two forms of clustering under a single framework. By generalizing the k-means objective function to use both weights and kernels, we show how the two approaches to clustering are related. Specifically, we can rewrite the weighted kernel k-means objective function as a trace maximization problem whose relaxation can be solved with eigenvectors. The result shows how a particular kernel and weight scheme is connected to the spectral algorithm of Ng, Jordan, and Weiss [10]. However, the advantage to our approach is that we can generalize the clustering algorithm to use arbitrary kernels and weights.

Further, we show that by choosing the weights in particular ways, the weighted kernel k-means objective function is *identical* to the normalized cut. Thus far, only eigenvector-based algorithms have been employed to minimize normalized cuts in spectral clustering and image segmentation. However, software to compute eigenvectors of large sparse matrices (often based on the Lanczos algorithm) can have substantial computational overheads, especially when a large number of eigenvectors are to be computed. In such situations, our equivalence has an important implication: we can use k-means-like iterative algorithms for directly minimizing the normalized-cut of a graph.

We show the usefulness of our approach to the application of clustering gene expression data by applying a quadratic kernel (squared correlation) to obtain anti-correlated gene clusters and we illustrate the scalability of our algorithms in terms of computation time by applying it to a large handwriting recognition data set.

A word about notation. Capital letters such as A, X, Y

KDD'04, August 22–25, 2004, Seattle, Washinton, USA

Polynomial Kernel	$\kappa(\mathbf{a},\mathbf{b}) = (\mathbf{a}\cdot\mathbf{b}+c)^d$
Gaussian Kernel	$\kappa(\mathbf{a},\mathbf{b}) = \exp(-\|\mathbf{a}-\mathbf{b}\|^2/2\sigma^2)$
Sigmoid Kernel	$\kappa(\mathbf{a},\mathbf{b}) = \tanh(c(\mathbf{a}\cdot\mathbf{b})+\theta)$

Table 1: Examples of kernel functions

and Φ denote matrices; lower-case bold letters such as $\mathbf{a},\mathbf{b}$ denote column vectors; script letters such as $\mathcal{A},\mathcal{B},\mathcal{V},\mathcal{E}$ represent sets; $||\mathbf{a}||$ denotes the L^2 norm of a vector; and $||X||_F$ denotes the Frobenius norm of a matrix, and is given by $||X||_F = (\sum_{i,j} X_{ij}^2)^{1/2}$ $= (\mathrm{Tr}(X^TX))^{1/2}$

2. THE ESSENTIALS

In this section, we summarize the seemingly different approaches of weighted kernel k-means and spectral clustering.

2.1 Weighted Kernel k-means

The k-means clustering algorithm can be enhanced by the use of a kernel function; by using an appropriate nonlinear mapping from the original (input) space to a higher-dimensional feature space, one can extract clusters that are non-linearly separable in input space. Furthermore, we can generalize the kernel k-means algorithm by introducing a weight for each point $\mathbf{a}$, denoted by $w(\mathbf{a})$. As we shall see later, this generalization is powerful and encompasses the normalized cut of a graph.

Let us denote clusters by π_j, and a partitioning of points as $\{\pi_j\}_{j=1}^k$. Using the non-linear function ϕ, the objective function of weighted kernel k-means is defined as:

$$\mathcal{D}(\{\pi_j\}_{j=1}^k) = \sum_{j=1}^{k}\sum_{\mathbf{a}\in\pi_j} w(\mathbf{a})\|\phi(\mathbf{a})-\mathbf{m}_j\|^2 \qquad (1)$$

$$\text{where} \quad \mathbf{m}_j = \frac{\sum_{\mathbf{b}\in\pi_j} w(\mathbf{b})\phi(\mathbf{b})}{\sum_{\mathbf{b}\in\pi_j} w(\mathbf{b})}.$$

Note that $\mathbf{m}_j$ is the "best" cluster representative since

$$\mathbf{m}_j = \mathrm{argmin}_{\mathbf{z}} \sum_{\mathbf{a}\in\pi_j} w(\mathbf{a})\|\phi(\mathbf{a})-\mathbf{z}\|^2.$$

