Approximate Kernel *k*-means: Solution to Large Scale Kernel Clustering

Radha Chitta
chittara@msu.edu

Rong Jin
rongjin@cse.msu.edu

Timothy C. Havens
havenst@gmail.com

Anil K. Jain
jain@cse.msu.edu

Dept. of Computer Science and Engineering
Michigan State University
East Lansing, MI, 48824, USA

ABSTRACT

Digital data explosion mandates the development of scalable tools to organize the data in a meaningful and easily accessible form. Clustering is a commonly used tool for data organization. However, many clustering algorithms designed to handle large data sets assume linear separability of data and hence do not perform well on real world data sets. While kernel-based clustering algorithms can capture the non-linear structure in data, they do not scale well in terms of speed and memory requirements when the number of objects to be clustered exceeds tens of thousands. We propose an approximation scheme for kernel k-means, termed **approximate kernel *k*-means**, that reduces both the computational complexity and the memory requirements by employing a randomized approach. We show both analytically and empirically that the performance of approximate kernel k-means is similar to that of the kernel k-means algorithm, but with dramatically reduced run-time complexity and memory requirements.

Categories and Subject Descriptors

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

General Terms

Performance

Keywords

Large-scale clustering, Kernel clustering, k-means

1. INTRODUCTION

Advances in data collection and storage technologies over the past few years have resulted in data explosion in every scientific domain. A study by IDC and EMC Corp [1] predicted the creation of 35 trillion gigabytes of digital data by the year 2020. Much of this data (text, images and videos) is generated through online services like e-mail, social networks and blogs. Large-scale clustering is one of the principal tools to efficiently organize the massive amount of data and to enable convenient access by users. Clustering has found use in various applications such as web search, image retrieval, gene expression analysis, recommendation systems and market research based on transaction data [8, 24].

Most large-scale clustering techniques reported in the literature focus on grouping based on the Euclidean distance with the inherent assumption that all the data points lie in a Euclidean geometry. Kernel-based clustering methods overcome this limitation by embedding the data points into a high-dimensional non-linear manifold and defining their similarity using a nonlinear kernel distance function [27]. A number of kernel-based clustering methods such as spectral clustering [34, 41], non-negative matrix factorization (NMF) [44], kernel Self-Organizing Maps (SOM) [23, 32] and kernel neural gas [36] have been proposed. In this study, we focus on kernel k-means [17, 39, 40] due to its simplicity and efficiency. For instance, unlike spectral clustering, kernel k-means does not require the computation of the top eigenvectors of the kernel matrix, whose time complexity is cubic in the number of data points. In addition, several studies [12, 13, 45] have established the equivalence of kernel k-means and other kernel-based clustering methods, suggesting that most kernel-based clustering methods yield similar results.

Kernel k-means is a nonlinear extension of the classical k-means algorithm. It replaces the Euclidean distance function $d^2(x_a, x_b) = \|x_a - x_b\|^2$ employed in the k-means algorithm with a non-linear kernel distance function defined as

$$d^2_\kappa(x_a, x_b) = \kappa(x_a, x_a) + \kappa(x_b, x_b) - 2\kappa(x_a, x_b),$$

where $x_a \in \Re^d$ and $x_b \in \Re^d$ are two data points and $\kappa(.,.) : \Re^d \times \Re^d \rightarrow \Re$ is the kernel function. While the kernel distance function enables the clustering algorithm to capture the nonlinear structure in data, it requires computation and storage of an $n \times n$ kernel matrix in memory, rendering it non-scalable to large data sets. In this paper, we address the challenge posed by the large kernel matrix.

Note that the full kernel matrix is required in kernel k-means because the cluster centers are represented as a linear combination of *all* the data points, in accordance with the representer theorem [38]. In other words, the cluster centers lie in the subspace spanned by all the data points to be clustered. We can avoid computing the full kernel matrix by restricting the cluster centers to a smaller subspace. We randomly select a small number of data points, and ap-

KDD'11, August 21–24, 2011, San Diego, California, USA.

proximate the cluster centers using vectors in the subspace spanned by this subset. This approximation requires the computation and storage of only a small portion of the full kernel matrix, leading to a significant speedup of kernel k-means. We demonstrate, both theoretically and empirically, that the approximate kernel k-means yields similar clustering performance as the kernel k-means using the full kernel matrix.

The rest of the paper is organized as follows. Section 2 outlines the related work on large scale clustering and the kernel k-means algorithm. We present the approximate kernel k-means algorithm in Section 3 and its analysis in Section 4. Section 5 summarizes the results of our empirical studies, and Section 6 concludes this study with future work.

2. RELATED WORK

We first review the related work on large scale clustering and kernel based clustering, and then describe the kernel k-means algorithm.

2.1 Large scale clustering

A number of methods have been developed to efficiently cluster large data sets. Incremental clustering [5, 6] and divide-and-conquer based clustering algorithms [3, 18] were designed to operate in a single pass over the data points, thereby reducing the time required for clustering. Sampling based methods, such as CLARA [26] and CURE [19], reduce the computation time by finding the cluster centers based on a small number of randomly selected data points. The coreset algorithms [21] apply a similar idea except that the cluster centers are found based on a small set of representative, instead of randomly selected data points. Clustering algorithms, such as BIRCH [47], CLARANS [35], improve the clustering efficiency by summarizing the data set into data structures like trees and graphs for efficient data access.

With the evolution of cloud computing, parallel processing techniques for clustering [7, 9, 10, 15, 25, 33] are gaining popularity. These techniques speedup the clustering process by first dividing the task into a number of independent sub-tasks that can be performed simultaneously, and then efficiently merging these solutions into the final solution.

2.2 Kernel based clustering

Most of the existing methods for large scale clustering are based on the Euclidean distance, and are therefore, unable to deal with the data sets that are not linearly separable. Kernel based clustering techniques address this limitation by introducing a kernel distance function to capture the nonlinear structure in data [27]. Various kernel based clustering algorithms have been developed, including kernel k-means [39, 40], spectral clustering [34, 41], NMF [44], kernel SOM [23, 32] and kernel neural gas [36]. A key challenge faced by all the kernel-based algorithms is scalability as they require computing the full kernel matrix whose size is quadratic in the number of data points. Sampling methods, such as the Nystrom method [20, 43], have been employed to address this challenge [16, 31]. The key idea is to obtain a low rank approximation of the kernel matrix by sampling a number of columns of the kernel matrix and performing clustering using the approximate kernel matrix. A variety of sampling techniques are proposed in [4, 14, 28] and it has been shown that uniform sampling yields the best performance when compared to the other sampling techniques [28]. Finally, in [37], random projection is used along with the sampling method to further improve the clustering efficiency.

