In Search of Deterministic Methods for Initializing K-Means and Gaussian Mixture Clustering

Ting Su*

Jennifer G. Dy

Department of Electrical and Computer Engineering

Northeastern University, Boston, MA 02115

Tel:617-373-3975

Fax:617-373-8970

{tsu,jdy}@ece.neu.edu

September 24, 2006

Abstract

The performance of K-means and Gaussian mixture model (GMM) clustering depends on the initial guess of partitions. Typically, clus-

*corresponding author

tering algorithms are initialized by random starts. In our search for a deterministic method, we found two promising approaches: principal component analysis (PCA) partitioning and Var-Part (Variance Partitioning). K-means clustering tries to minimize the sum-squared-error criterion. The largest eigenvector with the largest eigenvalue is the component which contributes to the largest sum-squared-error. Hence, a good candidate direction to project a cluster for splitting is the direction of the cluster's largest eigenvector, the basis for PCA partitioning. Similarly, GMM clustering maximizes the likelihood; minimizing the determinant of the covariance matrices of each cluster helps to increase the likelihood. The largest eigenvector contributes to the largest determinant and is thus a good candidate direction for splitting. However, PCA is computationally expensive. We, thus, introduce Var-Part, which is computationally less complex (with complexity equal to one K-means iteration) and approximates PCA partitioning assuming diagonal covariance matrix. Experiments reveal that Var-Part has similar performance with PCA partitioning, sometimes better, and leads K-means (and GMM) to yield sum-squared-error (and maximum-likelihood) values close to the optimum values obtained by several random-start runs and often at faster convergence rates.

Keywords: K-Means, Gaussian mixture, Initialization, PCA, Clustering

1 Introduction

Cluster analysis is the unsupervised classification of objects into similar groupings. It is useful in several exploratory pattern analysis, data mining, decision-making, machine-learning, vector quantization, and compression situations [3, 17].

One of the most commonly used clustering algorithm is the K-means algorithm [11, 22, 1]. Another is maximum likelihood (ML) through expectation maximization [7] of a finite mixture of multi-variate Gaussian density models [23]. The K-means algorithm is an iterative algorithm for minimizing the sum of distance between each data point and its cluster center (centroid). There are various ways for measuring distance (e.g., squared Euclidean distance, city-block, hamming distance, cosine dissimilarity). Among these distance measures, squared Euclidean distance is the most widely used distance measure for K-means, and the corresponding criterion function that a Euclidean K-means algorithm optimizes is the sum-squared-error (SSE) criterion [8]. In this paper, we focus on K-means with the SSE criterion. Gaussian mixture modeling through expectation maximization is also an iterative algorithm for maximizing the likelihood. K-means clustering and Gaussian mixture clustering may not converge to the global optimum. The performance of K-means and Gaussian mixture clustering strongly depends on the initial starting points.

Several random initialization methods for K-means have been developed.

Two classical methods are random seed [11, 1] and random partition [1]. Random seed randomly selects K instances (seed points), and assigns each of the other instances to the cluster with the nearest seed point. Random partition assigns each data instance into one of the K clusters randomly. To escape from getting stuck at a local minimum, one can apply r random starts. Specifically, one can perform one of the above methods to initialize K-means, repeat the process r times, and select the final clustering with the minimum SSE from the r runs. To cope with large data sets, [5] and [10] introduced a sub-sampling version of random restart. The problem with random methods is that they are not repeatable (unless one stores all the starting points applied or the seeds of the random-number generator), and they may still lead to a solution with bad quality unless we allow r to be very large (thereby, making the clustering time-consuming for large data sets).

In this paper, we search for deterministic techniques that can compete with classical random methods. We pick three candidates for this investigation: KKZ [20], principal components analysis (PCA) based partitioning, and Var-Part (variance partitioning). We choose KKZ in this study because a recent comparative study [15] claims that KKZ is the best method among the methods compared in that paper; PCA based partitioning because we believe that it will provide an initial guess in the vicinity of the optimum solution; and, Var-Part because PCA can be computationally expensive and Var-Part serves as a faster alternative to PCA partitioning.

KKZ was introduced by [20] for initializing vector quantization. A PCA

based divisive hierarchical approach was introduced by two sources: [16] and [4]. This approach was called directed-search binary-splitting (DSBS) and was applied for initializing code-vectors in [16]. In [4], it is called Principal Direction Divisive Partitioning (PDDP) and was introduced as a clustering algorithm for partitioning document data. Both DSBS and PDDP are similar. We will call it PCA partitioning throughout this paper.