The Euclidean distance from $\phi(\mathbf{a})$ to center $\mathbf{m}_j$ is given by

$$\left\|\phi(\mathbf{a}) - \frac{\sum_{\mathbf{b}\in\pi_j} w(\mathbf{b})\phi(\mathbf{b})}{\sum_{\mathbf{b}\in\pi_j} w(\mathbf{b})}\right\|^2 = \phi(\mathbf{a})\cdot\phi(\mathbf{a}) - \frac{2\sum_{\mathbf{b}\in\pi_j} w(\mathbf{b})\phi(\mathbf{a})\cdot\phi(\mathbf{b})}{\sum_{\mathbf{b}\in\pi_j} w(\mathbf{b})} + \frac{\sum_{\mathbf{b},\mathbf{c}\in\pi_j} w(\mathbf{b})w(\mathbf{c})\phi(\mathbf{b})\cdot\phi(\mathbf{c})}{(\sum_{\mathbf{b}\in\pi_j} w(\mathbf{b}))^2}. \qquad (2)$$

The dot products $\phi(\mathbf{a})\cdot\phi(\mathbf{b})$ are computed using kernel function κ (see Table 1 for examples of popular kernel functions), and are contained in the kernel matrix K. All computation is in the form of such inner products, hence we can replace all inner products by entries of the kernel matrix.

The weighted kernel k-means algorithm (Algorithm 1) shares many properties of standard k-means; for example, the objective function value defined in (1) monotonically decreases with each iteration.

Assuming we are able to store the whole affinity matrix in main memory, we can analyze the time complexity of Algorithm 1. It is clear that the bottleneck is Step 3, i.e., the computation of distances. The first term in (2), $\phi(\mathbf{a})\cdot\phi(\mathbf{a})$, need not be computed since it is a constant for $\mathbf{a}$ and thus does not affect the assignment of $\mathbf{a}$ to clusters. The second term is calculated once per data point, and costs $O(n)$ each time it is computed, leading to a cost of $O(n^2)$ per iteration. For the third term, notice that $\frac{\sum_{\mathbf{b},\mathbf{c}\in\pi_j} w(\mathbf{b})w(\mathbf{c})\phi(\mathbf{b})\cdot\phi(\mathbf{c})}{(\sum_{\mathbf{b}\in\pi_j} w(\mathbf{b}))^2}$ is fixed for cluster j, so in each iteration it is computed once and stored. Thus the complexity is $O(n^2)$ scalar operations per iteration. Initially, we must compute the kernel matrix K, which usually takes time $O(n^2m)$, where m is the dimension of the original points. If the total number of iterations is τ, then the time complexity of Algorithm 1 is $O(n^2(\tau+m))$.

Algorithm 1: Weighted Kernel k-means.

```
WEIGHTED_KERNEL_KMEANS($K$, $k$, $w$, $C_1, ..., C_k$)
Input: $K$: kernel matrix, $k$: number of clusters, $w$: weights for each point
Output: $C_1, ...., C_k$: partitioning of the points
1. Initialize the $k$ clusters: $C_1^{(0)}, ..., C_k^{(0)}$.
2. Set $t = 0$.
3. For each point $\mathbf{a}$, find its new cluster index as
       $j^*(\mathbf{a}) = \mathrm{argmin}_j \|\phi(\mathbf{a}) - \mathbf{m}_j\|^2$, using (2).
4. Compute the updated clusters as
       $C_j^{t+1} = \{\mathbf{a} : j^*(\mathbf{a}) = j\}$.
5. If not converged, set $t = t + 1$ and go to Step 3; Otherwise, stop.
```

2.2 Spectral clustering

Spectral clustering has emerged recently as a popular clustering method that uses eigenvectors of a matrix derived from the data. Several algorithms have been proposed in the literature [9, 10, 12], each using the eigenvectors in slightly different ways. In this paper, we will focus on the normalized cut spectral algorithm.

