Farthest Centroids Divisive Clustering *

Haw-ren Fang† Yousef Saad†

January 28, 2008

Abstract

A method is presented to partition a given set of data entries embedded in Euclidean space by recursively bisecting clusters into smaller ones. The initial set is subdivided into two subsets whose centroids are farthest from each other, and the process is repeated recursively on each subset. The bisection task can be formulated as an integer programming problem, which is NP-hard. Instead, an approximate algorithm based on a spectral approach is given. Experimental evidence shows that the clustering method often outperforms a standard spectral clustering method, but at a higher computational cost. The paper also discusses how to improve the standard K-means algorithm, a successful clustering method that is sensitive to initialization. It is shown that the quality of clustering resulting from the K-means technique can be enhanced by using the proposed algorithm for its initialization.

Keywords: graph partitioning, K-means algorithm, Lanczos method, spectral bisection, unsupervised clustering

1 Introduction

Research on effective methods to deal with ever larger data sets has been gaining importance in recent years. The goal of clustering is to organize a data collection into clusters, such that items within each cluster are more similar to each other than to items in other clusters. While supervised clustering assumes that some information is available concerning the membership of data items to predefined classes, unsupervised clustering does not require *a priori* knowledge of data contents.

There are many applications of unsupervised clustering in computer vision, pattern recognition, information retrieval, data mining, etc. Examples include document clustering [2], clustering of protein sequences [10], content-based image retrieval [14], image segmentation [12], and DNA microarray analysis [17]. In many cases the data is converted into numerical form, as a set of points in the Euclidean space. The task is to partition the data into subsets

*This work was supported by NSF grants DMS 0510131 and DMS 0528492 and by the Minnesota Supercomputing Institute.

†Computer Science & Engineering, University of Minnesota; Minneapolis, MN 55455.

according to a criterion of closeness. One way to address this problem is to partition the set into subsets whose centroids are as far apart as possible from each other. To simplify the procedure, we follow the recursive bisection approach, which has been used in clustering (e.g., [2, 6, 15, 17]), and is common in graph partitioning (e.g., [7, 11]).

Data clustering algorithms can be categorized into hierarchical or partitional algorithms. There are two types of hierarchical clustering methods; one is agglomerative (bottom-up) and the other is divisive (top-down). The method introduced in this paper is hierarchical and belongs to the latter class, since in order to perform the clustering, we repeatedly bisect a subset into two smaller ones, until some stopping criterion is satisfied or the number of required subsets is reached. A divisive clustering algorithm like this consists of two key components: a way to select a subset to split, and a method to split it into smaller subsets [15]. This article focuses on the second part. We call our method *Farthest Centroids Divisive Clustering* (FCDC), since the bisection scheme aims at obtaining two sets with farthest centroids.

The bisection scheme for farthest centroids is based on a spectral method, followed by a tuning phase to increase the distance between the centroids. Spectral bisection for clustering is not a new idea, see, e.g., [2, 6, 17]. In addition, tuning strategies have been utilized for graph partitioning [8] and for the traveling salesman problem [9]. Experiments show that FCDC usually outperforms spectral clustering methods, albeit at an extra computational cost.

When an appropriate stopping scheme is incorporated, a hierarchical clustering algorithm processes the data into clusters without knowledge of how many clusters there should be. In contrast, partitional algorithms need to be given an idea of how the data should eventually be partitioned. K-Means clustering is such an algorithm. However, while K-means is considered one of the best clustering methods, its results are strongly dependent on the initial choice of cluster representatives. Our experiments showed that the performance of K-Means is usually improved when it is initialized by FCDC, which has the potential to provide more coherent clusters.

Table 1 lists some notation used in this paper. The rest of this paper is organized as follows. Section 2 reviews the spectral bisection method and gives our approximation scheme for dividing a given set into two subsets with farthest centroids. Section 3 presents some clustering algorithms, including FCDC and K-means. Section 4 reports on experimental results. A conclusion is given in Section 5.

Table 1: Notation.

Symbol	*Description*
$\hat{X}$	(raw) data matrix formed by $\hat{x}_1, \ldots, \hat{x}_n \in \mathbb{R}^m$
X	data matrix $\hat{X} - \frac{1}{n}\hat{X}ee^T$ by shifting the centroid of $\hat{X}$ to the origin
n	total number of data points
m	dimension of data points
p, p_1, p_2	number of marginal points in the bisection procedure
K	number of clusters
e	column vector of ones of appropriate dimension

2 Bisection methods

Given a set of points represented in matrix form by $\hat{X} = (\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_n) \in \mathbb{R}^{m\times n}$, the class of problems we address in this section is to partition $\hat{X}$ into two subsets Y and Z of $\hat{X}$ such that the centroids of Y and Z are as far as possible from each other. The bisection methods discussed in this section all preprocess the data to shift its centroid to the origin, so we consider the zero-mean data set $X := \hat{X} - \frac{1}{n}\hat{X}ee^T$. Hereafter, the data matrix considered is the preprocessed X, unless otherwise noted.

2.1 Spectral bisection

Let u_1 and v_1 be the lead left and right singular vectors of $X = (x_1, x_2, \cdots, x_n) \in \mathbb{R}^{m\times n}$ that correspond to the largest singular value σ_1. Spectral bisection separates the translated points by the hyperplane $u_1^T x = 0$. It splits the set into the two subsets, one with all x_i having $u_1^T x_i \leq 0$ and the other containing the rest.

The bisection algorithm just described is equivalent to partitioning the set X according to the signs of the entries of v_1, since $u_1^T X = \sigma_1 v_1$. From this viewpoint, v_1 is just the scaled result of mapping the raw data $\hat{X}$ into one-dimensional space by Principal Component Analysis (PCA), and then partitioning this set into the sets of positive and negative components. Another interesting property is that this scheme maximizes the sum of squared distances from the data points to a bisecting hyperplane. Given a hyperplane $u^T x = 0$ with $u^T u = 1$, the sum of squared distances from $x_1, \ldots, x_n$ to $u^T x = 0$ is

$$\sum_{i=1}^{n} (u^T x_i)^2 = \|u^T X\|_2^2 \leq \sigma_1^2.$$

The maximum is achieved if we choose $u := u_1$, the largest left singular vector of X.

Recall that $Xe = 0$, since the centroid is at the origin. The smallest right singular vector of X is e, which is orthogonal to the largest right singular vector v_1. Therefore $v_1^T e = 0$. This explains why in practice we often obtain approximately an even number of positive and negative entries in v_1.

Spectral bisection has two drawbacks. First, the bisection is made with respect to a linear hyperplane, but often some nonlinear separation is likely to perform better. Second, the resulting two sets are approximately even, whereas a given set may consist of two coherent uneven subsets. Both drawbacks are inherited in the resulting spectral clustering methods.

Alternatively, if we partition the set by seeking two subsets with farthest centroids, the two drawbacks can sometimes be avoided. Figure 1 gives an example, where blue circles and red crosses indicate the two subsets after bisection, respectively. The centroids are marked by solid triangles. The set contains 1,000 points uniformly sampled from $S_1 \cup S_2$, where $S_1 = \{(x, y) : (x + \frac{1}{4})^2 + y^2 < \frac{1}{4}\}$ and $S_2 = \{(x, y) : x > 0, x^2 + y^2 > 4, \frac{x^2}{9} + \frac{x^2}{4} < 1\}$. The result of bisection with farthest centroids has significantly better quality than that obtained from the spectral method.

2.2 Two even sets with farthest centroids

In order to separate $X = (x_1, x_2, \cdots, x_n) \in \mathbb{R}^{m\times n}$ into two subsets in some optimal way, we can try to find these two subsets in such a way that their centroids have the maximum

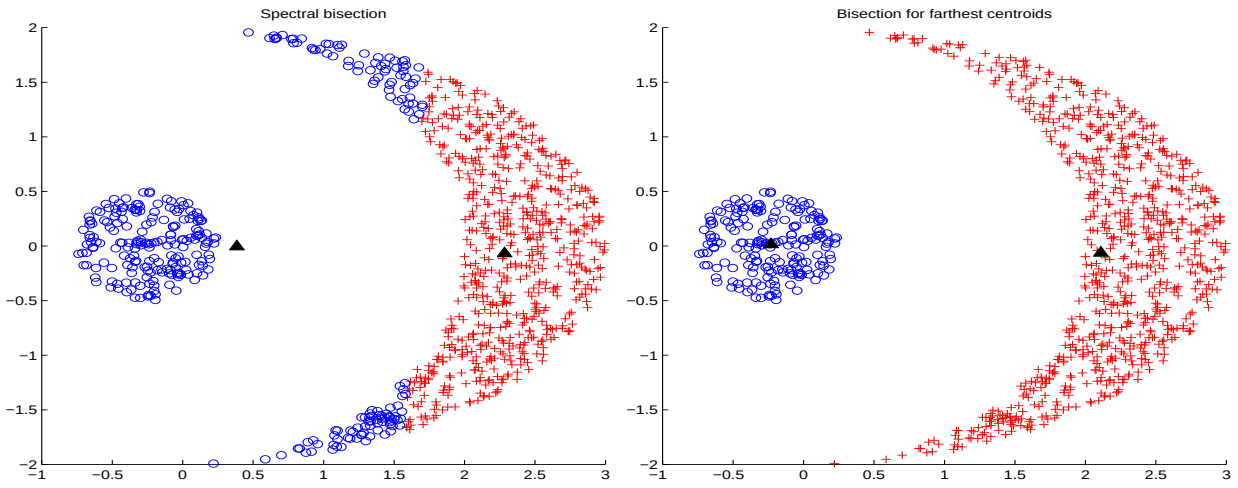

Figure 1: Results of (a) spectral bisection; (b) bisection for farthest centroids.