The contributions of this paper are:

1. Perform a comparative study among three deterministic initialization methods, and between deterministic versus random techniques.

2. Provide the motivation on why PCA based methods are good for initializing K-means and Gaussian mixture clustering, and also to present their limitations.

3. Introduce Var-part initialization method, which has similar performance with PCA partitioning and a time complexity equal to one K-means iteration.

4. Find a reasonable deterministic initialization method for clustering.

In Section 2, we describe K-means and Gaussian mixture clustering, and at the same time define the notations for this paper. In Section 3.2, we describe the motivation for PCA partitioning for initializing K-means, and in Section 3.3, we present Var-Part, a faster approximation to PCA partitioning. In Section 4, we illustrate when PCA partitioning would fail and suggest a

possible extension, PCA-Part*. In Section 5, we show the motivation for PCA partitioning for initializing Gaussian mixture clustering. We provide a review of related work in Section 6. We, then, report our comparative study in Section 7. Finally, in Section 8 we summarize the paper, draw conclusions and suggest avenues for future research.

2 The Clustering Algorithms and Notations

We denote our data set as $X = \{x_1, x_2, \ldots, x_N\}$. X consists of N data instances $x_i (i = 1, 2, \ldots, N)$, and each x_i represents a single d-dimensional instance.

2.1 The K-means Algorithm

The goal of K-means is to partition X into K clusters $\{C_1, \ldots, C_K\}$. The most widely used criterion function for the K-means algorithm is the sum-squared-error (SSE) criterion [8]. Let n_j denote the number of instances in cluster C_j, and let μ_j denote the mean (centroid) of those instances in C_j. Then μ_j is defined as:

$$\mu_j = \frac{1}{n_j} \sum_{x_i \in C_j} x_i \tag{1}$$

And, SSE is defined as:

$$SSE = \sum_{j=1}^{K} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2 \tag{2}$$

K-means is an iterative algorithm that minimizes the SSE criterion. "K-means" denotes the process of assigning each data point, x_i, to the cluster with the nearest mean. The K-means algorithm starts with initial K centroids, then it assigns each remaining point to the nearest centroid, updates the cluster centroids, and repeats the process until the K centroids do not change (convergence). There are two versions of K-means, one version originates from Forgy [11] and the other version from Macqueen [22]. The difference between the two is on when to update the cluster centroids. In Forgy's K-means [11], cluster centroids are re-computed after all the data points have been assigned to their nearest centroids. In Macqueen's K-means [22], the cluster centroids are re-computed after each data assignment. In this paper, we report the results using Forgy's K-means. We obtained similar results for Macqueen's version.

2.2 Gaussian Mixture Clustering

Clustering using finite mixture models is thoroughly described in [23]. In this model, one assumes that data is generated from a mixture of K component density functions, in which $p(x_i|\theta_j)$ represents the density function of component j for all $j's$, where θ_j is the parameter (to be estimated) for cluster j. The probability density of data x_i, is expressed by:

$$p(x_i) = \sum_{j=1}^{K} \alpha_j p(x_i|\theta_j) \tag{3}$$

where the $\alpha's$ are the mixing proportions of the components (subject to: $\alpha_j \geq 0$ and $\sum_{j=1}^{K} \alpha_j = 1$). The log-likelihood of the N observed data points is then given by:

$$\mathcal{L} = \sum_{i=1}^{N} \ln\{\sum_{j=1}^{K} \alpha_j p(x_i|\theta_j)\} \tag{4}$$

It is difficult to directly optimize (4), therefore we apply the Expectation-Maximization (EM) [7] algorithm to find a (local) maximum likelihood or maximum a posteriori (MAP) estimate of the parameters for the given data set. The EM algorithm iterates between an Estimation-step and a Maximization-step until convergence. Throughout this paper, we assume all the variables are continuous and each mixture has a Gaussian distribution:

$$p(x_i|\theta_j) = \frac{1}{(2\pi)^{\frac{d}{2}}|S_j|^{\frac{1}{2}}} e^{-\frac{1}{2}(x_i - m_j)^\top S_j^{-1}(x_i - m_j)} \tag{5}$$

where m_j and S_j are the mean and the covariance matrix of mixture j, respectively. $|\cdot|$ is the determinant operator and $\top$ means the transpose of a matrix.

3 Three Deterministic Initialization Methods

In this section, we describe KKZ, PCA based partitioning, and variance based partitioning.