2.3 Kernel *k*-means

Let $X = \{x_1, x_2, ..., x_n\}$ be the input data set consisting of n data points, where $x_i \in \Re^d$, C be the number of clusters and $K \in \Re^{n\times n}$ be the kernel matrix with $K_{ij} = \kappa(x_i, x_j)$, where $\kappa(\cdot,\cdot)$ is the kernel function. Let $\mathcal{H}_\kappa$ be the Reproducing Kernel Hilbert Space (RKHS) endowed by the kernel function $\kappa(\cdot,\cdot)$, and $|\cdot|_{\mathcal{H}_\kappa}$ be the functional norm for $\mathcal{H}_\kappa$. The objective of kernel k-means is to minimize the *clustering error*, defined as the sum of squared distances between the data points and the center of the cluster to which the point is assigned. Hence, the kernel k-means problem can be cast as the following optimization problem:

$$\min_{U\in\mathcal{P}} \max_{\{c_k(\cdot)\in\mathcal{H}_\kappa\}_{k=1}^C} \sum_{k=1}^{C}\sum_{i=1}^{n} U_{ki}|c_k(\cdot) - \kappa(x_i,\cdot)|^2_{\mathcal{H}_\kappa}, \qquad (1)$$

where $U = (\mathbf{u}_1, \ldots, \mathbf{u}_C)^\top$ is the cluster membership matrix, $c_k(\cdot) \in \mathcal{H}_\kappa, k \in [C]$ are the cluster centers, and domain $\mathcal{P} = \{U \in \{0,1\}^{C\times n} : U^\top\mathbf{1} = \mathbf{1}\}$, where $\mathbf{1}$ is a vector of all ones. For convenience of presentation, we introduce two normalized versions of U. Let $n_k = \mathbf{u}_k^\top\mathbf{1}$ be the number of data points assigned to the kth cluster. We denote by

$$\begin{aligned}\widehat{U} &= (\widehat{\mathbf{u}}_1, \ldots, \widehat{\mathbf{u}}_C)^\top = [\mathrm{diag}(n_1, \ldots, n_C)]^{-1}U,\\ \widetilde{U} &= (\widetilde{\mathbf{u}}_1, \ldots, \widetilde{\mathbf{u}}_C)^\top = [\mathrm{diag}(\sqrt{n_1}, \ldots, \sqrt{n_C})]^{-1}U,\end{aligned}$$

the ℓ_1 and ℓ_2 normalized membership matrices, respectively.

It is easy to verify that given the cluster membership matrix U, the optimal solution for cluster centers is

$$c_k(\cdot) = \sum_{i=1}^{n} \widehat{U}_{ki}\kappa(x_i,\cdot), k \in [C]. \qquad (2)$$

As a result, we can formulate (1) as an optimization problem over U:

$$\min_U \mathrm{tr}(K) - \mathrm{tr}(\widetilde{U}K\widetilde{U}^\top). \qquad (3)$$

As indicated in (3), a naive implementation of kernel k-means requires computation and storage of the full $n \times n$ kernel matrix K, rendering it unsuitable for data sets with n greater than a few ten thousands. To the best of our knowledge, only a few attempts have been made to scale kernel k-means for large data sets. In [46], the memory requirement is reduced by dividing kernel matrix into blocks and using one block of the kernel matrix at a time. Although this technique handles the memory complexity, it still requires the computation of the full kernel matrix. The objective of our work is to reduce both the computational complexity and the memory requirements of kernel k-means.

3. APPROXIMATE KERNEL *K*-MEANS

A simple and naive approach for reducing the complexity of kernel k-means is to randomly sample m points from the data set to be clustered, and find the optimal cluster centers based only on the sampled points; then assign every unsampled data point to the cluster whose center is nearest. We refer to this 2-step process as the **two-step kernel *k*-means**, detailed in Algorithm 1. Though this approach has reduced run-time complexity and memory requirements, its

performance does not match that of the full kernel k-means, unless it is provided with a sufficiently large sample of data points.

We propose a superior approach for reducing the complexity of kernel k-means based on a simple but important observation. Kernel k-means requires the computation of the full kernel matrix K as the cluster centers $\{c_k(\cdot), k \in [C]\}$ are linear combinations of *all* the data points to be clustered, in accordance with (2). In other words, the cluster centers lie in the subspace spanned by all the data points, i.e., $c_k(\cdot) \in \mathcal{H}_a = \text{span}(\kappa(x_1,\cdot),\ldots,\kappa(x_n,\cdot)), k \in [C]$. We can avoid computing the full kernel matrix if we restrict the solution for the cluster centers to a smaller subspace $\mathcal{H}_b \subset \mathcal{H}_a$. $\mathcal{H}_b$ should be constructed such that (i) $\mathcal{H}_b$ is small enough to allow efficient computation, and (ii) $\mathcal{H}_b$ is rich enough to yield similar clustering results as those using $\mathcal{H}_a$. We employ a simple randomized approach for constructing $\mathcal{H}_b$: we randomly sample m data points ($m \ll n$), denoted by $\widehat{X} = \{\widehat{x}_1,\ldots,\widehat{x}_m\}$, and construct the subspace $\mathcal{H}_b = \text{span}(\widehat{x}_1,\ldots,\widehat{x}_m)$. Given the subspace $\mathcal{H}_b$, we modify (1) as

$$\min_{U\in\mathcal{P}} \max_{\{c_k(\cdot)\in\mathcal{H}_b\}_{k=1}^C} \sum_{k=1}^{C}\sum_{i=1}^{n} U_{ki}|c_k(\cdot)-\kappa(x_i,\cdot)|^2_{\mathcal{H}_\kappa}. \quad (4)$$

Let $K_B \in \Re^{n\times m}$ represent the kernel similarity matrix between data points in X and the sampled data points $\widehat{X}$, and $\widehat{K} \in \Re^{m\times m}$ represent the kernel similarity between the sampled data points. The following lemma allows us to reduce (4) to an optimization problem involving only the cluster membership matrix U.