2.2.1 Normalized Cuts

In [13], the authors consider the k-way *normalized cut* problem. We are given a graph $G = (\mathcal{V},\mathcal{E},A)$, where $\mathcal{V}$ is the set of vertices, $\mathcal{E}$ is the set of edges connecting vertices, and A is an edge affinity matrix, assumed to be nonnegative and symmetric. Suppose $\mathcal{A},\mathcal{B}\subseteq\mathcal{V}$, we define

$$\mathrm{links}(\mathcal{A},\mathcal{B}) = \sum_{i\in\mathcal{A},j\in\mathcal{B}} A(i,j).$$

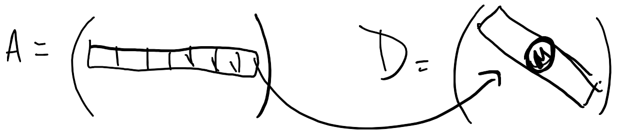
Then the normalized linkratio of $\mathcal{A},\mathcal{B}$ is:

$$\mathrm{normlinkratio}(\mathcal{A},\mathcal{B}) = \frac{\mathrm{links}(\mathcal{A},\mathcal{B})}{\mathrm{links}(\mathcal{A},\mathcal{V})}.$$

The k-way normalized cut problem is to minimize the links that escape a cluster relative to the total "weight" of the cluster. For a k-way partitioning of the vertices, we are interested in solving the following problem:

$$\text{minimize } \frac{1}{k}\sum_{j=1}^{k} \mathrm{normlinkratio}(\mathcal{V}_j, \mathcal{V}\setminus\mathcal{V}_j).$$

The authors of [13] obtain the following spectral relaxation to this problem. Let D be the diagonal matrix whose (i,i) entry is the sum of the entries of row i in matrix A. Then



check [13] for more details

the normalized cut criterion is equivalent to the following trace maximization problem:

$$\boxed{\text{maximize } \frac{1}{k}\text{trace}(Z^T AZ),}$$

where $Z = X(X^T DX)^{-1/2}$, and X is an $n \times k$ indicator matrix for the partitions. Note that $Z^T DZ = I_k$.

Letting $\tilde{Z} = D^{1/2}Z$ and relaxing the constraint that X is an indicator matrix results in the following problem: maximize the trace of $\tilde{Z}^T D^{-1/2}AD^{-1/2}\tilde{Z}$, where the constraints on $\tilde{Z}$ are relaxed such that $\tilde{Z}^T\tilde{Z} = I_k$. A well-known solution to this problem is obtained by setting the matrix $\tilde{Z}$ to be the top k eigenvectors of the matrix $D^{-1/2}AD^{-1/2}$. These eigenvectors are then used to compute a discrete partitioning of the points.

3. THE SPECTRAL CONNECTION

At first glance, weighted kernel k-means and normalized cuts using spectral clustering appear to be quite different. After all, spectral clustering uses eigenvectors to help determine the partitions, whereas eigenvectors do not appear to figure in kernel k-means. However, we saw that the normalized cut problem can be expressed as a trace maximization problem, and in this section, we show how we can express weighted kernel k-means as a trace maximization problem as well. This will show how to connect the two methods of clustering.

For ease in presentation, let us denote the "distortion" of a cluster π_j to be $d(\pi_j) = \sum_{\mathbf{a}\in\pi_j} w(\mathbf{a})\|\phi(\mathbf{a}) - \mathbf{m}_j\|^2$. Then we have that $\mathcal{D}(\{\pi_j\}_{j=1}^k) = \sum_{j=1}^k d(\pi_j)$. Moreover, let us denote, for a cluster π_j, the sum of the w weights of the points in π_j to be s_j; in other words, $s_j = \sum_{\mathbf{a}\in\pi_j} w(\mathbf{a})$. Finally, let us denote W to be the diagonal matrix of all the w weights, and W_j to be the diagonal matrix of the weights in π_j. Then we can rewrite the mean vector $\mathbf{m}_j$ as

$m_j = \sum_{a\in\pi_j} w(a)\phi(a)/s_j$

$$\mathbf{m}_j = \Phi_j \frac{W_j\mathbf{e}}{s_j},$$

where Φ_j is the matrix of points associated with cluster π_j (after the ϕ mapping), i.e., $\Phi = [\phi(\mathbf{a}_1), \phi(\mathbf{a}_2), \ldots, \phi(\mathbf{a}_n)]$, and $\mathbf{e}$ is the vector of all ones of appropriate size.