distance. We denote by Y, Z the two subsets. To simplify the discussion, we let Y, Z be even and will consider uneven bisection in Section 2.3. The optimization problem we need to solve is as follows:

$$\begin{aligned} \text{Maximize} \quad & \|Xc\|_2 && (1)\\ \text{subject to} \quad & c_i = \pm 1, && (2)\\ & \textstyle\sum_{i=1}^{n} c_i = 0, && (3) \end{aligned}$$

where $c = (c_1, c_2, \cdots, c_n)$ with n even. Equation (2) defines the two sets from the signs of the components c_i. All those x_i's for which $c_i = +1$ constitute Y and the others, with $c_i = -1$, constitute Z. The number of data points n is even. Define $c^{(+)} = \frac{1}{2}(c + e)$, the vector obtained from c by replacing its negative entries by zero, where e is the vector of ones. Similarly, let $c^{(-)} = \frac{1}{2}(-c+e)$ be the vector obtained from $-c$ by replacing its negative entries by zero. Then $c = c^{(+)} - c^{(-)}$, and $c^{(+)}$ and $c^{(-)}$ each have only zeros or ones and condition (3) forces them to have the same number of ones.

Since $Xc = Xc^{(+)} - Xc^{(-)}$, maximizing $\|Xc\|_2$ is equivalent to finding c so that the centroids of the corresponding sets Y, Z are farthest apart from each other. Indeed $Xc^{(+)}/(n/2)$ and $Xc^{(-)}/(n/2)$ are the centroids of the two sets Y, Z.

Solving (1) constrained by (2) and (3) can be formulated as a problem of integer programming [16]. However, integer programming is NP-hard and not practical when the data set is large. We look instead for an algorithm to solve approximately this problem. It is possible to employ a Newton-like method, such as a penalty method, or to solve the nonlinear Lagrangian system. However, these methods are easily trapped in local optimas, and are not reliable.

Alternatively, we can *relax* the condition (2) to be $c^T c = n$ and then obtain

$$\begin{aligned} \text{Maximize} \quad & \|Xc\|_2^2 && (4)\\ \text{subject to} \quad & c^T c = n, && (5)\\ & c^T e = 0. && (6) \end{aligned}$$

The solution to the above problem is well-known. Indeed, the problem is equivalent to maximizing the Rayleigh quotient $\langle X^T X c, c\rangle / \langle c, c\rangle$ subject to the constraint $e^T c = 0$. Since e is an eigenvector of $X^T X$ associated with the (smallest) eigenvalue zero, the maximum of the Rayleigh quotient is reached when c equals the eigenvector associated with the largest eigenvalue. Equivalently, we can say that the maximizer of (4) subject to (5) and (6) is the largest right singular vector of X scaled to satisfy (5). We denote it by $\bar{c} = (\,\bar{c}_1, \cdots, \bar{c}_n\,)^T$. Here, condition (6) is implicitly satisfied, since $\bar{c}$ is orthogonal to the least singular vector e of X. Similar properties can be found in graph partitioning by the recursive spectral bisection (RSB) algorithm (e.g., [11]). This discussion leads to the bisection scheme. We first find the median $\bar{c}_M$ of $\bar{c}_1, \ldots, \bar{c}_n$ and then divide X evenly into Y, Z, such that $x_i \in Y$ if $\bar{c}_i < \bar{c}_M$, else $x_i \in Z$. This initial partitioning with given $\bar{c}$ takes $O(n)$ time, using an effective algorithm to find the median of a set of numbers [4, chapter 9].

To improve the two sets in the sense of maximizing the distance between the centroids, we tune the sets Y and Z as follows. By abuse of notation, we use the symbols Y and Z to also denote the two matrices $Y = (\,y_1, y_2, \cdots, y_{n/2}\,)$ and $Z = (\,z_1, z_2, \cdots, z_{n/2}\,)$. The squared distance between the centroids are

$$\left\| \frac{2}{n} \left(\sum_{i=1}^{n/2} y_i - \sum_{i=1}^{n/2} z_i \right) \right\|_2^2 = \frac{4}{n^2} (Ye - Ze)^T (Ye - Ze), \tag{7}$$

where e is the column vector of ones of dimension $n/2$. Assume that we swap two elements $y^* \in Y$ and $z^* \in Z$, and set $\delta := y^* - z^*$. Then the squared distance between the centroids becomes

$$\begin{aligned} & \frac{4}{n^2} \left[((Ye - \delta) - (Ze + \delta))^T ((Ye - \delta) - (Ze + \delta)) \right] \\ = \; & \frac{4}{n^2} \left[(Ye - Ze)^T (Ye - Ze) - 4(Ye - Ze)^T \delta + 4\delta^T \delta \right]. \end{aligned} \tag{8}$$

Comparing (7) and (8), it is seen that the swap will result in an increased distance between the centroids when

$$-4(Ye - Ze)^T \delta + 4\delta^T \delta > 0. \tag{9}$$

Since we can update Ye and Ze as $Ye := Ye - \delta$ and $Ze := Ze + \delta$ after the swap, we need to compute Ye and Ze only once, which makes the use of criterion (9) inexpensive.

Following the terminology in [9], we say that a bisection Y, Z is 1-*opt* if no swapping of two points $y^* \in Y$ and $z^* \in Z$ can increase the distance between the centroids of Y and Z. Moreover, a bisection Y, Z is called k-*opt* ($k>1$), if it is $(k-1)$-opt and no swapping of two sets of points $\{y_1^*, \ldots y_k^*\} \subseteq Y$ and $\{z_1^*, \ldots z_k^*\} \subseteq Z$ can increase the distance between the centroids of Y and Z. A global maximizer of the program (1) is 1-opt but not vice versa. However, a 1-opt separation usually provides a good approximation and can be considered as a local minimizer of (1).

Given a separation Y, Z of X, we may repeatedly swap entries $y^* \in Y$ and $z^* \in Z$ to obtain a longer distance between the centroids of Y and Z (i.e., satisfying (9)). When there is no pair of points y^*, z^* which satisfy (9), we obtain a 1-opt separation. We call this process a *tuning phase*. In the literature, similar strategies have been applied to graph partitioning [8] and the traveling salesman problem [9].

When n is large, $Ye-Ze$ is normally much larger than δ in magnitude, and the dominating term in (9) is $-4(Ye-Ze)^T\delta$, which is positive if $(Ye-Ze)$ and δ form an angle larger than $\pi/2$. Therefore, when the estimation of the two sets with farthest centroids is good, a (y^*, z^*) pair that satisfies (9) typically occurs in the *marginal region* (i.e., the region where entries of the singular vector are smaller in magnitude). With this observation, we consider in the tuning phase only the pairs of points (y^*, z^*) located in the marginal region, rather than all $y^* \in Y$ and $z^* \in Z$. This strategy will reduce computational cost considerably.

Let u_1 and $\bar{c} = (\,\bar{c}_1, \bar{c}_2, \cdots, \bar{c}_n\,)^T$ be the left and right singular vectors of $X = (\,x_1, x_2, \cdots, x_n\,)$ corresponding to the largest singular value σ_1. Then $u_1^T x_i = \sigma_1 \bar{c}_i$. Therefore, the magnitude of c_i is proportional to the distance from x_i to the hyperplane $u_1^T x = 0$. Using this fact, we define the marginal region as the set of points x_i with $\bar{c}_i$ close to $\bar{c}_M$. The scale of the marginal region is controlled by how close $\bar{c}_i$ must be to $\bar{c}_M$ to be in the marginal region. The larger the marginal region, the better chance the resulting partition has of being 1-opt.

After initialization with marginal points determined, we repeat swapping the marginal points of X and Y that increase the distance between the centroids, until no pair is left whose exchange would result in an increased distance between the centroids. The pseudo-code of the proposed scheme is given in Algorithm 1, where we assume $\bar{c}_1, \ldots, \bar{c}_n$ are distinct for simplicity.