LEMMA 1. *Given the cluster membership matrix U, the optimal cluster centers to (4) are given by*

$$c_k(\cdot) = \sum_{i=1}^{m} \alpha_{ki}\kappa(\widehat{x}_i,\cdot), \quad (5)$$

where $\alpha = \widehat{U}K_B\widehat{K}^{-1}$. The optimization problem for U is given by

$$\min_U tr(K) - tr(\widetilde{U}K_B\widehat{K}^{-1}K_B^\top\widetilde{U}^\top). \quad (6)$$

PROOF. Let $\varphi_i = (\kappa(x_i,\widehat{x}_1),\ldots,\kappa(x_i,\widehat{x}_m))$ and $\alpha_i = (\alpha_{i1},\ldots,\alpha_{im})$ be the i-th rows of matrices K_B and α respectively. As $c_k(\cdot) \in \mathcal{H}_b = \text{span}(\widehat{x}_1,\ldots,\widehat{x}_m)$, we can write $c_k(\cdot)$ as

$$c_k(\cdot) = \sum_{i=1}^{m} \alpha_{ki}\kappa(\widehat{x}_i,\cdot).$$

and write the objective function in (4) as

$$\begin{aligned}&\sum_{k=1}^{C}\sum_{i=1}^{n} U_{ki}|c_k(\cdot)-\kappa(x_i,\cdot)|^2_{\mathcal{H}_\kappa}\\ =\ & \text{tr}(K) + \sum_{k=1}^{C}\left(n_k\alpha_k^\top\widehat{K}\alpha_k - 2\mathbf{u}_k^\top K_B\alpha_k\right). \quad (7)\end{aligned}$$

By minimizing over α_k, we have

$$\alpha_k = \widehat{K}^{-1}K_B^\top\widehat{\mathbf{u}}_k, k \in [C]$$

and therefore, $\alpha = \widehat{U}K_B\widehat{K}^{-1}$. We complete the proof by substituting the expression for α into (7). □

As indicated by Lemma 1, we need to compute only K_B [1] for finding the cluster memberships. When $m \ll n$, this computational cost would be significantly smaller than that of computing the full matrix. Note that the problem in (6) can also be viewed as approximating the kernel matrix K in (4) by $K_B\widehat{K}^{-1}K_B^\top$, which is equivalent to the Nystrom method for low rank matrix approximation. We refer to the proposed algorithm as **Approximate Kernel *k*-means**, outlined in Algorithm 2. Figure 1 illustrates and compares this algorithm with the two-step kernel k-means on a 2-dimensional synthetic data set.

Algorithm 1 Two-step Kernel k-means

1: **Input**:
- $X = (x_1,\ldots,x_n)$: the set of n data points to be clustered
- $\kappa(\cdot,\cdot) : \Re^d \times \Re^d \mapsto \Re$: kernel function
- m: the number of randomly sampled data points ($m \ll n$)
- C: the number of clusters

2: Randomly select m data points from X, denoted by $\widehat{X} = (\widehat{x}_1,\ldots,\widehat{x}_m)$.
3: *// Step 1*
4: Compute the cluster centers, denoted by $c_k(\cdot), k \in [C]$, by applying the standard kernel k-means to $\widehat{X}$.
5: *// Step 2*
6: **for** $i = 1,\ldots,n$ **do**
7: Find the closest cluster center k_* for x_i by

$$k_* = \arg\min_{k\in[C]} |c_k(\cdot)-\kappa(x_i,\cdot)|_{\mathcal{H}_\kappa}$$

8: Update the ith column of U by $U_{ki} = 1$ for $k = k_*$ and zero otherwise.
9: **end for**

4. ANALYSIS

In this section, we first analyze the computational complexity of the proposed approximate kernel k-means algorithm, and then examine the quality of the data partitions generated by the proposed algorithm.

4.1 Computational Complexity

The most expensive operation in the proposed algorithm is the matrix inversion $\widehat{K}^{-1}$ and calculation of $T = K_B\widehat{K}^{-1}$, which has a computational cost of $O(m^3 + m^2n)$. The computational cost of computing α and updating the membership matrix U is $O(mnCl)$, where l is the number of iterations needed for convergence. Hence, the overall computational cost is $O(m^3 + m^2n + mnCl)$. We can further reduce the computational complexity by avoiding the matrix inversion $\widehat{K}^{-1}$ and formulating the calculation of $\alpha = \widehat{U}T = \widehat{U}K_B\widehat{K}^{-1}$ as the following optimization problem:

$$\min_{\alpha\in\Re^{C\times m}} \frac{1}{2}\text{tr}(\alpha\widehat{K}\alpha) - \text{tr}(\widehat{U}K_B\alpha^\top) \quad (8)$$

If $\widehat{K}$ is well conditioned (i.e., the minimum eigen value of $\widehat{K}$ is significantly larger than zero), we can solve the optimization problem in (8) by a simple gradient descent method

[1]Note that $\widehat{K}$ is part of K_B and therefore does not need to be computed separately.

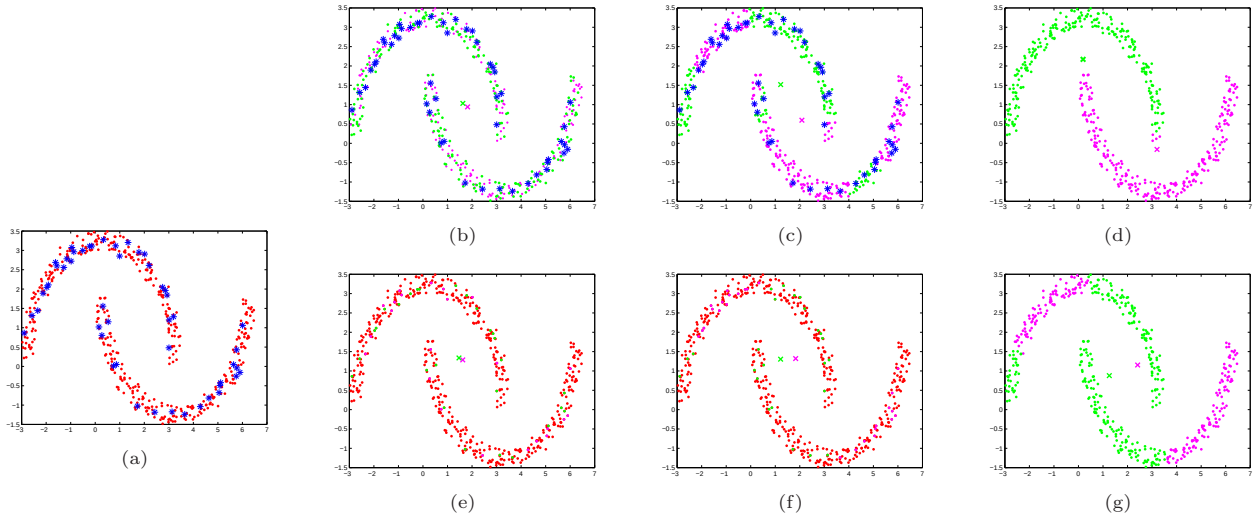


Figure 1: Illustration of the approximate kernel k-means algorithm and the two-step kernel k-means algorithm on a 2-D data set. Figure (a) shows all the data points (in red) and the sampled points (in blue). Figures (b)-(d) and (e)-(g) show the process of discovery of the two clusters in the data set and their centers (represented by x) by the proposed algorithm and the two-step algorithm, respectively. The approximate kernel k-means algorithm assigns labels to all the data points during each iteration while constraining the centers to the subspace spanned by the sampled points. The two-step algorithm clusters only the sampled points and assigns labels to the remaining points in the final phase.