We can rewrite the distortion of cluster π_j to be:

$$\begin{aligned} d(\pi_j) &= \sum_{\mathbf{a}\in\pi_j} w(\mathbf{a})\|\phi(\mathbf{a}) - \mathbf{m}_j\|^2 \\ &= \sum_{\mathbf{a}\in\pi_j} w(\mathbf{a})\|\phi(\mathbf{a}) - \Phi_j\frac{W_j\mathbf{e}}{s_j}\|^2 \\ &= \|(\Phi_j - \Phi_j\frac{W_j\mathbf{e}\mathbf{e}^T}{s_j})W_j^{1/2}\|_F^2 \\ &= \|(\Phi_j W_j^{1/2}(I - \frac{W_j^{1/2}\mathbf{e}\mathbf{e}^T W_j^{1/2}}{s_j})\|_F^2. \end{aligned}$$

Using the fact that $\text{trace}(AA^T) = \text{trace}(A^T A) = \|A\|_F^2$, and noting that $I - \frac{W_j^{1/2}\mathbf{e}\mathbf{e}^T W_j^{1/2}}{s_j} = P$ is an orthogonal projection, i.e. $P^2 = P$ since $s_j = \mathbf{e}^T W_j\mathbf{e}$, we get that

$Z = W^{1/2}Y$. $Z^TZ = Y^TWY$

$Tr(Z^TKZ)$

$$\begin{aligned} d(\pi_j) &= \text{trace}\ \ \Phi_j W_j^{1/2}\left(I - \frac{W_j^{1/2}\mathbf{e}\mathbf{e}^T W_j^{1/2}}{s_j}\right)^2 W_j^{1/2}\Phi_j^T \\ &= \text{trace}\ \ \Phi_j W_j^{1/2}\left(I - \frac{W_j^{1/2}\mathbf{e}\mathbf{e}^T W_j^{1/2}}{s_j}\right) W_j^{1/2}\Phi_j^T \\ &= \text{trace}(W_j^{1/2}\Phi_j^T\Phi_j W_j^{1/2}) - \frac{\mathbf{e}^T W_j}{\sqrt{s_j}}\Phi_j^T\Phi_j\frac{W_j\mathbf{e}}{\sqrt{s_j}}. \end{aligned}$$

If we represent the full matrix of points as $\Phi = [\Phi_1, \Phi_2, \ldots, \Phi_k]$, then we have that

$$\mathcal{D}(\{\pi_j\}_{j=1}^k) = \text{trace}(W^{1/2}\Phi^T\Phi W^{1/2}) - \text{trace}(Y^T W^{1/2}\Phi^T\Phi W^{1/2}Y),$$

where

$$Y = \begin{bmatrix} \frac{W_1^{1/2}\mathbf{e}}{\sqrt{s_1}} & & & \\ & \frac{W_2^{1/2}\mathbf{e}}{\sqrt{s_2}} & & \\ & & \cdots & \\ & & & \frac{W_k^{1/2}\mathbf{e}}{\sqrt{s_k}} \end{bmatrix}.$$

Note that Y is an $n \times k$ *orthonormal* matrix, i.e., $Y^T Y = I$.

Since $\text{trace}(\Phi W\Phi^T)$ is a constant, the minimization of the objective function in (1) is equivalent to the maximization of $\text{trace}(Y^T W^{1/2}\Phi^T\Phi W^{1/2}Y)$. The matrix $\Phi^T\Phi$ is simply the kernel matrix K of the data, so we can rewrite it as the maximization of $\text{trace}(Y^T W^{1/2}KW^{1/2}Y)$.

A standard result in linear algebra [8] provides a global solution to a relaxed version of this problem. By allowing Y to be an arbitrary orthonormal matrix, we can obtain an optimal Y by taking the top k eigenvectors of $W^{1/2}KW^{1/2}$. Similarly, the sum of the top k eigenvalues of $W^{1/2}KW^{1/2}$ gives the optimal trace value.

4. IMPLICATIONS

The previous two sections show that the seemingly unrelated graph cut problem and weighted kernel k-means problem can both be written as trace maximization problems. This hints at a connection between these problems. We now make this connection precise, and discuss its implications.

4.1 Normalized Cuts using Weighted Kernel k-means

As discussed in Section 2.2.1, the normalized cut problem can be recast as a trace maximization problem, where we attempt to maximize $\text{trace}(\tilde{Z}^T D^{-1/2}AD^{-1/2}\tilde{Z})$. A simple calculation reveals that $\tilde{Z}$ is analogous to the matrix Y from the previous section.