Algorithm 1 Bisection for farthest centroids of two even sets.

{Given $X = (\,x_1, \cdots, x_n\,) \in \mathbb{R}^{m\times n}$, partition it into two even sets Y and Z.}
{Here $Xe = 0$ and n even.}
Compute the largest right singular vector of X as $\bar{c} = (\,\bar{c}_1, \cdots, \bar{c}_n\,)^T$.
Find the median of $\bar{c}_1, \ldots, \bar{c}_n$ as $\bar{c}_M$.
Partition the columns of X into Y, Z by $\begin{cases} x_i \in Y & \text{if } \bar{c}_i < \bar{c}_M; \\ x_i \in Z & \text{if } \bar{c}_i > \bar{c}_M. \end{cases}$
Determine s such that indices of all $|\bar{c}_i - \bar{c}_M| \le s$ represent the marginal region of size p.
Set $Y^* := \{x_i \in Y : |\bar{c}_i - \bar{c}_M| \le s\}$ and $Z^* := \{x_i \in Z : |\bar{c}_i - \bar{c}_M| \le s\}$.
repeat
  for all $y^* \in Y^*$ and $z^* \in Z^*$ **do**
    if swapping y^* and z^* increases the distance between the centroids **then**
      Swap y^* and z^* so now $y^* \in Z^* \subseteq Z$ and $z^* \in Y^* \subseteq Y$.
    end if
  end for
until no swapping will increase the distance between the centroids

2.3 Two possibly uneven sets with farthest centroids

In some applications, it may be desirable to partition a given set of points into two possibly uneven sets with farthest centroids. Motivated by the scheme in Section 2.2, we still compute the largest right singular vector $\bar{c}$ of $X = (\,x_1, x_2, \cdots, x_n\,)$, and partition it into Y, Z, according to the signs of the elements in $\bar{c} = (\,\bar{c}_1, \bar{c}_2, \cdots, \bar{c}_n\,)^T$. In other words, x_i is a column vector of Y if $\bar{c}_i \le 0$; otherwise, x_i belongs to Z. Note that this initialization scheme, though motivated differently, is identical with that of the spectral method discussed in Section 2.1.

After the initialization, we now need to tune the resulting sets Y, Z to increase the distance between the centroids. Since the two sets can be uneven, we consider not only swapping two elements $y^* \in Y$ and $z^* \in Z$, but we also allow to move a single entry $y^* \in Y$ to Z or $z^* \in Z$ to Y. The distance between the centroids of the two sets Y and Z is $\|\frac{1}{n_1}Ye - \frac{1}{n_2}Ze\|_2$, where n_1 and n_2 are the numbers of entries in Y and Z, respectively. When two entries $y^* \in Y$ and $z^* \in Z$ are swapped, the distance becomes $\|\frac{1}{n_1}(Ye - y^* + z^*) - \frac{1}{n_2}(Ze + y^* - z^*)\|_2$. On the other hand, moving a single entry $y^* \in Y$ to Z results in the distance $\|\frac{1}{n_1-1}(Ye - y^*) - \frac{1}{n_2+1}(Ze + y^*)\|_2$. Since updating Ye and Ze for a single swap or move only takes four and two vector additions/subtractions, respectively, we do not have to recompute Ye and Ze, and the cost to evaluate the new distance is low.

In the tuning phase, we consider only the marginal points as in Section 2.2, and repeatedly try swapping pairs and moving singles entries from the marginal set, until no swap or single move can increase the distance between the centroids. The marginal points consist of the points x_i with $\bar{c}_i$ close to zero. Let p be the number of marginal points. The overall cost of single moves is normally $O(pm)$, whereas for swaps it takes $O(p^2m)$. This observation suggests that it is worthwhile to use two different marginal regions, a primary one of size p_1 for single moves, and a secondary one of size $p_2 < p_1$ for swaps. In practice, bisections of reasonably good quality can be obtained with single moves – but not as good, on average, as when swaps are also performed. To reduce the computational cost, p_2 is set much smaller than p_1. Exchanges of pairs only provide small adjustments for minor improvements. The pseudo-code is given in Algorithm 2.

Algorithm 2 Bisection for farthest centroids of two possibly uneven sets.

{Given $X = (\, x_1, \cdots, x_n\,) \in \mathbb{R}^{m\times n}$, partition it into two possibly uneven sets Y and Z.}
Compute the largest right singular vector of X as $\bar{c} = (\, \bar{c}_1, \cdots, \bar{c}_n\,)^T$. Note that $Xe = 0$.
Partition the columns of X into Y, Z by $\begin{cases} x_i \in Y & \text{if } \bar{c}_i \le 0; \\ x_i \in Z & \text{if } \bar{c}_i > 0. \end{cases}$
Find s such that indices of all $|\bar{c}_i| \le s$ represents the primary marginal region of size p_1.
Find t such that indices of all $|\bar{c}_i| \le t$ represents the secondary marginal region of size p_2.
repeat
  for all $w^* \in \{x_i : |\bar{c}_i| \le s\}$ **do**
    if $w^* \in Y$ and moving w^* to Z repels the centroids from each other **then**
      Move $w^* \in Y$ to Z.
    else if $w^* \in Z$ and moving w^* to Y repels the centroids from each other **then**
      Move $w^* \in Z$ to Y.
    end if
  end for
  for all $y^* \in \{x_i : x_i \in Y, |\bar{c}_i| \le t\}$ and $z^* \in \{x_i : x_i \in Z, |\bar{c}_i| \le t\}$ **do**
    if swapping y^* and z^* repels the centroids from each other **then**
      Swap y^* and z^* so now $y^* \in Z$ and $z^* \in Y$.
    end if
  end for
until no swap or single move can increase the distance between the centroids.

Sometimes the full flexibility of unevenness may result in an undesired partitioning.

For example, suppose the data points are sampled uniformly from the unit disk in two-dimensional space. When the number of data points n is large, the distance between the farthest centroids is approximately 1 and it corresponds to taking one set to consist of one edge point and the other set to consist of the remaining $n-1$ points. In practice, we can balance the evenness of the two subsets and the distance of their centroids by controlling the magnitude of the marginal points.

One practical issue not addressed in Algorithms 1 and 2 is the order in which $z^* \in Z^*$ and $y^* \in Y^*$ are to be tried for swaps or single moves. Different orders may result in different bisections and thus different clusterings. In our experiments, we found that the order had no significant impact on the quality of clustering. In particular, one can try to select the pair which will cause the largest move of the centroids at each step. A few tests along these lines yielded considerable increase of computational cost but little improvements of the clusters.

2.4 Approximation by the Lanczos method

The largest right singular vector of $X \in R^{m\times n}$ is the largest eigenvector of X^TX, and it can be computed by the Lanczos method. The number of Lanczos steps is *at most* $\text{rank}(X^TX) \leq \min\{m, n-1\}$. Assuming that the the matrix-vector multiplication $(X^TX)q$ required at each Lanczos iteration are computed as $X^T(Xq)$, the cost to compute the largest singular vector of X is *at most* $O(m^2n)$. For data in a high dimensional space (i.e., m is comparable to or larger than n), this worst-case cost could be $O(n^3)$. When the data set is sparse (as in text data for example), each matrix-vector product will cost only $O(\alpha n)$, where α is the average number of nonzero entries per column of $\hat{X}$. This would lead to a maximum cost of $O(\alpha mn)$ whether or not $m \ll n$. Note however that these costs are pessimistic since the number of steps required by the Lanczos algorithm will typically be much smaller than m. This important consideration is discussed next.

The spectral bisection requires only the signs of the entries of the largest right singular vector $\bar{c}$ of X. In addition, the tuning phase requires the information of the order of entries in $\bar{c}$ to determine the marginal points. However, an accurate singular vector is not required. For data from high dimensional space, we may use the Lanczos algorithm with few Lanczos steps to find an approximate singular value and vector. If the number of Lanczos steps is a small constant, then the cost is of order $O(mn)$.

Recall that $X = \hat{X} - we^T$, where $\hat{X}$ is the raw data matrix and $w = \frac{1}{n}\hat{X}e$ the centroid. When the raw data $\hat{X}$ is sparse, it is most economical to compute the matrix-vector products Xq as $\hat{X}q - w(e^Tq)$ and X^Tr as $\hat{X}^Tr - e(w^Tr)$. These properties have been observed in [2].

Table 2: Data sets used for Lanczos test.