Algorithm 2 Approximate Kernel k-means

Input:
- $X = (x_1, \ldots, x_n)$: the set of n data points to be clustered
- $\kappa(\cdot,\cdot) : \Re^d \times \Re^d \mapsto \Re$: kernel function
- m: the number of randomly sampled data points ($m \ll n$)
- C: the number of clusters
- $MAXITER$: maximum number of iterations

Randomly sample m data points from X, denoted by $\widehat{X} = (\widehat{x}_1, \ldots, \widehat{x}_m)$.
Compute $K_B = [\kappa(x_i, \widehat{x}_j)]_{n\times m}$ and $\widehat{K} = [\kappa(\widehat{x}_i, \widehat{x}_j)]_{m\times m}$.
Compute $T = K_B \widehat{K}^{-1}$.
Randomly initialize the membership matrix U.
Set $t = 0$.
repeat
  Set $t = t + 1$.
  Compute the ℓ_1 normalized membership matrix $\widehat{U}$ by $\widehat{U} = [\mathrm{diag}(U\mathbf{1})]^{-1}U$.
  Calculate $\alpha = \widehat{U}T$.
  for $i = 1, \ldots, n$ **do**
    Find the closest cluster center k_* for x_i by

$$k_* = \arg\min_{k\in[C]} \alpha_k^\top \widehat{K}\alpha_k - 2\varphi_k^\top \alpha_k$$

    where α_k and φ_k are the kth rows of matrix α and K_B, respectively.
    Update the ith column of U by $U_{ki} = 1$ for $k = k_*$ and zero otherwise.
  end for
until the membership matrix U does not change or $t > MAXITER$

with a convergence rate of $O(\log(1/\varepsilon))$, where ε is the desired accuracy. As the computational cost of each step in the gradient descent method is $O(m^2C)$, the overall computational cost is only $O(m^2Cl\log(1/\varepsilon)) \ll O(m^3)$ when $Cl \ll m$. Using this trick, we reduce the overall computational cost to $O(m^2Cl + mnCl + m^2n)$. As the largest matrix that needs to be stored in memory in Algorithm 2 is K_B, the memory requirement is only $O(mn)$. This is a dramatic decrease in the run-time and memory requirements for large data sets when compared to the $O(n^2)$ complexity of classical kernel k-means.

4.2 Error Bound

In this section, we compare the clustering error of our algorithm with that of kernel k-means run using the full kernel. As the proposed algorithm differs from the standard kernel k-means algorithm only by the restriction of the cluster centers to a smaller subspace $\mathcal{H}_b$, constructed using the sampled data points, our analysis will be focused on bounding the expected error due to this constraint.

We introduce binary random variables $\xi = (\xi_1, \xi_2, ..., \xi_n)^\top \in \{0,1\}^n$ to represent the random sampling process, where $\xi_i = 1$ if x_i is selected for constructing the subspace and zero otherwise. The following proposition allows us to write the clustering error in terms of random variable ξ.

PROPOSITION 1. *Given the cluster membership matrix* $U = (\mathbf{u}_1, \ldots, \mathbf{u}_C)^\top$, *the clustering error can be expressed in* ξ *as*

$$\mathcal{L}(U,\xi) = tr(K) + \sum_{k=1}^{C} \mathcal{L}_k(U,\xi), \tag{9}$$

where $\mathcal{L}_k(U,\xi)$ *is*

$$\mathcal{L}_k(U,\xi) = \min_{\alpha_k\in\Re^n} -2\mathbf{u}_k^\top K(\alpha_k \circ \xi) + n_k(\alpha_k \circ \xi)^\top K(\alpha_k \circ \xi).$$

Note that $\xi = \mathbf{1}$, where $\mathbf{1}$ is a vector of all ones, implies that all the data points are chosen for constructing the subspace $\mathcal{H}_b$, which is equivalent to kernel k-means using the full ker-

nel matrix. As a result, $\mathcal{L}(U, \mathbf{1})$ is the clustering error of the standard kernel k-means algorithm.

The following lemma bounds the expectation of the clustering error.

LEMMA 2. *Given the membership matrix U, we have the expectation of $\mathcal{L}(U, \xi)$ bounded as follows*

$$\mathrm{E}_\xi[\mathcal{L}(U,\xi)] \leq \mathcal{L}(U,\mathbf{1}) + tr\left(\widetilde{U}\left[K^{-1} + \frac{m}{n}[diag(K)]^{-1}\right]^{-1}\widetilde{U}^\top\right),$$

where $\mathcal{L}(U,\mathbf{1}) = tr(K) - tr(\widetilde{U}K\widetilde{U}^\top)$.

PROOF. We first bound $\mathrm{E}_\xi[\mathcal{L}_k(U,\xi)]$ as

$$\begin{aligned}
&\frac{1}{n_k}\mathrm{E}_\xi[\mathcal{L}_k(U,\xi)] \\
=\ & \mathrm{E}_\xi\left[\min_\alpha -2\widehat{\mathbf{u}}_k^\top K(\alpha\circ\xi) + (\alpha\circ\xi)^\top K(\alpha\circ\xi)\right] \\
\leq\ & \min_\alpha \mathrm{E}_\xi\left[-2\widehat{\mathbf{u}}_k^\top K(\alpha\circ\xi) + (\alpha\circ\xi)^\top K(\alpha\circ\xi)\right] \\
=\ & \min_\alpha -2\frac{m}{n}\widehat{\mathbf{u}}_k^\top K\alpha + \frac{m^2}{n^2}\alpha^\top K\alpha + \frac{m}{n}\left(1-\frac{m}{n}\right)\alpha^\top \mathrm{diag}(K)\alpha \\
\leq\ & \min_\alpha -2\frac{m}{n}\widehat{\mathbf{u}}_k^\top K\alpha + \frac{m}{n}\alpha^\top\left(\frac{m}{n}K + \mathrm{diag}(K)\right)\alpha.
\end{aligned}$$

By minimizing over α, we obtain

$$\alpha_* = \left(\frac{m}{n}K + \mathrm{diag}(K)\right)^{-1} K\widehat{\mathbf{u}}_k.$$