We now show a direct relationship between the trace maximizations of the normalized cut and kernel k-means problems. Consider weighted kernel k-means with $W = D$ and $K = D^{-1}AD^{-1}$. The trace maximization of weighted kernel k-means is then $\text{trace}(Y^T D^{-1/2}AD^{-1/2}Y)$, which is *equivalent* to the trace maximization for normalized cut. If the affinity matrix K is positive definite, we can use the weighted kernel k-means procedure described in Algorithm 1 in order to minimize the normalized cut (positive definiteness allows us to factor K into $\Phi^T\Phi$, and allows us to prove convergence). Indeed, any starting partition can potentially be improved by Algorithm 1.

One advantage to our use of an iterative algorithm for these graph problems is that we can use different improvement methods, such as local search, to increase the quality of the results. In situations where eigenvector computation is difficult, for example, when the affinity matrix is large and sparse, and many eigenvectors are desired, our iterative algorithm is particularly useful.

4.2 Kernel k-means using Eigenvectors

The reformulation of the kernel k-means objective function allows us to solve a relaxed problem using the eigenvectors of the matrix $W^{1/2}KW^{1/2}$. This yields a spectral approach to minimizing the objective function: we first compute the top k eigenvectors of the matrix $W^{1/2}KW^{1/2}$. This maps the original points to a lower-dimensional space. Once postprocessing is performed and a discrete clustering solution has been attained, one can treat the resulting partitioning as a good initialization to kernel k-means on the full data set. This two-layer approach – first running spectral clustering to get an initial partitioning and then refining the partitioning by running kernel k-means on the partitioning – typically results in a robust partitioning of the data.

4.3 Interpreting NJW As Kernel k-means

The results from the previous sections give us novel ways to interpret the spectral algorithm of Ng, Jordan, and Weiss [10]. Their algorithm first computes the kernel matrix K, where the kernel that is used is the Gaussian Kernel. They compute a diagonal matrix D such that the diagonal entries of D are the sums of the rows of K. Then they compute the eigenvectors of the matrix $D^{-1/2}KD^{-1/2}$, and form a discrete clustering using these eigenvectors.

Hence, we see that the NJW algorithm can be viewed as either a spectral relaxation to the weighted kernel k-means objective function or as a normalized cut problem. The connection to normalized cuts is clear: we view the affinity matrix K in the spectral algorithm as defining the edge weights of a graph, and their algorithm attempts to minimize the normalized cut in this graph.

5. SCALABILITY ISSUES

In this section, we discuss methods for scaling the kernel k-means algorithm to large data sets.

To speed up the distance computation in our weighted kernel k-means algorithm, we can adapt the pruning procedure used in [4]. The idea behind the acceleration scheme is that we can use the triangle inequality to avoid unnecessary computation. We compute the distances between corresponding new and old centers, $\|\mathbf{m}_j^n - \mathbf{m}_j^o\|$ for all j, and store the information in a $k \times k$ matrix D. Similarly, we keep a $k \times n$ matrix L that contains *lower bound* for the distance from each point to each center. The distance from a point to its cluster center is exact in L. After the centers are updated, we estimate the lower bound from each point $\mathbf{a}$ to new cluster center, say $\mathbf{m}_j^n$, to be the difference between the lower bound from $\mathbf{a}$ to $\mathbf{m}_j^o$ and $\|\mathbf{m}_j^n - \mathbf{m}_j^o\|$. We actually compute the distance from $\mathbf{a}$ to $\mathbf{m}_j^n$ only if the estimation is smaller than distance from $\mathbf{a}$ to its cluster center. Figure 3 shows significant computation savings due to this estimation.

6. EXPERIMENTAL RESULTS

We now provide experimental results to validate the usefulness of the results presented in the previous sections. We first illustrate "diametric clustering" of genes with degree-2 polynomial kernel k-means. Then, with the handwriting recognition data set, we show that using eigenvectors to initialize kernel k-means gives better initial and final objective function values and better clustering results. Thus the theoretical connection between spectral clustering and kernel k-means helps in obtaining higher quality results. Finally, we show that our distance estimation techniques save a considerable amount of computation time, verifying the scalability of our approach.

6.1 Data sets

The human fibroblast gene expression has 517 genes across 12 conditions and the yeast dataset of Rosetta Inpharmatics has 5246 genes across 300 conditions. They are used and preprocessed as in [6].