Data set	*Image type*	*Image size*	*# subjects*	*# images per subject*	*data matrix size*
ORL face database	grayscale	112×92	40	10	$10,304 \times 400$
UMIST face database	grayscale	112×92	15	19–48	$10,304 \times 575$
digit images	binary	20×16	10	39	320×390
alphabet images	binary	20×16	26	39	$320 \times 1,014$

We conducted four tests with data in high dimensional space, from ORL face database [13], UMIST face database [5], and binary images of digits and alphabet letters, respectively. These data sets were also used in the experiments in Section 4. The first test set was the ORL (Olivetti Research Laboratory) database [13], which contains 400 face images of 40 subjects, 10 images per subject. For the second test we used the UMIST face database [5] which contains 575 pre-cropped images of 20 subjects. In both face databases each image has 112×92 grayscale pixels. In the third test we used a set of $10 \times 39 = 390$ images of digits, and in the fourth we had $26 \times 39 = 1,014$ images of alphabet letters, all handwritten and of size 20-by-16 with all entries 0 or 1. The data matrices were obtained by 'vectorizing' the images. Additional information is shown in Table 2.

We assessed the approximate largest singular vector at each Lanczos iteration by the *mean sign error* and the *relative mean order error*. The mean sign error is the number of all sign errors divided by the number of data entries n. The relative order error of an entry in a vector is the difference in magnitude between its order in the vector and that in the approximate vector, divided by n. Figure 2 shows the result. In all cases we obtained the correct signs of the entries of the largest right singular vector of X within 10 Lanczos iterations. For 15 Lanczos iterations we also obtained good estimate of the order of entries.

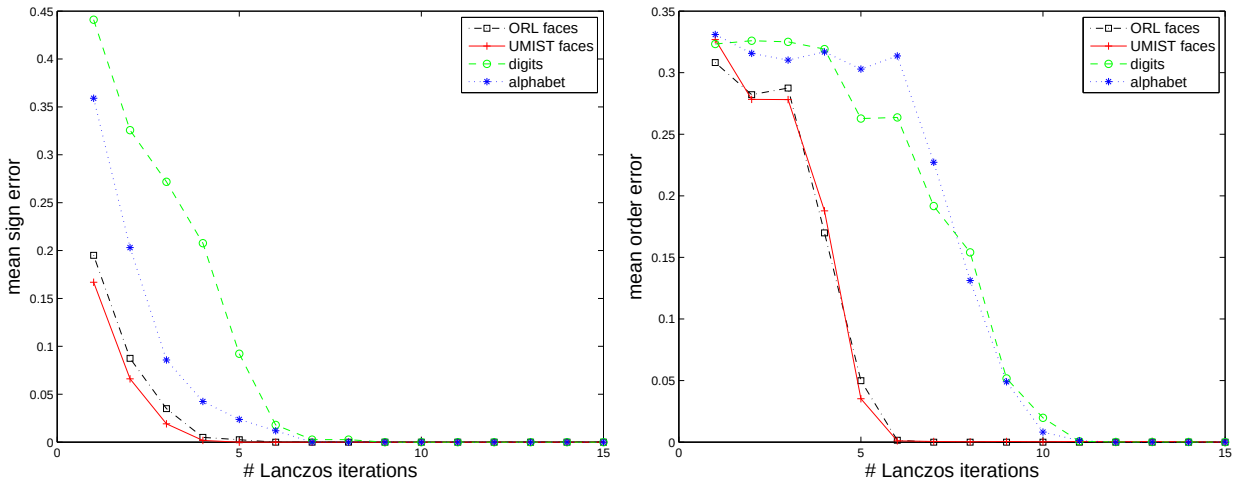

Figure 2: Results of the Lanczos method for approximating the largest right singular vector.

We now discuss the overall computational cost for a farthest centroids bisection. After getting the largest right singular vector of X or its approximation, it takes $O(n)$ time to determine the marginal region, using an algorithm for the selection problem in [4, chapter 9.3]. Then, the tuning normally takes $O(p_1 m)$ time for single moves of points and $O(p_2^2 m)$ for swaps, where p_1 and p_2 are the numbers of points in the primary and secondary marginal regions, and m is the dimension of data points. Here p_1 is at most the total number of points n. If $p_2 = O(\sqrt{n})$, then the tuning cost is $O(nm)$, the same order of Lanczos approximation for the largest singular vector of X. Therefore the extra cost in the tuning phase over the spectral bisection is a constant factor. In our experiments, we set $p_2 = \sqrt{n}$ (rounded to the nearest integer) for computational efficiency. To achieve significant improvement over the

spectral methods, we usually need the number of marginal points $p_1 = 0.1n$ or higher. To control the cost, we normally set $p_1 = 0.4n$ or lower.

3 Unsupervised clustering

Given a set of points $x_1, \ldots, x_n$ in Euclidean space, we want to partition it into a certain number of subsets, called *clusters*, which are as 'distinct' as possible. Sections 3.1 and 3.2 describe two different types of clustering methods. The criteria of the performance evaluation are given in Section 3.3.

3.1 Clustering by recursive bisections

One can cluster by means of recursive bisections. A clustering algorithm of this type comprises two ingredients. The first consists of ways in which to select the subset to split and the second consists of methods for splitting the selected subset [15].

For the second part, a spectral method [2, 6, 17], performs a bisection by using the principal direction as described in Section 2.1. On the other hand, we can employ the bisecting K-means algorithm [15] (i.e., by setting $K := 2$ in the K-means algorithm). The resulting clustering method will be referred to as the *bisecting K-means divisive partitioning* (BKDP) in this paper. Alternatively, the bisection may exploit farthest centroids as discussed in Sections 2.2 and 2.3. We call the resulting method the *Farthest Centroids Divisive Clustering* (FCDC).

For the first part, various methods have been proposed in the literature to choose the next cluster to bisect, including:

1. Select the set according to the largest singular values of the current clusters [6, 17].

2. Select the cluster with the largest *total scatter value* [2]. The total scatter value of a cluster Y with data points $y_1, \ldots, y_k$ is defined by

$$\sum_{i=1}^{k} \|y_i - \bar{y}\|_2^2, \qquad \bar{y} = \frac{1}{k} \sum_{i=1}^{k} y_i.$$

3. If we know in advance that the clusters are of approximately equal size, we may choose the largest cluster [15].

Principal Direction Divisive Partitioning (PDDP) [2], a spectral clustering method, selects the cluster with the largest total scatter value to bisect. In our experiments, we also choose the cluster with the largest total scatter value for FCDC and BKDP.

3.2 K-means clustering algorithm

The K-means algorithm is one of the best-known clustering methods available. The algorithm can be succinctly described by defining the *quantization error*, the sum of squared distances

from the entries to the their cluster prototypes:

$$E(s, w) = \sum_{i=1}^{n} \|x_i - w(s(i))\|_2^2, \tag{10}$$

where $s(i)$ is the index of the cluster to which x_i belongs, and $w(j)$ is the prototype, e.g., the centroid, of cluster j. When the clustering s is fixed, the minimizer of (10) in terms of w is when $w(j)$ is the centroid of the data entries in cluster j. On the other hand, if w is fixed, the minimizer of (10) in terms of s is reached when $s(i)$ is the cluster index of the closest prototype to x_i. K-means iteratively minimizes $E(s, w)$ in terms of s and w, until the value of $E(s, w)$ cannot be further reduced. K-means is an Expectation-Minimization (EM) algorithm whose goal is to (locally) minimize (10) [3]. The pseudo-code is given in Algorithm 3.

Algorithm 3 K-means clustering algorithm.

{Given $X = (\, x_1, \cdots, x_n\,) \in \mathbb{R}^{m\times n}$, partition it into K clusters $S_1, \ldots, S_K$.}
Initialize K prototypes $p_1, \ldots, p_K$, randomly or by another clustering algorithm.
repeat
  Set $S_j := \emptyset$ for $j = 1, \ldots, K$.
  for $i = 1, 2 \ldots, n$ **do**
    Find k such that $\|x_i - p_k\| \le \|x_i - p_j\|$ for $j = 1, \ldots, K$.
    Set $S_k := S_k \cup \{x_i\}$.
  end for
  for $j = 1, 2, \ldots, K$ **do**
    Set $p_j =$ the mean of points in S_j.
  end for
until it converges (i.e., $p_1, \ldots, p_K$ unchanged).

It is not yet specified how the prototypes are initialized. In fact the K-means algorithm is sensitive to the initialization. In our experiments we used four different initializations: random, by PDDP, by BKDP, and by FCDC.

3.3 Clustering evaluation

Several criteria exist to measure the performance of clustering. One of them is the quantization error (10). For comparisons between different clustering tasks, we use the *relative quantization error*:

$$\bar{E}(s) = \hat{E}(s)/\|X\|_2^2, \qquad \hat{E}(s) = \min_w E(s, w). \tag{11}$$

If each data entry is assigned a label (i.e., the 'right' clustering is known in advance), then we can evaluate the clustering performance by total entropy [2]. The entropy of cluster j is defined by

$$e_j = -\sum_i \frac{s(i,j)}{n_j} \log_2 \frac{s(i,j)}{n_j}, \qquad n_j = \sum_i s(i,j),$$

where $s(i,j)$ is the number of occurrences of label i in cluster j, and n_j is the number of data entries in cluster j. The total entropy is the weighted average of e_j,

$$e_{total} = \frac{1}{n}\sum_j n_j e_j. \tag{12}$$

The smaller the total entropy, the better the performance. If the clusters exactly match the labels, then the total entropy is zero.