Thus, $\mathrm{E}_\xi[\mathcal{L}_k(U,\xi)]$ is bounded as

$$\begin{aligned}
&\mathrm{E}_\xi[\mathcal{L}_k(U,\xi)] + n_k\widehat{\mathbf{u}}_k^\top K\widehat{\mathbf{u}}_k \\
\leq\ & n_k\widehat{\mathbf{u}}_k^\top\left(K - K\left[K + \frac{n}{m}\mathrm{diag}(K)\right]^{-1}K\right)\widehat{\mathbf{u}}_k \\
=\ & \widetilde{\mathbf{u}}_k^\top\left(K^{-1} + \frac{m}{n}[\mathrm{diag}(K)]^{-1}\right)^{-1}\widetilde{\mathbf{u}}_k.
\end{aligned}$$

We complete the proof by adding up $\mathrm{E}_\xi[\mathcal{L}_k(U,\xi)]$ and using the fact that

$$\mathcal{L}_k(U,\mathbf{1}) = \min_\alpha -2\mathbf{u}_k^\top K\alpha + n_k\alpha^\top K\alpha = -\widetilde{\mathbf{u}}_k^\top K\widetilde{\mathbf{u}}_k.$$

□

COROLLARY 1. *Assume $\kappa(x,x) \leq 1$ for any x. Let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$ be the eigen values of matrix K.Given the membership matrix U, we have*

$$\begin{aligned}
\frac{\mathrm{E}_\xi[\mathcal{L}(U,\xi)]}{\mathcal{L}(U,\mathbf{1})} &\leq 1 + \frac{\sum_{i=1}^C \lambda_i/[1+\lambda_i m/n]}{tr(K) - \sum_{i=1}^C \lambda_i} \\
&\leq 1 + \frac{C/m}{\sum_{i=C+1}^n \lambda_i/n}.
\end{aligned}$$

PROOF. As $\kappa(x,x) \leq 1$ for any x, we have $\mathrm{diag}(K) \preceq I$, where I is an identity matrix. As $\widetilde{U}$ is an ℓ_2 normalized matrix, we have

$$\begin{aligned}
&\mathrm{tr}\left(\widetilde{U}\left[K^{-1} + \frac{m}{n}[\mathrm{diag}(K)]^{-1}\right]^{-1}\widetilde{U}^\top\right) \\
\leq\ & \mathrm{tr}\left(\widetilde{U}\left[K^{-1} + \frac{m}{n}I\right]^{-1}\widetilde{U}^\top\right) \\
\leq\ & \sum_{i=1}^C \frac{\lambda_i}{1+m\lambda_i/n} \leq \frac{Cn}{m}
\end{aligned}$$

and

$$\mathcal{L}(U,\mathbf{1}) = \mathrm{tr}(K - UKU^\top) \geq \mathrm{tr}(K) - \sum_{i=1}^C \lambda_i.$$

We complete the proof by combining the above inequalities. □

To illustrate the result of Corollary 1, consider a special kernel matrix K that has its first a eigen values equal n/a and the remaining eigen values equal zero; i.e. $\lambda_1 = \ldots = \lambda_a = n/a$ and $\lambda_{a+1} = \ldots = \lambda_n = 0$. We further assume $a > 2C$; i.e., the number of non-zero eigen values of K is larger than twice the number of clusters. Then, according to Corollary 1, we have

$$\frac{\mathrm{E}_\xi[\mathcal{L}(U,\xi)] - \mathcal{L}(U,\mathbf{1})}{\mathcal{L}(U,\mathbf{1})} \leq 1 + \frac{Ca}{m(a-C)} \leq 1 + \frac{2C}{m},$$

indicating that when the number of non-zero eigen values of K is significantly larger than the number of the clusters, the difference in the clustering errors between the standard kernel k-means and our approximation scheme will decrease at the rate of $O(1/m)$.

5. EXPERIMENTS

In this section, we demonstrate empirically that the proposed algorithm is not only more efficient than kernel k-means in terms of run-time complexity and memory, its performance in terms of clustering error and prediction accuracy is on par with that of the kernel k-means. We have evaluated our algorithm on two popular image data sets: Imagenet and MNIST. We first execute our algorithm on small and medium-sized data sets and compare its performance with that of the full kernel k-means, and then assess its performance on a large data set, for which computing the full kernel is infeasible. All algorithms were implemented in MATLAB and run on an Intel Xeon 2.8GHz processor with 140GB RAM.

5.1 Data sets

- **Small Imagenet:** The Imagenet data set [11] consists of over 1.2 million images that are organized according to the WordNet hierarchy. Each node in this hierarchy represents a concept (known as the "synset"). In order to perform preliminary analysis, we chose $20,000$ images from 12 synsets (odometer, geyser, manhole cover, rapeseed, cliff dwelling, door, monarch butterfly, mountain, daily paper, shoji, villa and website) at the leaf nodes in the hierarchy. We call this data set the *small Imagenet data set*. Figure 2 shows a few examples of the images from the synsets used to form this data set. We extracted keypoints from each image using the VLFeat library [42] and represented each keypoint as a 128 dimensional SIFT descriptor; an average of $3,055$ keypoints were extracted from each image. We employed the spatial pyramid kernel [30] to calculate the pairwise similarity with the number of pyramid levels set to be 4, which has been shown to be effective for object recognition and image retrieval.
- **MNIST:** The MNIST data set [2] is a subset of the database of handwritten digits available from NIST. It contains a total of $70,000$ 784-dimensional binary images from 10 digit classes. Two types of kernel functions were used for this data set, as suggested in [46]: the neural kernel defined as $\kappa(x,y) = tanh(ax^\top y + b)$, and the normalized polynomial kernel defined by $\kappa(x,y) = (x^\top y + 1)^d$. We set the parameters a, b and d to 0.0045, 0.11 and 5, respectively, according to [46].