The Pendigits is downloaded from UCI machine learning repository (ftp://ftp.ics.uci.edu/ pub/machine-learning-databases/pendigits), which contains (x, y) coordinates of hand-written digits. This dataset contains 7494 training digits and 3498 testing digits. Each digit is represented as a vector in 16-dimensional space.

6.2 Implementation details

Our kernel k-means algorithm is implemented in C++ and all experiments are done on a PC (Linux, two AMD 1.19GHz processors, 1GB main memory). In our implementation, we store the kernel matrix in main memory. All the plots are generated using Matlab.

6.3 Results

Diametrical Clustering of Gene Expression Data

In gene expression clustering, identifying anti-correlated relationship among genes is important, as it has been observed that genes whose expression patterns are strongly anti-correlated may also be functionally similar. We show that degree-2 polynomial kernel k-means can identify anti-correlated genes as was done in [6]. We cluster human fibroblast genes into 5 clusters. Then for each cluster $A_i, i = 1, ..., 5$, we compute the dot product of each gene vector and the leading eigenvector of $A_iA_i^T$ and plot genes across experiments in red or blue depending on whether the dot product value is positive or negative. The first 3 plots of Figure 1 show some sample results. Then we cluster 5246 yeast genes into 40 clusters. This took approximately 4.5 minutes, including forming the kernel matrix and clustering. The last 3 plots in Figure 1 correspond to one cluster of yeast genes. We magnify three parts of one cluster plot across 300 experiments in order to show the details. From the plots we see that degree-2 polynomial kernel k-means captures the anti-correlation similar to those captured by the diametric clustering algorithm. For example, cluster 2 of the human fibroblast dataset includes a number of genes involved in inter-cellular signaling, inflammation, angiogenesis and re-epithelialization, such as IL1beta, thrombomodulin, IL8, etc., corresponding to Figure 3*d* in [6].

Clustering Handwriting Recognition Data Set

Since the eigenvectors of the kernel matrix are the optimizers of a relaxed trace maximization problem, using the output of spectral clustering to initialize kernel k-means can often produce better clustering results than pure random initialization. To illustrate this, we run sigmoid kernel k-means on the Pendigits.tes dataset 10 times with random initialization, then average the initial and final objective function values. Then we run sigmoid kernel k-means 10 times again,

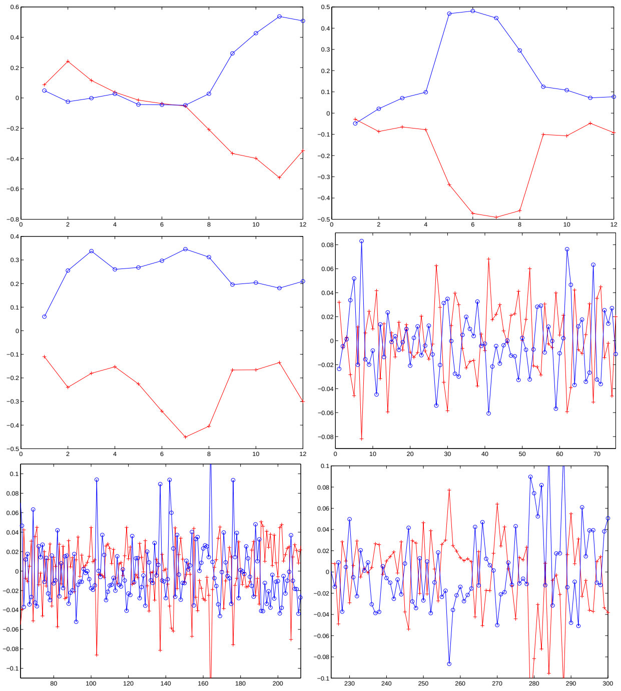


Figure 1: In each figure are plotted the mean expression profiles of two opposed clusters obtained on the human fibroblast dataset (first 3 plots) and the Rosetta dataset (last 3 plots). The clustering algorithm used is degree 2 polynomial kernel k-means.