3.1 KKZ

KKZ is proposed by [20]. The idea behind KKZ is to pay attention to the data points that are most far apart from each other, since those data points are more likely to belong to different clusters. The pseudo-code for KKZ is as follows:

1. Choose the point with the maximum $L2$-norm as the first centroid.

2. For $j = 2, \ldots, K$, each centroid μ_j is set in the following way: For any remaining data x_i, we compute its distance d_i to the existing centroids. d_i is calculated as the distance between x_i to its closest existing centroid. Then, the point with the largest d_i is selected as μ_j.

The computational complexity of KKZ is (NKd).

3.2 PCA Based Partitioning Initialization Method

In this subsection, we present a theoretical analysis behind PCA-based approaches. In the process, we describe and motivate a particular version, PCA-Part, which has the same structure as Var-Part.

Projecting data instances on a single direction and then performing the initial partition on that direction was proposed in [1]. This partitions data only on one dimension. An alternative method of dividing the sample space is to partition it hierarchically. Starting with one cluster, cut it in half. Pick the next cluster to partition, and repeat the process until K clusters are

obtained. Figures 1 and 2 illustrate these two methods of cutting the space. PCA-Part partitions data using the latter approach. For future work, one may wish to explore other ways of cutting the "pie" such as in a "radial" fashion.

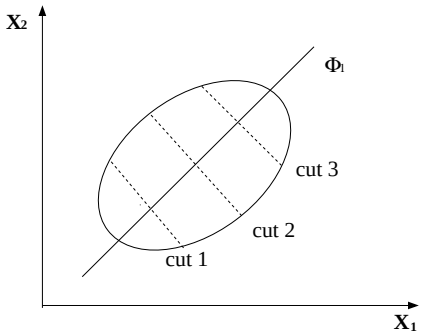

Figure 1: Sample Covariance of the data in two dimensions, X_1 and X_2. Cuts 1, 2 and 3 partition the data on one direction (axis ϕ_1) into $k = 4$ clusters.

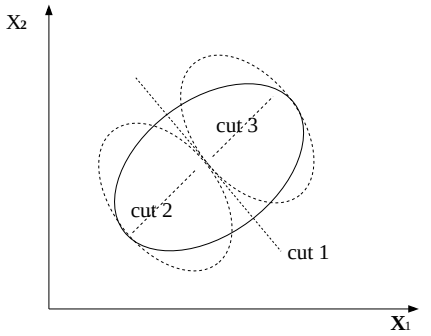

Figure 2: Sample Covariance of the same data in two dimensions, X_1 and X_2. Cuts 1, 2 and 3 partition the data into $k = 4$ clusters hierarchically.

Now, which direction should we split the chosen cluster? Let μ be the

mean for a given cluster. The SSE of the data within this cluster C is

$$SSE = \sum_{x_i \in C} \|x_i - \mu\|^2.$$

After dividing this cluster into two clusters, C_1 with mean μ_1 and C_2 with mean μ_2, the new SSE is

$$SSE_{new} = \sum_{x_i \in C_1} \|x_i - \mu_1\|^2 + \sum_{x_i \in C_2} \|x_i - \mu_2\|^2.$$

Each d-dimensional vector x_i can be represented by a weighted sum of d linearly independent orthonormal basis vectors, $\Phi = [\phi_1, \ldots, \phi_d]$:

$$x_i = \sum_{s=1}^{d} y_{is}\phi_s.$$

Similarly, the mean μ_j can be represented as:

$$\mu_j = \sum_{s=1}^{d} \alpha_{js}\phi_s.$$

We restate our question as, which direction ϕ_p should we project our data for splitting? Assuming that the old mean μ and the new means, μ_1 and μ_2 lie on the axis chosen for projecting, now we show that the ϕ_p which minimizes SSE_{new} is the ϕ_p that maximizes

$$\sum_{x_i \in C} (y_{ip}\phi_p - \alpha_p\phi_p)^2 \quad - \quad \sum_{x_i \in C_1} (y_{ip}\phi_p - \alpha_{1p}\phi_p)^2$$

$$- \sum_{x_i \in C_2} (y_{ip}\phi_p - \alpha_{2p}\phi_p)^2 \qquad (6)$$

where y_{ip}, α_p, α_{1p}, and α_{2p} correspond to the projected value of x_i, μ, μ_1, and μ_2 respectively on the direction ϕ_p.