We can also define the clustering error rate and count as

$$e_{rate} = \frac{1}{n} e_{count}, \qquad e_{count} = \sum_j (n_j - \max_i s(i,j)), \tag{13}$$

respectively. If the clusters match the labels, then the clustering error is zero.

4 Clustering experiments

We conducted five experiments on clustering: clustering of ORL face images [13], clustering of UMIST face images [5], clustering of digit and alphabet images, document clustering, and color clustering. A total of seven different methods were used: PDDP, BKDP, FCDC, and K-means initialized randomly, by PDDP, by BKDP, and by FCDC. The results are reported in Sections 4.1, 4.2, 4.3, 4.4, and 4.5, respectively. For BKDP, the spectral bisection method is used for K-means ($K = 2$) initialization. PDDP and FCDC also utilize spectral bisections. In all cases we used a MATLAB routine **svd1triple** from [2] that approximates the lead singular value and vectors by the Lanczos method. It also takes advantage of the sparsity of the data, if any, to reduce the computation time. For FCDC, we set the marginal region to be of size $\sqrt{n}$ (rounded to the nearest integer) for swaps, where n is the number of data points. The size of the marginal region for single moves varies from $0.1n$ to $0.4n$, to balance the computational cost and quality of clusters. The experiments were performed in sequential mode on a PC equipped with two Intel(R) Xeon(TM) 3.00GHz processors.

4.1 ORL face images

In the first experiment we used ORL (Olivetti Research Laboratory) database of faces [13]. It contains 40 subjects each having 10 grayscale images of size 112-by-92 with various facial expressions, giving a total of 400 images. Figure 3 displays the images of the first two subjects.

We used the first k images of each subject for $k = 2, \ldots, 10$. After vectorizing the images, we obtained a matrix X of size 10304-by-$40k$. Then we clustered it into 40 subsets. For FCDC we set the size of the marginal region for single moves of points be $0.25n$, with n the number of images in each bisection. The result is shown in Figure 4, where we report the total entropy (12) and (used) CPU time. The performance comparison is summarized as follows. FCDC was usually better than BKDP with comparable used CPU time. Both outperformed PDDP, which is the most economic. FCDC initialization also usually performed better than PDDP or BKDP initializations for enhancing the K-means algorithm.

Figure 3: Sample images from the ORL face database.

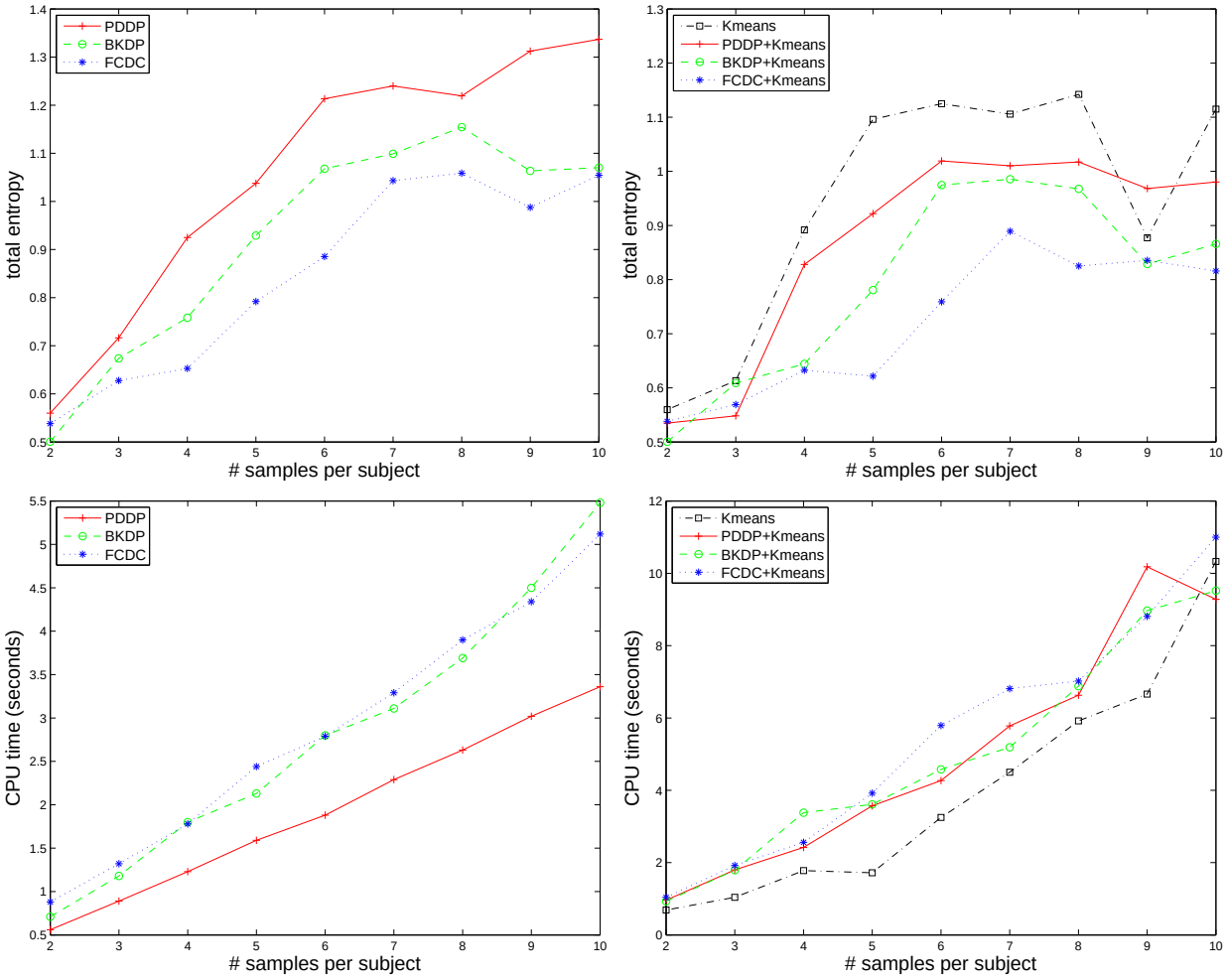

Figure 4: Total entropies and CPU time for ORL face image clustering.

For the number of images per subject $k = 10$ (i.e., using all images), the quantization error (10) (11), total entropy (12), the clustering error (13), and used CPU time for each method are displayed in Table 3.

Table 3: Performance of various clustering methods on ORL face images.

Method	*PDDP*	*BKDP*	*FCDC*	*K-means*	*PDDP + K-means*	*BKDP + K-means*	*FCDC + K-means*
Quantization error	44516.56 (4.63%)	42541.48 (4.42%)	42375.99 (4.40%)	42988.88 (4.47%)	40758.23 (4.24%)	40579.25 (4.22%)	39813.95 (4.14%)
Total entropy	1.337	1.070	1.055	1.115	0.980	0.866	0.816
Clustering error	166 (41.50%)	135 (33.75%)	133 (33.25%)	143 (35.75%)	132 (33.00%)	112 (28.00%)	109 (27.25%)
Time (secs)	3.36	5.48	5.12	10.33	9.28	9.52	11.00

4.2 UMIST face images

In the second experiment we employed UMIST face database [5] which contains 575 pre-cropped images of 20 subjects. For FCDC we set the size of the marginal region for single moves of points to be $0.4n$, with n being the number of images in each bisection. The images are of size 112-by-92 in 256 shades of grey. The number of images per subject varies from 19 to 48, covering a range of poses from profile to frontal views. This set gives a more challenging task for clustering than the ORL face database. Figure 5 displays the 38 images of the first subject.

Figure 5: Images of the first subject in UMIST database.

In each test we used a certain fraction of images of each subject, for 10%,20%,...,100%. Then they were clustered into 20 subsets. Figure 6 shows the total entropy (12) using various clustering methods. We summarize the result as follows. On average , FCDC slightly outperformed BKDP using a comparable CPU time. PDDP was least expensive but usually resulted in higher entropy. K-means algorithm also worked well. There was no clear improvement by FCDC, BKDP, or PDDP initialization.