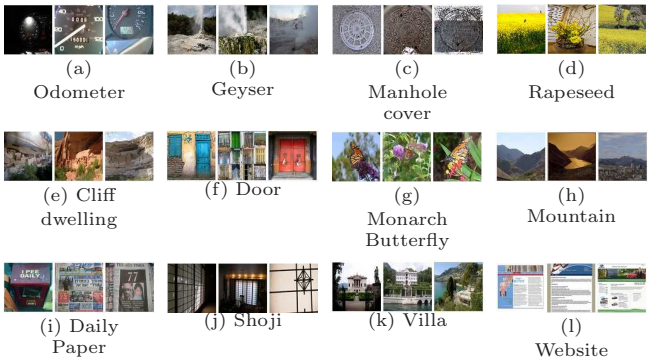


Figure 2: Imagenet Data set

- **Full Imagenet:** To evaluate the performance of our algorithm on very large data sets, we tested it on the full Imagenet data set containing 1,262,102 images. The images are organized to form a synset tree with 1000 leaf nodes, each leaf node representing a class of images. We merged the leaves of this synset tree based on their similarity to form a 164-class data set. The SIFT descriptors were computed using the VLFeat library [42] and a random subset of 10 million SIFT features were clustered to form a visual vocabulary. Each SIFT descriptor was then quantized into a visual word using the nearest cluster center.[2] We obtain a 900-dimensional vector representation for each image, which is then normalized to lie in the range $[0, 1]$. We employed the RBF kernel, with the parameter σ set to 0.1, to compute their pairwise similarity.

5.2 Baseline and evaluation metrics

For the small Imagenet and MNIST data sets, we compare the proposed approximate kernel k-means algorithm with the full kernel k-means and the two-step kernel k-means algorithms using the same randomly initialized membership matrix. For the full Imagenet data set, it is currently infeasible to compute and store the full kernel on a single system due to memory constraints. We compare the performance of our algorithm on this data set with that of the two-step kernel k-means algorithm.

We evaluate the efficiency of the proposed algorithm for different sample sizes ranging from 50 to $10,000$. We measure the time taken for computing the kernel matrix and clustering the data points. We measure the clustering performance using the *error reduction*, defined as the ratio of the difference between the initial clustering error (on random initialization) and the final clustering error (after running the clustering algorithm) to the initial clustering error. Larger the error reduction, greater is the cluster compactness. To evaluate the difference in the clustering results of our algorithm and the full kernel k-means, we calculate the *Adjusted Rand Index* (ARI) [22], a measure of similarity between two data partitions. The adjusted Rand index value lies in $[0, 1]$. A value close to 1 indicates better matching between the two partitions than a value close to 0. We finally examine the prediction accuracy of our algorithm by calculating the Normalized Mutual Information (NMI) [29] with respect to the true class distribution. Analogous to the adjusted Rand index, an NMI value of 1 indicates perfect matching with the true class distribution whereas 0 indicates perfect mismatch.

[2]This data set can be downloaded from `http://image-net.org/download-features`.

Table 1: Results for the small Imagenet data set

C = 12	Preprocessing time: 24,236.22 seconds				
Sample size	Running time (seconds)			ARI	
	Kernel calculation	Approx Kernel k-means	Two-step Kernel k-means	Approx vs Full kernel	Two-step vs Full kernel
Full	33,132.06	81.23 (±43.66)		-	
1000	1564.26 (±63.66)	34.56 (±4.20)	1.05 (±0.28)	0.77 (±0.18)	0.58 (±0.06)
500	810.03 (±24.36)	10.81 (±2.31)	0.01 (±0.001)	0.75 (±0.11)	0.51 (±0.04)
100	203.65 (±20.87)	4.33 (±1.32)	0.02 (±0.005)	0.71 (±0.05)	0.28 (±0.06)
50	79.88 (±4.61)	3.99 (±1.61)	0.01 (±0.001)	0.68 (±0.05)	0.26 (±0.13)

5.3 Experimental results

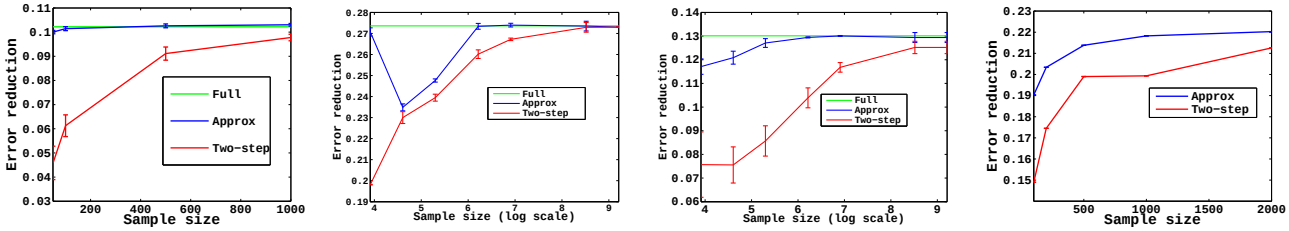
Small Imagenet: Table 1, and Figures 3(a) and 4(a) compare the performance of the proposed algorithm on the small Imagenet data set with full kernel k-means and the two-step kernel k-means. Table 1 lists the running time of the algorithms and the ARI values. Figures 3(a) and 4(a) show the error reduction and NMI achieved by the the three algorithms, respectively. All the results are averaged over 10 runs.

In order to make the comparison meaningful, we divide the running time into three categories: the preprocessing time, needed for constructing the multi-resolution histogram and the spatial pyramid from the SIFT descriptors, the time for computing the kernel matrix, and the time for clustering. Note that the preprocessing time is shared by all the clustering algorithms. We observe that a significant speedup (over 90%) is achieved by both the proposed algorithm and the two-step algorithm in kernel computation when compared to the full kernel k-means. When we compare the clustering time of our algorithm with that of the two-step algorithm for the same sample sizes, the two-step algorithm seems more efficient. This is due to the fact that our algorithm needs to compute the inverse matrix $\widehat{K}^{-1}$. However, when the two algorithms are compared with the requirement that they yield the same clustering performance, we observe from the ARI, error reduction and NMI values that our algorithm is more efficient. With just 50 samples, our algorithm significantly outperforms the two-step algorithm provided with 1000 samples.

We also observe that our algorithm is able to achieve clustering performance equivalent to that of the full kernel k-means. The two-step kernel k-means algorithm yields a much lower error reduction and NMI. This observation indicates that it is insufficient to estimate the cluster centers using only the randomly sampled data points and further justifies the design of the proposed algorithm.

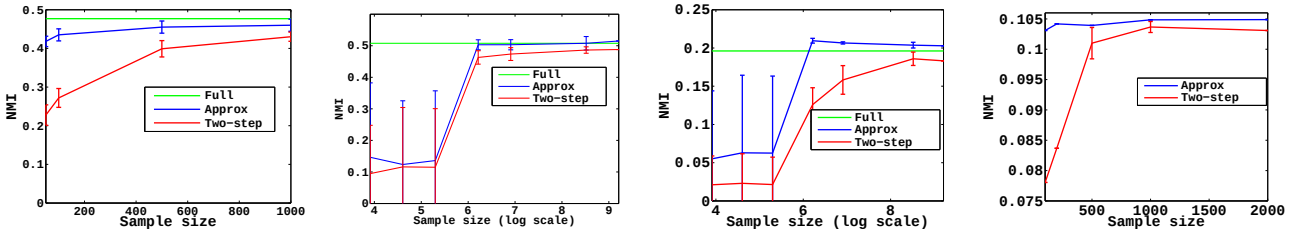
MNIST: Tables 2 and 3 show the results for the MNIST data set using the neural and polynomial kernels, respectively. Compared to the full kernel k-means, we observe that a significant amount of time was saved by the proposed clustering algorithm as well as by the two-step algorithm in clustering the data.