Equation 6 is SSE due to the component ϕ_p without splitting minus SSE due to the component ϕ_p after splitting.

Proof:

The old SSE can be expressed as:

$$SSE_{old} = \sum_{x_i \in C} \| \sum_{s=1}^{d} y_{is}\phi_s - \sum_{s=1}^{d} \alpha_s \phi_s \|^2$$

Due to the orthonormality assumption among ϕ_s's,

$$SSE_{old} = \sum_{x_i \in C} \| \sum_{s=1, s\neq p}^{d} y_{is}\phi_s - \sum_{s=1, s\neq p}^{d} \alpha_s \phi_s \|^2 + \sum_{x_i \in C} (y_{ip}\phi_p - \alpha_p \phi_p)^2$$

Since the old mean lies on the axis chosen for projecting, ϕ_p,

$$SSE_{old} = \sum_{x_i \in C} \| \sum_{s=1, s\neq p}^{d} y_{is}\phi_s \|^2 + \sum_{x_i \in C} (y_{ip}\phi_p - \alpha_p \phi_p)^2$$

And, the new SSE can be expressed as:

$$\begin{aligned} SSE_{new} &= \sum_{x_i \in C_1} \| \sum_{s=1}^{d} y_{is}\phi_s - \sum_{s=1}^{d} \alpha_{1s}\phi_s \|^2 \\ &\quad + \sum_{x_i \in C_2} \| \sum_{s=1}^{d} y_{is}\phi_s - \sum_{s=1}^{d} \alpha_{2s}\phi_s \|^2 \end{aligned}$$

Due to the orthonormality assumption among ϕ_s's,

$$\begin{aligned} SSE_{new} &= \sum_{x_i \in C_1} \| \sum_{s=1, s\neq p}^{d} y_{is}\phi_s - \sum_{s=1, s\neq p}^{d} \alpha_{1s}\phi_s \|^2 \\ &\quad + \sum_{x_i \in C_1} (y_{ip}\phi_p - \alpha_{1p}\phi_p)^2 \\ &\quad + \sum_{x_i \in C_2} \| \sum_{s=1, s\neq p}^{d} y_{is}\phi_s - \sum_{s=1, s\neq p}^{d} \alpha_{2s}\phi_s \|^2 \\ &\quad + \sum_{x_i \in C_2} (y_{ip}\phi_p - \alpha_{2p}\phi_p)^2 \end{aligned}$$

since the new means lie on the axis chosen for projecting, ϕ_p,

$$\begin{aligned} SSE_{new} &= \sum_{x_i \in C_1} \| \sum_{s=1, s\neq p}^{d} y_{is}\phi_s \|^2 + \sum_{x_i \in C_1} (y_{ip}\phi_p - \alpha_{1p}\phi_p)^2 \\ &\quad + \sum_{x_i \in C_2} \| \sum_{s=1, s\neq p}^{d} y_{is}\phi_s \|^2 + \sum_{x_i \in C_2} (y_{ip}\phi_p - \alpha_{2p}\phi_{2p})^2 \\ &= \sum_{x_i \in C} \| \sum_{s=1, s\neq p}^{d} y_{is}\phi_s \|^2 \\ &\quad + \sum_{x_i \in C_1} (y_{ip}\phi_p - \alpha_{1p}\phi_p)^2 \\ &\quad + \sum_{x_i \in C_2} (y_{ip}\phi_p - \alpha_{1p}\phi_p)^2 \end{aligned}$$

Since $\sum_{x_i \in C_1} (y_{ip}\phi_p - \alpha_{1p}\phi_p)^2 + \sum_{x_i \in C_2} (y_{ip}\phi_p - \alpha_{2p}\phi_p)^2 \leq \sum_{x_i \in C} (y_{ip}\phi_p -$

$\alpha_p\phi_p)^2$, we have $SSE_{new} \leq SSE_{old}$. The ϕ_p which minimizes SSE_{new} is equivalent to the ϕ_p that maximizes SSE_{old} minus SSE_{new}.