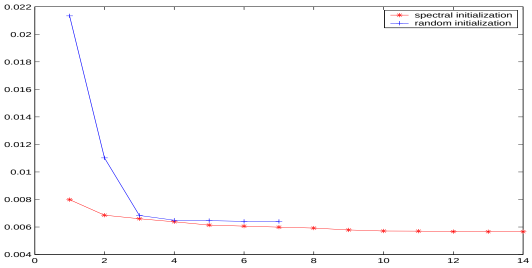


Figure 2: Two sample runs of sigmoid kernel k-means clustering of Pendigits.tes dataset; $a = 0.0045$ and $b = 0.11$ are used.

	initial	final	NMI
random	.0213	.0062	.666
spectral	.0081	.0059	.698

Table 2: Average initial and final objective function values and normalized mutual information values over 10 runs for the Pendigits.tes dataset.

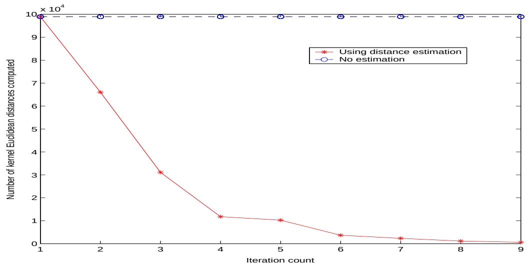


Figure 3: Computation savings due to triangle inequality estimation.

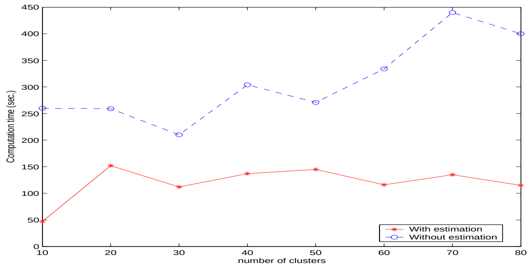


Figure 4: Computation time required to cluster the whole Pendigits dataset.

but initialized with the output of the spectral clustering algorithm. We also compute the average of the initial and final objective function values. For this dataset, since we know the underlying class labels we can evaluate clustering results by forming a confusion matrix, where entry (i, j), $n_i^{(j)}$ gives the number of points in cluster i and class j. From such a confusion matrix, we compute normalized mutual information (NMI) as

$$\frac{2\sum_{l=1}^{k}\sum_{h=1}^{c}\frac{n_l^{(h)}}{n}\log\frac{n_l^{(h)}n}{\sum_{i=1}^{k}n_i^{(h)}\sum_{i=1}^{c}n_l^{(i)}}}{H(\pi)+H(\zeta)},$$

where c is the number of classes, $H(\pi) = -\sum_{i=1}^{k}\frac{n_i}{n}\log\frac{n_i}{n}$, and $H(\zeta) = -\sum_{j=1}^{c}\frac{n^{(j)}}{n}\log\frac{n^{(j)}}{n}$. High NMI value indicates that the clustering and true class labels match well. In Table 2, we compare the average initial and final objective function values as well as the average normalized mutual information values of 10 runs each with random initialization and spectral initialization. Clearly, using spectral initialization can improve clustering quality. Figure 2 shows two sample runs of sigmoid kernel k-means clustering, one with random initialization and the other with spectral initialization.

The bottleneck in Algorithm 1 is the computation of Euclidean distances in kernel space. In order to avoid unnecessary computation, we incorporate the triangle inequality estimation mentioned in Section 5 into our kernel k-means software. Figure 3 shows the considerable savings in the number of Euclidean distance computations as the iteration count increases in a typical run of Algorithm 1 on the whole

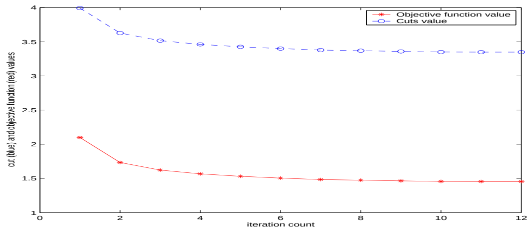


Figure 5: Objective function value of kernel k-means and normalized cut values monotonically decrease in Algorithm 1. Corresponding objective function value and cut value at each iteration differ by a constant.

Pendigits dataset, which contains 10992 digits. Without using the estimation, in every iteration nk distance calculations, 109920 in this case, need to be performed. However, after incorporating the estimation, we save considerable computation; for example, in the ninth iteration only 621 distances need to be computed. Figure 4 gives the computation time taken to cluster 10992 digits into a varied number of clusters. Times for both the original clustering algorithm and the one with distance estimation are shown. The distance estimation technique yields more savings in computation time as the number of clusters grows.