As seen in Figures 3(b) and 3(c), equal amount of error reduction is achieved by both the full kernel k-means and the



(a) Small Imagenet(20k) (b) MNIST using the neural kernel (c) MNIST using the polynomial kernel (d) Full Imagenet

Figure 3: Clustering error reduction



(a) Small Imagenet(20k) (b) MNIST using the neural kernel (c) MNIST using the polynomial kernel (d) Full Imagenet

Figure 4: Normalized Mutual Information with respect to the true class labels

Table 2: Results for the MNIST data set using the neural kernel

C = 10	Kernel parameters: a=0.0045 b=0.11				
Sample size	Running time (seconds)			ARI	
	Kernel calculation	Approx Kernel k-means	Two-step Kernel k-means	Approx vs Full kernel	Two-step vs Full kernel
Full	514.54	3953.62 (±2157.69)		-	
10000	107.58 (±41.38)	13759.05 (±6273.2)	75.93 (±0.01)	0.71 (±0.13)	0.71 (±0.08)
5000	41.07 (±5.12)	1478.85 (±178.97)	14.63 (±6.58)	0.71 (±0.12)	0.70 (±0.12)
1000	7.88 (±0.18)	75.26 (±22.42)	0.59 (±0.08)	0.70 (±0.12)	0.58 (±0.09)
500	5.37 (±0.53)	72.24 (±29.44)	0.25 (±0.02)	0.69 (±0.11)	0.48 (±0.07)
200	5.34 (±0.33)	53.48 (±30.17)	0.18 (±0.03)	0.61 (±0.24)	0.36 (±0.05)
100	2.02 (±0.09)	22.7 (±13.30)	0.06 (±0.002)	0.47 (±0.14)	0.37 (±0.05)
50	1.46 (±0.14)	12.11 (±8.42)	0.05 (±0.002)	0.40 (±0.10)	0.25 (±0.03)

Table 3: Results for the MNIST data set using the polynomial kernel

C = 10	Kernel parameter: degree=5				
Sample size	Running time (seconds)			ARI	
	Kernel calculation	Approx Kernel k-means	Two-step Kernel k-means	Approx vs Full kernel	Two-step vs Full kernel
Full	472.19	2027.68 (±570.63)		-	
10000	119.71 (±38.09)	14407.35 (±7161.1)	47.72 (±0.10)	0.95 (±0.05)	0.87 (±0.03)
5000	47.72 (±5.88)	1449.33 (±229.83)	6.25 (±1.29)	0.90 (±0.06)	0.81 (±0.04)
1000	9.13 (±0.07)	51.54 (±13.23)	0.54 (±0.09)	0.91 (±0.06)	0.72 (±0.07)
500	5.49 (±0.38)	45.59 (±18.40)	0.24 (±0.02)	0.90 (±0.03)	0.65 (±0.08)
200	5.21 (±0.82)	43.03 (±15.47)	0.16 (±0.05)	0.84 (±0.03)	0.33 (±0.05)
100	2.27 (±0.6)	21.55 (±5.60)	0.09 (±0.002)	0.68 (±0.03)	0.41 (±0.03)
50	1.84 (±0.48)	13.16 (±4.32)	0.05 (±0.002)	0.62 (±0.06)	0.32 (±0.16)

Table 4: Results for the full Imagenet data set using the RBF kernel

C = 20	Kernel parameter: $\sigma = 0.1$		
Sample size	Running time (seconds)		
	Kernel calculation	Approx Kernel k-means	Two-step Kernel k-means
2000	344.39	22890.79	24.06
	(±3.77)	(±1987.79)	(±1.97)
1000	181.93	11406.77	11.81
	(±11.95)	(±1170.08)	(±1.36)
500	168.83	2649.13	5.44
	(±0.27)	(±12.15)	(±0.02)
200	68.15	1912.05	2.36
	(±0.16)	(±8.98)	(±0.02)
100	47.29	1119.33	1.34
	(±1.12)	(±43.25)	(±0.01)

proposed algorithm for $m \geq 500$. For $m < 500$, we observed that the algorithm does not always converge to the optimal solution owing to the relatively small sample size, leading to significant variations in the error reduction. Figures 4(b) and 4(c) show the NMI plots for the neural and polynomial kernels, respectively. We again observe that the performance of the proposed algorithm is significantly better than that of the two-step kernel k-means and comparable to that of the full kernel k-means.

Full Imagenet: Over 6.4TB of memory is required to store the full kernel matrix for this data set. Hence, it is currently infeasible to store the full kernel and execute the full kernel k-means on a single system. The proposed approximate scheme alleviates this issue as it does not require the computation of the full kernel. Table 4 compares the running time of our algorithm with that of the two-step kernel k-means algorithm for different sample sizes with the RBF kernel parameter σ set to 0.1. We observe that although the time taken for clustering the points by the our method is considerably greater than that of the two-step method, our algorithm performs significantly better than the two-step algorithm, in terms of both error reduction and NMI, as seen

in Figures 3(d) and 4(d). These results demonstrate that our algorithm is more effective than the two-step algorithm for clustering large data sets.

6. CONCLUSIONS

We have proposed an efficient approximation for the kernel k-means algorithm, suitable for large data sets. The key idea is to avoid computing the full kernel matrix by restricting the cluster centers to a small subspace spanned by a set of randomly sampled data points. We show theoretically and empirically that the proposed algorithm is (i) efficient in both computational complexity and memory requirement, and (ii) is able to yield similar clustering results as the kernel k-means algorithm using the full kernel matrix. In the future, we plan to analytically investigate the sample complexity of the proposed algorithm, i.e. the minimum number of samples required to yield similar clustering results as the full kernel K-mean algorithm. We also plan to further enhance the efficiency of the proposed algorithm through ensemble and semi-supervised clustering techniques.

Acknowledgements

This research was partially supported by the Office of Naval Research (ONR Grant N00014-11-1-0100). Timothy Havens is supported by the National Science Foundation under Grant 1019343 to the Computing Research Association for the CI Fellows Project. Part of Anil Jain's research was also supported by WCU (World Class University) program funded by the Ministry of Education, Science and Technology through the National Research Foundation of Korea (R31-10008). Anil Jain is also affiliated with Korea University, Anam-Dong, Seoul, S. Korea.