To find this optimal direction, we need to know the means, μ_1 and μ_2. This leads us back to a $K = 2$ clustering problem for minimizing SSE. To avoid solving a clustering problem, PCA-Part resorts to a suboptimal direction which assumes that the SSE_{new} component due to the candidate direction, $\sum_{y_i \in C_1}(y_{ip}\phi_p - \alpha_{1p}\phi_p)^2 + \sum_{y_i \in C_2}(y_{ip}\phi_p - \alpha_{2p}\phi_p)^2$, is proportional to the SSE_{old} due to this direction, $\sum_{y_i \in C}(y_{ip}\phi_p - \alpha_p\phi_p)^2$, and this proportionality constant, a, is the same for all directions and $0 \leq a \leq 1$. The optimization problem is now simplified to finding the direction, ϕ_p, that maximizes $\sum_{y_i \in C}(y_{ip}\phi_p - \alpha_p\phi_p)^2$.

Thus, PCA-Part chooses ϕ_p to be the component which contributes to the largest SSE. The largest eigenvector of the covariance matrix, is the direction which contributes to the largest SSE [12]. Hence, PCA-Part picks the largest eigenvector of the covariance matrix as the direction for projecting.

We still need to determine how to partition the cluster in this principal direction. One method is to pick two data points with the maximum and minimum value in the projected axis, then grow the seeds from these two points (i.e., assign each data point to the seed closest to it). This type of partitioning is similar to the method used in KKZ [20]. In KKZ, the first seed is chosen as the data point with maximum norm, and the second seed is the data instance farthest from the first centroid. The problem with these partitioning methods is that they are sensitive to outliers as shown in Figure

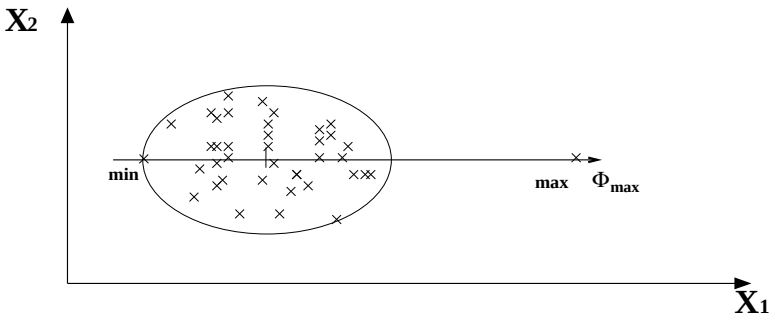

Figure 3: Scatter-plot of a two-dimensional data, X_1 and X_2. The minimum and maximum data points (min and max respectively) on the projected axis, Φ_{max}, are shown.

3, which might lead to two clusters, one consisting of just the maximum data point and the second consisting of the rest of the data.

An alternative is to partition the data at the mean (middle). This way, the center of gravity between the two halves will be balanced at the mean.

Let us assume that we need $K = 3$ clusters. After splitting the data into two, which of those two clusters should we split next? There are several different ways of determining which cluster to split. For example, we can pick the cluster with the largest number of instances, or pick the cluster with the highest variance. Since SSE is the criterion function K-means wishes to minimize, we decide to split the cluster with the largest with-in cluster $SSE_j = \sum_{x_i \in C_j} \|x_i - \mu_j\|^2$.

We now give a summary for the PCA-Part initialization method.

Starting from a single cluster, divide this cluster into two, choose the next cluster C_j to partition by selecting the cluster with the largest within-cluster

SSE_j, repeat the process until K clusters are produced.

For the selected cluster C_j,

Step 1: Project $x_i \in C_j$ to the largest principal component axis of $x_i \in C_j$. y_i is the projected version of x_i in this axis.

Step 2: Divide C_j into two sub-clusters C_{j1} and C_{j2}, according to the following rule: For any x_i, if $y_i \leq \alpha_j$, assign x_i to C_{j1}, else, assign x_i to C_{j2}. α_j is the projected version of the mean μ_j of cluster C_j.

Figure 4: PCA-Part

Figure 4 shows the pseudocode for PCA-Part. This PCA based partitioning algorithm that we developed above for initialization ends up to be similar to the Principal Direction Divisive Partitioning (PDDP) hierarchical clustering algorithm introduced by Boley [4] for partitioning document data, and to the directed-search binary-splitting (DSBS) applied by [16] for generating code-vectors. The main difference is that in DSBS, they only consider the data points which are inside two standard deviations away from the mean to exclude some possible outliers.