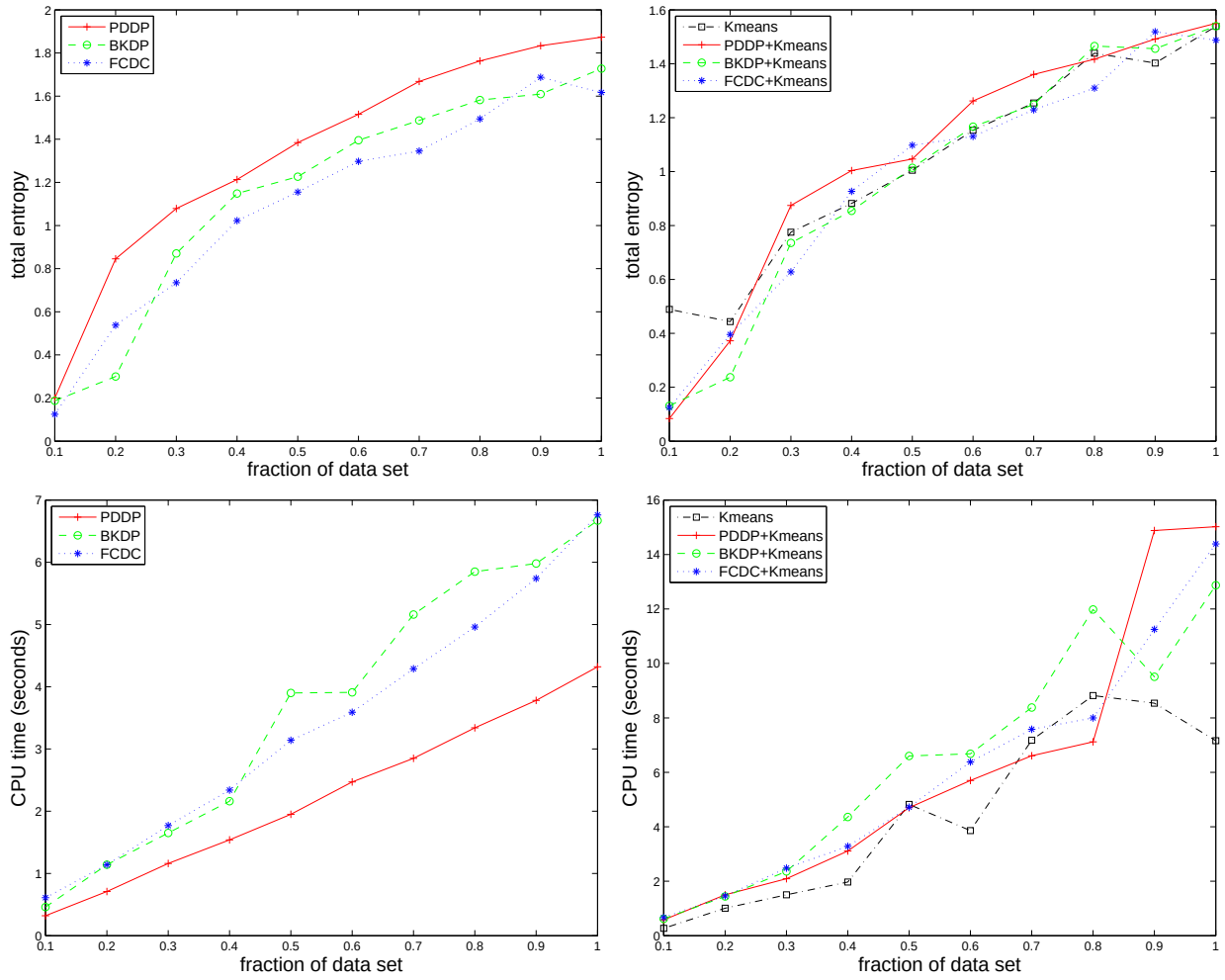

Figure 6: Total entropies and CPU time for UMIST face image clustering.

Using the whole 575 face images, the quantization errors (10) (11), total entropies (12), the clustering errors (13), and CPU time used are reported in Table 4.

4.3 Alphabet and digit images

The third experiment set consists of binary images of written digits and alphabet letters, 39 images per digit and letter, in a total of $(10 + 26) \times 39 = 1,404$ images. Each image is of size 20-by-16. The means (centroids) of the digits and letters are shown in Figure 7. The look of these generally suggests that the digits are well-written (The images would be very fuzzy if many badly written digits/letters were present.)

We divided the set into two subsets of digits and alphabet letters, respectively. For the subset of digits, we used the first k images of each digit for $k = 2, \ldots, 39$. After vectorizing the images, we obtained a data matrix of size 320-by-$10k$, and then partitioned it into 10 clusters using various methods. For FCDC we set the number of points (marginal size) for single moves of points be $0.4n$, where n is the number of data points in each bisection. Figure 8

Table 4: Performance of various clustering methods on UMIST face images.

Method	*PDDP*	*BKDP*	*FCDC*	*K-means*	*PDDP + K-means*	*BKDP + K-means*	*FCDC + K-means*
Quantization error	77012.34 (8.09%)	71898.65 (7.55%)	71324.06 (7.49%)	72283.33 (7.60%)	68441.44 (7.19%)	69661.93 (7.32%)	67987.88 (7.14%)
Total entropy	1.873	1.728	1.617	1.539	1.550	1.540	1.487
Clustering error	315 (54.78%)	322 (56.00%)	296 (51.48%)	281 (48.87%)	293 (50.96%)	307 (53.39%)	288 (50.09%)
Time (secs)	4.32	6.67	6.76	7.16	15.02	12.87	14.39

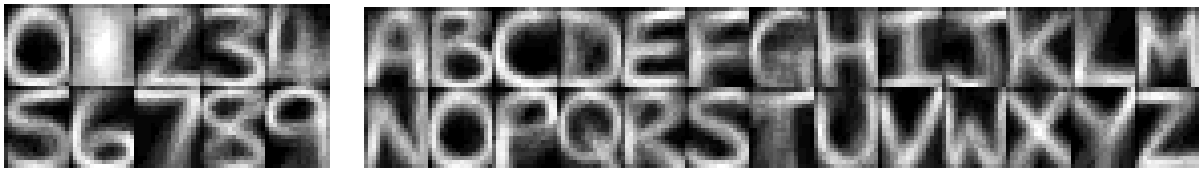

Figure 7: The means (centroids) of all digits and alphabet letters.

shows the result, with the performance measured by total entropy (12). Figure 9 exhibits the result of clustering the subset of alphabet letters into 26 clusters. In this experiment FCDC was usually slightly faster than BKDP without losing the quality of the clusters. Both FCDC and BKDP, with a higher cost, outperformed PDDP. Using the K-means algorithm, FCDC initialization also slightly improved the K-means algorithm. The details are not reported.

For the number of images per subject $k = 39$ (i.e., using all images), the results of various performance evaluation criteria are shown in Tables 5 and 6.

Table 5: Performance of various clustering methods on digit images.

Method	*PDDP*	*BKDP*	*FCDC*	*K-means*	*PDDP + K-means*	*BKDP + K-means*	*FCDC + K-means*
Quantization error	21641.29 (40.30%)	21079.03 (39.25%)	21167.05 (39.42%)	20337.76 (37.87%)	20310.38 (37.82%)	20340.89 (37.88%)	20090.92 (37.41%)
Total entropy	1.887	1.586	1.498	1.312	1.423	1.308	1.082
Clustering error	209 (53.59%)	182 (46.67%)	151 (38.72%)	141 (36.15%)	182 (46.67%)	144 (36.92%)	118 (30.26%)
Time (secs)	0.13	0.32	0.23	0.57	0.49	0.76	0.85

4.4 Document clustering

The clustering methods we have presented can also be used in document clustering. A document vector $d = [d_1, d_2, \ldots, d_n]^T$ is a column vector whose ith entry is the relative

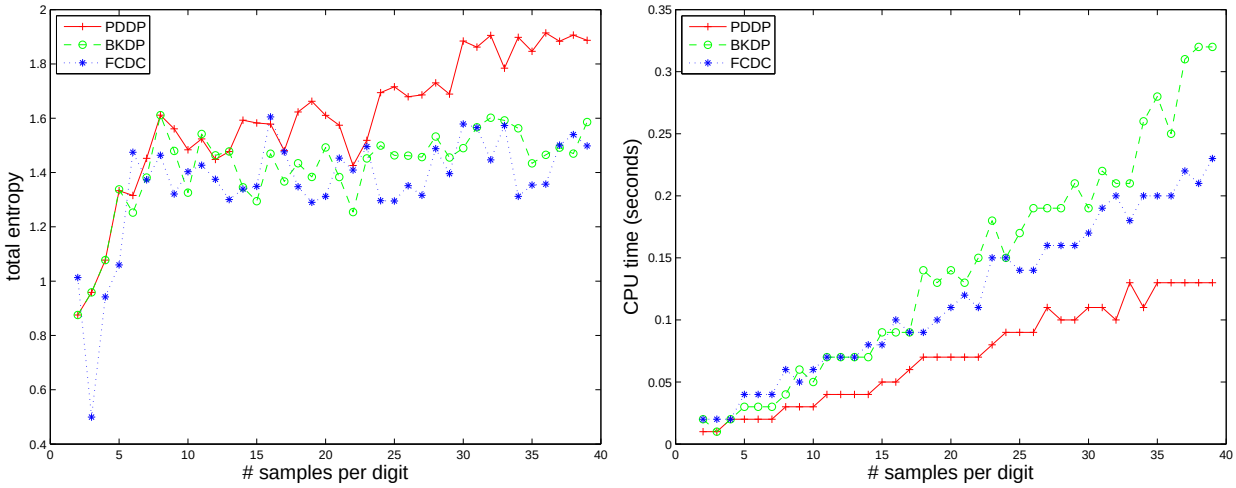

Figure 8: Total entropies and CPU time for digit image clustering.