7. REFERENCES

[1] `http://gigaom.files.wordpress.com/2010/05/2010-digital-universe-iview_5-4-10.pdf`.

[2] `http://yann.lecun.com/exdb/mnist`.

[3] C.C. Aggarwal, J. Han, J. Wang, and P.S. Yu. A framework for clustering evolving data streams. In *Proceedings of the International Conference on Very Large Databases*, pages 81–92, 2003.

[4] M.A. Belabbas and P.J. Wolfe. Spectral methods in machine learning and new strategies for very large datasets. *Proceedings of the National Academy of Sciences*, 106(2):369–374, 2009.

[5] F. Can. Incremental clustering for dynamic information processing. *ACM Transactions on Information Systems*, 11(2):143–164, 1993.

[6] F. Can, E.A. Fox, C.D. Snavely, and R.K. France. Incremental clustering for very large document databases: Initial MARIAN experience. *Information Sciences*, 84(1-2):101–114, 1995.

[7] C.T. Chu, S.K. Kim, Y.A. Lin, Y.Y. Yu, G. Bradski, A.Y. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. In *Advances in Neural Information Processing Systems 19*, pages 281–288, 2007.

[8] S. Daruru, N.M. Marin, M. Walker, and J. Ghosh. Pervasive parallelism in data mining: dataflow solution to co-clustering large and sparse netflix data. In *Proceedings of the SIGKDD conference on Knowledge Discovery and Data mining*, pages 1115–1124, 2009.

[9] A.S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the International Conference on World Wide Web*, pages 271–280, 2007.

[10] S. Datta, C. Giannella, and H. Kargupta. K-means clustering over a large, dynamic network. In *Proceedings of the SIAM Data Mining Conference*, pages 153–164, 2006.

[11] J. Deng, W. Dong, R. Socher, L.J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[12] I. Dhillon, Y. Guan, and B. Kulis. A unified view of kernel k-means, spectral clustering and graph cuts. Technical report, University of Texas at Austin, 2004. (Tech. rep. TR-04-25).

[13] C. Ding, X. He, and H.D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the SIAM Data Mining Conference*, pages 606–610, 2005.

[14] P. Drineas and M.W. Mahoney. On the Nystrom method for approximating a Gram matrix for improved kernel-based learning. *The Journal of Machine Learning Research*, 6:2153–2175, 2005.

[15] D. Foti, D. Lipari, C. Pizzuti, and D. Talia. Scalable parallel clustering for data mining on multicomputers. *Parallel and Distributed Processing*, pages 390–398, 2000.

[16] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nystrom method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 214–225, 2004.

[17] M. Girolami. Mercer kernel-based clustering in feature space. *IEEE Transactions on Neural Networks*, 13(3):780–784, 2002.

[18] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams: Theory and practice. *IEEE Transactions on Knowledge and Data Engineering*, pages 515–528, 2003.

[19] S. Guha, R. Rastogi, and K. Shim. Cure: an efficient clustering algorithm for large databases. *Information Systems*, 26(1):35–58, 2001.

[20] W. Hackbusch. *Integral equations: Theory and Numerical treatment*. Birkhauser, 1995.

[21] S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 291–300, 2004.

[22] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[23] R. Inokuchi and S. Miyamoto. LVQ clustering and SOM using a kernel function. In *IEEE International Conference on Fuzzy Systems*, volume 3, pages 1497–1500, 2005.

[24] D. Jiang, C. Tang, and A. Zhang. Cluster analysis for gene expression data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1370–1386, 2004.

[25] D. Judd, P.K. McKinley, and A.K. Jain. Large-scale parallel data clustering. *IEEE Transactions on*

Pattern Analysis and Machine Intelligence, 20(8):871–876, 1998.

[26] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Blackwell, 2005.

[27] D.W. Kim, K.Y. Lee, D. Lee, and K.H. Lee. Evaluation of the performance of clustering algorithms in kernel-induced feature space. *Pattern Recognition*, 38(4):607–611, 2005.

[28] S. Kumar, M. Mohri, and A. Talwalkar. Sampling techniques for the Nystrom method. In *Proceedings of the Conference on Artificial Intelligence and Statistics*, pages 304–311, 2009.

[29] T.O. Kvalseth. Entropy and correlation: Some comments. *IEEE Transactions on Systems, Man and Cybernetics*, 17(3):517–519, 1987.

[30] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2169–2178, 2006.

[31] L.L. Liu, X.B. Wen, and X.X. Gao. Segmentation for SAR Image Based on a New Spectral Clustering Algorithm. *Life System Modeling and Intelligent Computing*, pages 635–643, 2010.

[32] D. MacDonald and C. Fyfe. The kernel self-organising map. In *Proceedings of the International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies*, volume 1, pages 317–320, 2002.

[33] D. Newman, A. Asuncion, P. Smyth, and M. Welling. Distributed inference for latent dirichlet allocation. In *Advances in Neural Information Processing Systems*, volume 20, pages 17–24, 2007.

[34] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856, 2001.

[35] R.T. Ng and J. Han. CLARANS: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, pages 1003–1016, 2002.

[36] A. K. Qinand and P. N. Suganthan. Kernel neural gas algorithms with application to cluster analysis. *Pattern Recognition*, 4:617–620, 2004.

[37] T. Sakai and A. Imiya. Fast spectral clustering with random projection and sampling. *Machine Learning and Data Mining in Pattern Recognition*, pages 372–384, 2009.

[38] B. Scholkopf, R. Herbrich, and A. Smola. A generalized representer theorem. In *Proceedings of Computational Learning Theory*, pages 416–426, 2001.

[39] B. Scholkopf, A. Smola, and K.R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1314, 1996.

[40] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[41] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2002.

[42] A. Vedaldi and B. Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org`, 2008.

[43] C. Williams and M. Seeger. Using the Nystrom method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688, 2001.

[44] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the ACM SIGIR Conference*, pages 267–273, 2003.

[45] H. Zha, X. He, C. Ding, M. Gu, and H. Simon. Spectral relaxation for k-means clustering. In *Advances in Neural Information Processing Systems*, volume 2, pages 1057–1064, 2002.

[46] R. Zhang and A.I. Rudnicky. A large scale clustering scheme for kernel k-means. In *Proceedings of the International Conference on Pattern Recognition*, pages 289–292, 2002.

[47] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. *ACM SIGMOD Record*, 25(2):103–114, 1996.