Figure 5 shows the objective function values of kernel k-means and the corresponding normalized cut values at each iteration of Algorithm 1 on the human fibroblast gene data. Again, a degree-2 polynomial kernel is used and 5 diametrical clusters are generated. We see that cut values decrease monotonically along with the objective function value of the kernel k-means algorithm. As can be seen, the difference between the corresponding cut value and objective function value at each iteration is a constant, which is equal to $k - \text{trace}(D^{-1}A)$.

7. RELATED WORK

Our work has been largely inspired by recent results on spectral clustering and relaxation methods presented in [10, 13, 1, 14]. In [14], the authors show that the traditional k-means objective function can be recast as a trace maximization problem of the Gram matrix for the original data. We generalize their work to the case when non-linear kernels are used, plus we treat the weighted version of the kernel k-means algorithm, which allows us to encompass spectral algorithms that use normalized cuts.

Although focusing on a different issue, [13, 1] also discusses a relation to normalized cuts, as well as a method for finding a good discrete clustering from the eigenvector matrix. In [1], they hint at a way to run an iterative algorithm for normalized cuts but their algorithm considers the factorization of a semi-definite matrix K such that $K = GG^T$, which takes $O(n^3)$ time, and thus is computationally worse than our kernel k-means approach.

The notion of using a kernel to enhance the k-means objective function was first described in [11]. Kernel-based learning methods have appeared in a number of other areas, especially in the area of Support Vector Machines [3].

In [7], the objective function was recast as a trace maximization, but they developed an EM-style algorithm to solve the kernel k-means problem.

8. CONCLUSION AND FUTURE WORK

In this paper, we have presented a theoretical connection between weighted kernel k-means and spectral clustering. We show that the weighted kernel k-means formulation is very general, and that the normalized cut objective can be recast as a special case of the weighted kernel k-means objective function. We also show that, given weights and a kernel matrix, it is possible to derive a spectral algorithm that solves a relaxed version of the objective function. We also provide new interpretations of the spectral algorithm of Ng, Jordan, and Weiss, while generalizing them to use other kernels, such as the degree-2 kernel for diametric clustering.

In future work, we would like to incorporate alternate objectives, such as ratio cut and ratio association, into our framework. So far, we have assumed that the affinity matrix is positive definite. In the future, we would like to be able to handle indefinite matrices.

Acknowledgments. This research was supported by NSF CAREER Award No. ACI-0093404, Texas Advanced Research Program grant 003658-0431-2001 and NSF ITR Award No. IIS-0325116.

9. REFERENCES

[1] F. Bach and M. Jordan. Learning spectral clustering. In *Proc. of NIPS-16*. MIT Press, 2004.

[2] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh. Clustering with Bregman divergence. *Proceeding of SIAM Data Mining conference*, pages 234–245, 2004.

[3] N. Cristianini and J. Shawe-Taylor. *Introduction to Support Vector Machines: And Other Kernel-Based Learning Methods*. Cambridge University Press, Cambridge, U.K., 2000.

[4] I. S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In *Data Mining for Scientific and Engineering Applications*, pages 357–381. Kluwer Academic Publishers, 2001.

[5] I. S. Dhillon, Y. Guan, and J. Kogan. Iterative clustering of high dimensional text data augmented by local search. In *Proceedings of The 2002 IEEE International Conference on Data Mining*, pages 131–138, 2002.

[6] I. S. Dhillon, E. M. Marcotte, and U. Roshan. Diametrical clustering for identifying anti-correlated gene clusters. *Bioinformatics*, 19(13):1612–1619, September 2003.

[7] M. Girolami. Mercer kernel based clustering in feature space. *IEEE Transactions on Neural Networks*, 13(4):669–688, 2002.

[8] G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1989.

[9] R. Kannan, S. Vempala, and A. Vetta. On clusterings – good, bad, and spectral. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, 2000.

[10] A. Y. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Proc. of NIPS-14*, 2001.

[11] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[12] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

[13] S. X. Yu and J. Shi. Multiclass spectral clustering. In *International Conference on Computer Vision*, 2003.

[14] H. Zha, C. Ding, M. Gu, X. He, and H. Simon. Spectral relaxation for k-means clustering. In *Neural Info. Processing Systems*, 2001.