In each stage, PCA-Part bisects the data in Euclidean space by a hyperplane which passes through the centroid of the selected cluster, orthogonal to the largest eigenvector. The most popular technique for finding all the eigenvectors of a real, symmetric matrix is to first transform the matrix into a tridiagonal matrix using Householder reduction, and then to use the QL algorithm to determine the eigenvectors and the eigenvalues. For a d by d symmetric matrix, the time complexity for the Householder-QL method

is $O(d^3)$ [30]. Instead of directly computing the principal components, one may use Singular Value Decomposition (SVD). Boley in [4] applied Lanczos method [13] for computing SVD, taking advantage of the sparsity of document data. However, the time complexity for computing SVD for a dense matrix is still costly. Another alternative is to use the power method [32], which is utilized by DSBS, to perform PCA. the power method is an iterative method for computing the q ($q << d$) largest eigenvectors at $O(qd^2)$ per iteration. For PCA-Part, we have $q = 1$, since we only need the largest eigenvector. Since the time complexity for computing a covariance matrix is $O(nd^2)$, and we need $K-1$ stages to divide the data into K clusters, the time complexity to perform PCA-Part using Householder-QL is $O(nd^2K + d^3K)$ and using the power method is $O(nd^2K)$.

In the following section, we describe another bisecting divisive initialization method, which approximates PCA and requires much less computation.

3.3 Variance Partitioning Initialization Method

For the selected cluster C_j,

Step 1: Compute the variance in each dimension, find the dimension with the largest variance, say d_p;

Step 2: Let x_{ip} denote the value of instance x_i in feature d_p, μ_{jp} denote the mean of C_j in feature d_p, divide C_j into two sub-clusters C_{j1} and C_{j2}, according to the following rule: If $x_{ip} \leq \mu_{jp}$, assign x_i to C_{j1}; otherwise, assign x_i to C_{j2}.

Figure 5: Var-Part

Var-Part (Variance Partitioning) assumes that the covariance matrix is diagonal, which is not true in general, but is a widely used assumption for clustering. In each cutting stage, Var-Part bisects the data in Euclidean space with a hyperplane which passes through the centroid of the selected cluster, orthogonal to the feature (dimension) with the largest variance. Var-Part results in axis-parallel cuts similar to decision trees [31]. Figure 5 shows the pseudocode for Var-Part.

The time complexity for Var-Part is $O(nd)$ in each stage, and since we need $K-1$ stages to divide the data set into K clusters, the total time complexity is $O(ndK)$. The time complexity for one iteration of K-means clustering is $O(ndK)$. Thus, Var-Part as an initialization method is equivalent to running one K-means iteration. The significance of this observation will become apparent in the experimental discussion of the speed of convergence in Section 7.3.2 and 7.4.2.

4 An Extension, PCA-Part*

Note that the largest principal component solution only maximizes the first term in Equation 6, Section 3.2, SSE due to component ϕ_p before partitioning, and did not take into account the effect of the second and third terms, SSE due to component ϕ_p after splitting. Although this is a reasonable direction to project the data for splitting, this solution is not optimal for data sets which are well-separated in directions different from the largest eigenvec-

tor (See Figure 6.). Equation 6 suggests a possible extension to PCA-Part, PCA-Part*: Choose the PCA component which maximizes Equation 6 as the direction to project the data for splitting. This solution is still suboptimal as we search only among PCA components and not the infinite space of possible directions. Moreover, PCA-Part* is more computationally complex compared to PCA-Part by a factor d (the number of dimensions). It would take a long time to initialize K-means using PCA-Part* when d is large. Hence, in this paper, we focus on the basic and simpler methods: PCA-Part and Var-Part.

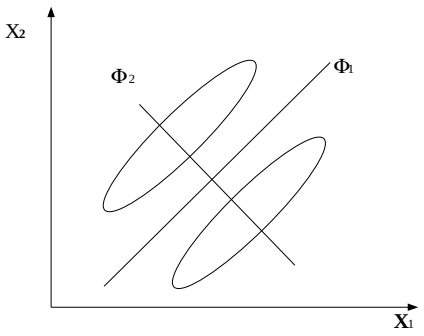

Figure 6: Here is a cluster configuration in which the largest principal component does not provide the best projection for splitting.