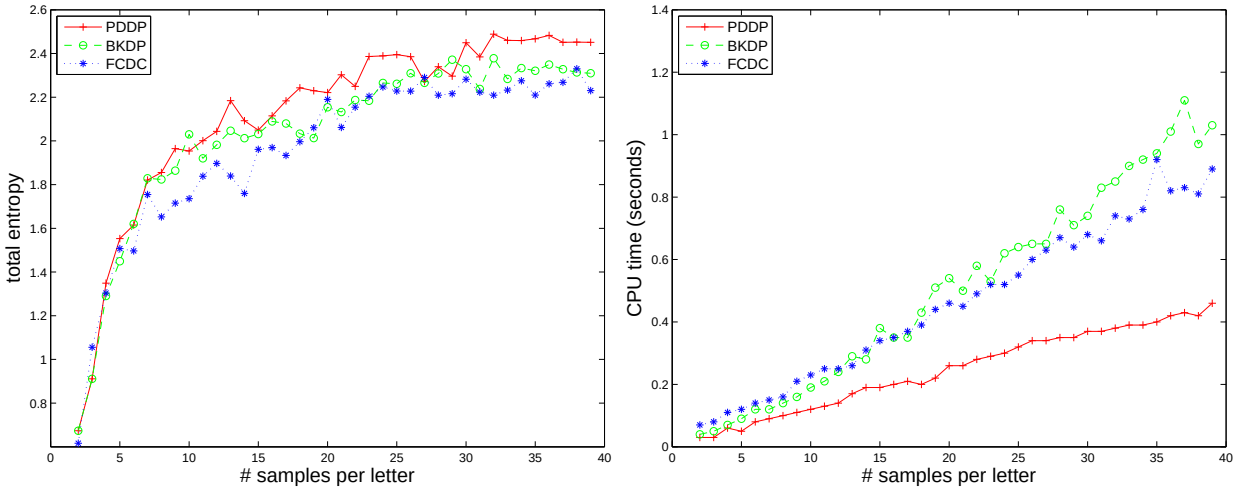

Figure 9: Total entropies and CPU time for alphabet image clustering.

Table 6: Performance of various clustering methods on alphabet images.

Method	*PDDP*	*BKDP*	*FCDC*	*K-means*	*PDDP* + *K-means*	*BKDP* + *K-means*	*FCDC* + *K-means*
Quantization error	54055.61 (41.06%)	53073.60 (40.31%)	53013.76 (40.27%)	51441.60 (39.08%)	51286.01 (38.96%)	51023.83 (38.76%)	51196.15 (38.89%)
Total entropy	2.451	2.310	2.230	2.054	2.130	2.031	1.988
Clustering error	585 (57.69%)	574 (56.61%)	560 (55.23%)	520 (51.28%)	544 (53.65%)	528 (52.07%)	521 (51.38%)
Time (secs)	0.46	1.03	0.89	5.82	3.13	3.49	3.33

frequency of the ith word. We scale the document vectors to have Euclidean norm equal to 1, so that each entry has numerical value equal to $d_i = \frac{TF_i}{\sqrt{\sum_j (TF_j)^2}}$, where TF_i is the number of occurrences of word i in the particular document set. This scaling is referred to as norm scaling.

For our experiments we used the data sets J1-J11 from Boley [2]. All these 11 sets consist of the same 185 documents but differ in the number of key words from 183 to 10,536, where J1 contains all words and forms a matrix of size 10,536-by-185. Each document has been assigned by hand a label according to its topic. There are 10 different labels (topics) in total. These labels are required for the computation of the total entropy (12), to evaluate the quality of the clusters. In all tests this is the only experiment with sparse matrices, which are favored by the spectral methods such as PDDP. See [2, section 5] for more information of these data sets.

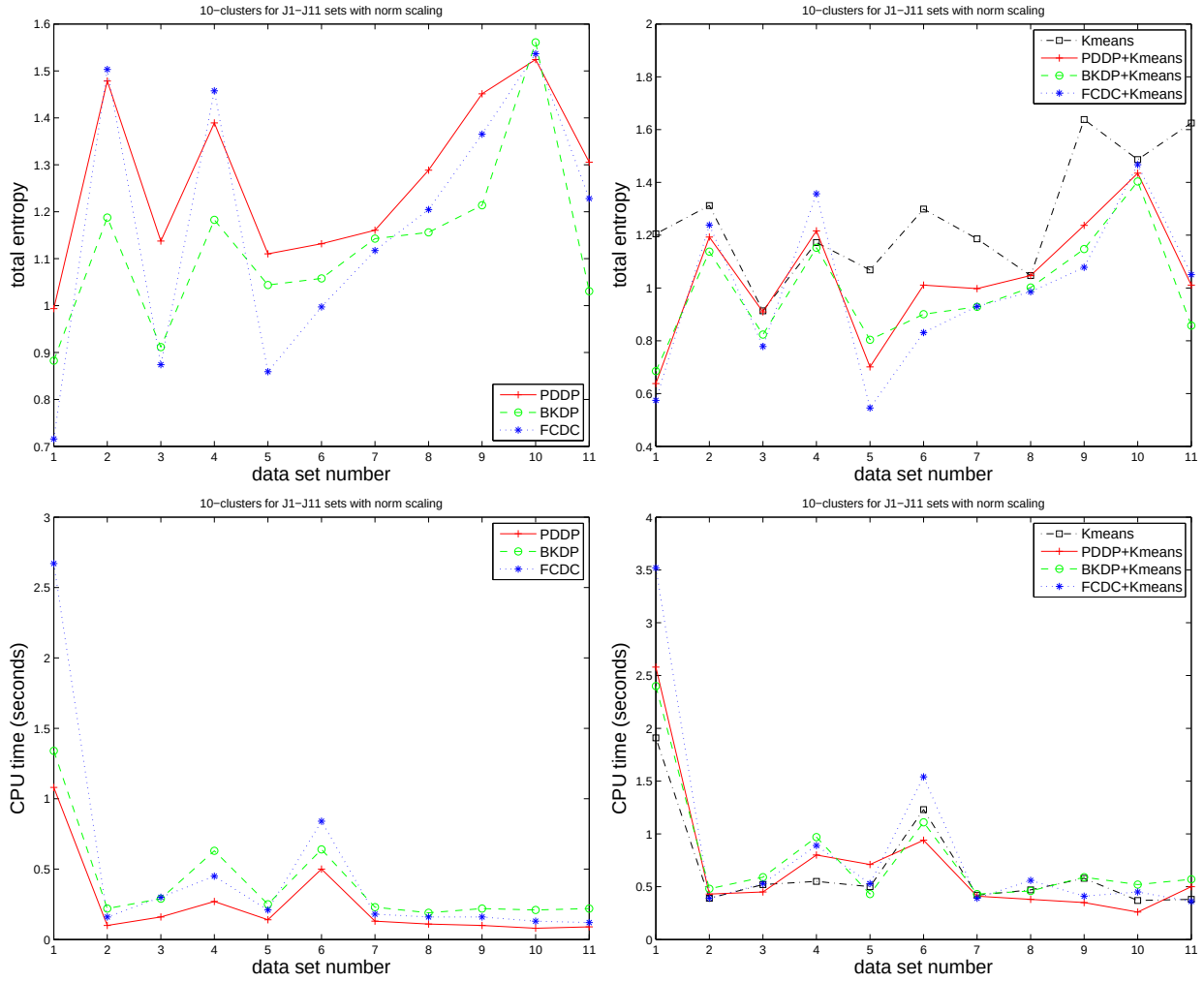

Figure 10: Total entropies and CPU time for document clustering.