5 Initializing Gaussian Mixture Clustering

In this section, we motivate the use of PCA-Part and Var-Part for initializing Gaussian mixture clustering. The maximum likelihood criterion is the

objective function that a finite mixture of Gaussians tries to optimize. As we mentioned before, the density function of the data set X, can be expressed as:

$$P(X) = \prod_{i=1}^{N} \sum_{j=1}^{K} \alpha_j \frac{1}{(2\pi)^{\frac{d}{2}} |S_j|^{\frac{1}{2}}} e^{-\frac{1}{2}(x_i - m_j)^\top S_j^{-1}(x_i - m_j)}.$$

If we assume "hard" clustering (i.e., a data point can belong to only one cluster), this is the assumption made in K-means and in both PCA-Part and Var-Part, the likelihood can be simplified as:

$$P(X) = \prod_{j=1}^{K} \prod_{x_i \in j} \alpha_j \frac{1}{(2\pi)^{\frac{d}{2}} |S_j|^{\frac{1}{2}}} e^{-\frac{1}{2}(x_i - m_j)^\top S_j^{-1}(x_i - m_j)}.$$

The parameters that maximize the likelihood are the same as that which maximize the log-likelihood. We take the log and get:

$$\begin{aligned}
\mathcal{L} = \ln P(X) &= \sum_{j=1}^{K} \sum_{x_i \in j} (\ln \alpha_j - \frac{d}{2} \ln(2\pi) - \frac{1}{2} \ln |S_j| - \frac{1}{2}(x_i - m_j)^\top S_j^{-1}(x_i - m_j)) \\
&= \sum_{j=1}^{K} \{n_j \ln \alpha_j - \frac{dn_j}{2} \ln(2\pi) - \frac{n_j}{2} \ln |S_j| - \frac{1}{2} \sum_{x_i \in j} trace(S_j^{-1}(x_i - m_j)(x_i - m_j)^\top)\} \\
&= \sum_{j=1}^{K} \{n_j \ln \alpha_j - \frac{dn_j}{2} \ln(2\pi) - \frac{n_j}{2} \ln |S_j| - \frac{1}{2} trace(S_j^{-1} \sum_{x_i \in j} (x_i - m_j)(x_i - m_j)^\top)\} \\
&= \sum_{j=1}^{K} \{n_j \ln \alpha_j - \frac{dn_j}{2} \ln(2\pi) - \frac{n_j}{2} \ln |S_j| - \frac{1}{2} trace(S_j^{-1} S_j n_j)\} \\
&= \sum_{j=1}^{K} \{n_j \ln \frac{n_j}{N} - \frac{dn_j}{2} \ln(2\pi) - \frac{n_j}{2} \ln |S_j| - \frac{dn_j}{2}\} \qquad (7)
\end{aligned}$$

We need to answer two questions using Equation 7. First, which cluster

should we split? Since our goal is to maximize the likelihood, we should split the cluster giving the least likelihood (i.e, we can split the cluster giving the least value of $n_j \ln \frac{n_j}{N} - \frac{dn_j}{2} \ln(2\pi) - \frac{n_j}{2} \ln |S_j| - \frac{dn_j}{2})$. However, it will involve computing the determinant of the covariance matrix for every component. To save computation time, we suggest that we split the cluster with the largest with-in sum-squared-error $SSE_j = \sum_{x_i \in S_j} \|x_i - \mu_j\|^2$.

Second, for a chosen cluster j, which direction should we split? Observe that the likelihood increases as $|S_j|$ decreases. Thus, a good candidate direction for splitting the data would be the direction that contributes the most to $|S_j|$, which is the eigenvector associated with the largest eigenvalue, since $|S|$ is the product of all the d eigenvalues [18]. It makes sense to remove or decrease this largest variance. In the case of Var-Part, it will be the feature with the largest variance.

6 Related work

Al-Daoud et al. [27] present two "one-run" initialization methods for K-means: AD1 and AD2, which are designed for clustering when the number of clusters K is large. Both AD1 and AD2 require the division of the data space into several subspaces, then they determine the number of clusters in each subspace according to the density of data points in that subspace. Although AD1 and AD2 are "one-run" initialization methods, they have some random components within their algorithm which may lead to non-repeatable clus-

tering solutions. In addition, it is hard to decide the appropriate number of subspaces for AD1. Another common initialization method, random perturbation, is to randomly perturb the mean of the entire data K times, which does not appear to be better than random seed [15]. Some deterministic methods have been introduced in the literature. One is the KA algorithm [21]. KA chooses the most centrally located instances as the first seed, then selects the next seed according to the heuristic rule of choosing the point which has more of the remaining instances around it. KA repeats this process until K seeds are found. A recent study [15] showed that usually KKZ gets better performance than KA. Besides, KA is unfavorable for large data sets, because of the $O(n^2dk)$ time complexity since the distances between each pair of instances are needed. Simple Cluster Seeking (SCS) is another deterministic method [19], it takes the new incoming data input as a new initial centroid as long as the new input is far from all the existing centroids than a predefined threshold. It is difficult to decide the threshold for SCS, besides the performance of SCS depends on the order of the data. Other deterministic methods utilize hierarchical clustering as an initialization method for K-means. A hierarchical clustering method can often produce an excellent initial partition for the K-means algorithm [1, 14]. For example, Milligan [26] claimed that using Ward's agglomerative hierarchical method [34] as the initialization method for the K-means algorithm could yield good final clusterings. The majority of the existing hierarchical initialization methods for the K-means algorithm use agglomerative (bottom-up) approach. Generally,