The results of clustering the data into $K = 10$ clusters are presented in Figure 10. For FCDC we set the marginal region to be of size $0.1n$, where n is the number of documents

in each bisection. We can observe that FCDC and BKDP produced lower entropy values than PDDP for almost all the data sets, and that FCDC, BKDP and PDDP initializations improved the K-means algorithm. Experiments were also conducted with other numbers of clusters $K = 8, 16, 32$. The relative performances were comparable. For example, Table 7 lists the results on J1 data set which contains the complete 10,536 key words. In this case FCDC outperformed BKDP and PDDP in terms of the quality of clusters, but with a higher computational cost. FCDC also achieved the best enhancement of the K-means algorithm.

Table 7: Performance of various clustering methods on J1 data set.

# of clusters	*Methods*	*PDDP*	*BKDP*	*FCDC*	*K-means*	*PDDP + K-means*	*BKDP + K-means*	*FCDC + K-means*
8	Total entropy	1.236	1.132	0.953	1.405	1.005	1.015	0.795
	Time (secs)	1.10	1.32	2.65	2.48	1.89	1.79	3.35
10	Total entropy	0.993	0.883	0.716	1.205	0.638	0.685	0.574
	Time (secs)	1.08	1.34	2.67	1.91	2.58	2.40	3.52
16	Total entropy	0.690	0.655	0.492	1.251	0.655	0.660	0.436
	Time (secs)	1.23	1.53	2.83	2.11	2.25	2.22	3.50
32	Total entropy	0.513	0.512	0.392	0.656	0.513	0.512	0.392
	Time (secs)	1.39	1.81	3.08	2.83	2.07	2.51	3.81

In additional to norm scaling, Boley also used TFIDF scaling in his experiments [2]. In our tests on J1-J11 sets with data matrices TFIDF scaled, we found that both FCDC and BKDP did not really improve PDDP. Nevertheless, TFIDF scaling is less appropriate than norm scaling for clustering tasks on these sets [2].

4.5 Color clustering

The clustering methods described in this paper can also be employed to partition the colors in an image, with applications in image retrieval [18]. The task is to use a small number of colors to represent a given color image with thousands of colors or more. Each pixel is regarded as a data point in three-dimensional space representing the RGB color components, and therefore the clustering algorithms can be applied. Each image can be regarded as an $n_r \times n_c \times 3$ tensor. Replacing each pixel by the mean (centroid) of its cluster, we obtained the image with reduced colors. We measured the performance by *root mean squared errors* (RMSE):

$$RMSE = \sqrt{\frac{1}{3n_r n_c}\|M - \hat{M}\|_F^2},$$

where M and $\hat{M}$ are the original image and the image with reduced colors, respectively.

We report the result of an experiment on two photos, `shadows` and `pumpkins`, both of size 256-by-256. A few other photos were also tested and the conclusion is consistent. Relatively, the photo `shadows` has high color saturation, and the photo `pumpkins` has rich colors. For each photo, we partitioned the pixels into 4, 16 and 64 color clusters by seven methods: PDDP, BKDP, FCDC, and K-means initialized randomly, by PDDP, by BKDP,

and by FCDC. For FCDC we set the marginal region to be of size 10% of the total pixels for single moves of pixels. The root mean squared errors (RMSE) and time used are reported in Table 8.

Table 8: Root mean squared errors (RMSE) and CPU time (seconds) of color clustering.

Photo	# of colors	Measure	*PDDP*	*BKDP*	*FCDC*	*K-means*	*PDDP+ K-means*	*BKDP+ K-means*	*FCDC+ K-means*
shadows	4	RMSE	0.093	0.091	0.091	0.085	0.085	0.085	0.085
		Time	0.13	39.50	0.45	43.27	38.32	83.07	45.05
	16	RMSE	0.047	0.045	0.046	0.042	0.042	0.042	0.042
		Time	0.30	65.53	0.96	1565.79	435.79	669.27	633.17
	64	RMSE	0.027	0.026	0.027	0.025	0.024	0.024	0.024
		Time	0.61	86.86	1.81	9219.22	2186.07	2627.89	3472.56
pumpkins	4	RMSE	0.095	0.095	0.095	0.095	0.095	0.095	0.095
		Time	0.14	23.79	0.67	110.17	81.95	84.60	51.56
	16	RMSE	0.053	0.053	0.053	0.050	0.050	0.050	0.050
		Time	0.30	51.01	1.25	639.23	441.79	1108.10	1059.82
	64	RMSE	0.030	0.030	0.030	0.028	0.028	0.028	0.028
		Time	0.63	68.59	2.09	4876.71	5789.18	8194.68	6005.59

In terms of the resulting quality of clusters, there was no significant difference between PDDP, BKDP, and FCDC, whereas K-means worked better in terms of RMSE. This may be because the squared RMSE is proportional to the quantization error defined in (10), which the K-means algorithm aims to minimize.

Computationally, PDDP was the most economical, FCDC needed some more time, and BKDP took much more time. K-means was most expensive in this experiment. The cost of K-means increases sharply as the number of colors increases. The images with reduced colors by FCDC are shown in Figure 11.

5 Conclusion

This paper described an unsupervised clustering method named FCDC which is based on a recursive bisection approach. The performance of the algorithm was illustrated on various applications and the tests reveal that good improvements over spectral techniques can be obtained, albeit at a higher cost. An appealing use of FCDC is for initializing the K-means algorithm. Tests showed a marked improvement of the K-means algorithm when it is initialized with FCDC. There are several possible improvements to the algorithm which were not explored in this paper. One of these is to automate the choice of the optimal sizes of the marginal regions from which entries are moved in order to improve the distance between centroids. Another interesting extension is to use kernels to redefine distances and yield nonlinear versions of the algorithm. (See [1] for an example of kernel-based clustering methods.) This is possible because the tuning phase of FCDC relies on distances and inner products which can be modified by resorting to kernels.

Figure 11: Results of color clustering by FCDC. First column: raw image; second column: 64 colors; third column: 16 colors; last column: 4 colors.

Acknowledgments

We are indebted to Dan Boley for making his code and the test data used in [2] available to us. We would also like to thank Sofia Sakellaridi for her assistance with the document clustering experiment, and Efi Kokiopoulou who gathered and processed a few of the data sets used in this paper.

References

[1] A. Abraham, S. Das, and A. Konar. Kernel based automatic clustering using modified particle swarm optimization algorithm. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 2–9, 2007.

[2] D. L. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4), 1998.

[3] L. Bottou and Y. Bengio. Convergence properties of the K-means algorithms. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 585–592. The MIT Press, 1995.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.

[5] D. B. Graham and N. M. Allinson. Face recognition: From theory to applications. In H. Wechsler, P. J. Phillips, V. Bruce, F. Fogelman-Soulie, and T. S. Huang, editors, *NATO ASI Series F, Computer and Systems Sciences, Vol. 163*, pages 446–456, 1998.

[6] F. Juhász and K. Mályusz. Problems of cluster analysis from the viewpoint of numerical analysis. In *Colloquia Mathematica Societatis Janos Bolyai*, volume 22, pages 405–415, North-Holland, Amsterdam, 1980.

[7] G. Karypis and V. Kumar. Analysis of multilevel graph partitioning. In *Supercomputing '95: Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM)*, page 29, New York, NY, USA, 1995. ACM Press.

[8] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. In *Bell System Technical I.*, volume 29, pages 291–307, 1970.

[9] S. Lin. Computer solutions of the traveling salesman problem. In *Bell System Tech. J.*, volume 44, pages 2245–2269, 1965.

[10] A. Paccanaro, J. A. Casbon, and M. A. S. Saqi. Spectral clustering of protein sequences. *Nucleic Acids Research*, 34(5):1571–1580, 2006.

[11] A. Pothen, H. D. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. and Appl.*, 11(3):430–452, 1990.

[12] C. Rosenberger and K. Chehdi. Unsupervised clustering method with optimal estimation of the number of clusters: Application to image segmentation. In *15th International Conference on Pattern Recognition (ICPR'00)*, volume 1, page 1656, 2000.

[13] F. S. Samaria and A. C. Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of 2nd IEEE Workshop on Applications of Computer Vision*, pages 138–142, 1994.

[14] B. Le Saux and N. Boujemaa. Unsupervised robust clustering for image database categorization. In *IEEE-IAPR International Conference on Pattern Recognition (ICPR'2002)*, August 2002.

[15] S. Savaresi, D. L. Boley, S. Bittanti, and G. Gazzaniga. Choosing the cluster to split in bisecting divisive clustering algorithms. In *Second SIAM International Conference on Data Mining (SDM'2002)*, 2002.

[16] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley and Sons, 1998.

[17] D. Tritchler, S. Fallah, and J. Beyene. A spectral clustering method for microarray data. *Comput. Stat. & Data Anal.*, 49:63–76, 2005.

[18] J. Wang, W. Yang, and R. Acharya. Color clustering techniques for color-content-based image retrieval from image databases. In *International Conference on Multimedia Computing and Systems (ICMCS'97)*, pages 442–450, 1997.