the time complexity of a hierarchical agglomerative method is $O(n^2 \log n)$ [17] and it needs $O(n^2)$ memory space to store the distances between each pair of instances [17]. There have been a few comparative study on initialization methods. The most recent comparative study [15] compared random seed, random perturbation, KKZ, KA and SCS for initializing K-means. KA, random partition, random seed for initializing K-means was compared on three small data sets in [28]. Another work[24] compared three initialization methods: parameters sampled from an uninformative prior, random perturbation of the marginal distribution of the data and a hierarchical agglomerative initialization method (HAC) for initializing multinomial mixture clustering on discrete data, and have shown that these three methods obtain comparable performance, except for HAC which has a longer running time.

7 Experiments

In this section, we perform a comparative study among three deterministic initialization methods, PCA-Part (PCA-P), Var-Part (Var-P), and KKZ, and investigate their performance compared to the classical random seed method (Rand-S) and the faster sub-sampling plus refinement (Refine) random methods [5, 10] for initializing K-means and Gaussian mixture clustering. The abbreviations in parenthesis are the ones we utilize to represent these methods in our tables.

Random Seed Method: Random seed method randomly selects the K ini-

tial cluster seeds, then runs K-means or Gaussian mixture clustering until convergence. We cannot run the clustering algorithms on all the possible initial conditions, therefore we sample the space of possible K partitions.

Refinement Method: A sub-sampling versions of random restart to cope with large data sets for K-means and Gaussian mixture clustering was introduced in [5] and[10] respectively. The refinement methods initialize K-means (or Gaussian mixture clustering) as follows: it chooses J random sub-samples of the data, $S_i, i = 1, ..., J$. The sub-samples are clustered via K-means (or Gaussian mixture clustering). If there exists empty clusters, the centroids of empty clusters will be reassigned to the data points which are the furthest from their centroids (or have the least likelihood given the current model), then re-run the K-means algorithm (or the Gaussian mixture clustering). Each sub-sample results in a set of cluster centroids $CM_i, i = 1, ...J$. The combined set CM, of all CM_i's, is clustered via K-means initialized with CM_i, resulting in new centroids FM_i. The refined initial centroids are then chosen as the FM_i with the minimal MSE (or the largest likelihood). In our experiments, we set J corresponding to 1% sub-sampling as suggested in [5] and[10] for the larger data sets, 5% for the smaller data sets with respect to the number of clusters (ionosphere and mfeat-mor) and 10% for the glass data.

Because random seed and refinement are random methods, in our experiments, we run them 100 times for the smaller data sets, 20 times for the Covtype data, and 50 times for the HRCT data, and show the maximum, minimum, and average values from these runs.

7.1 Performance Measures

The final result of K-means and Gaussian mixture clustering depends on the initial partition. We measure the performance of the initialization methods based on the following criteria:

Quality: Evaluating the quality of clustering results is not a trivial task. Choices for criteria could be internal criteria such as SSE, log-likelihood, scatter separability [12] or external criteria such as normal mutual information between clustering result and class labels [33]. Different criteria may give different quality values for the same clustering result. Since SSE and log-likelihood are measures that K-means and Gaussian mixtures try to optimize respectively, in this paper, we quantify the quality of the initialization methods based on the MSE (or the log-likelihood) values of the final clustering obtained by K-means (or Gaussian mixtures). MSE is just the normalized version of SSE, $MSE = SSE/N$, where N is the total number of samples. We report the average, maximum, and minimum of the MSE and the log-likelihood values.

Speed: We evaluate the speed of convergence through the number of itera-

tions needed for the clustering algorithms to converge and through the total time in seconds for the clustering to converge including the time to perform the initialization.

7.2 Data Sets

Table 1: Data set descriptions

Data set	# of Samples	# of Features	# of Classes
Glass	214	7	6
Segmentation	2310	16	7
Satellite image	6435	36	6
Letter	20000	16	26
Pen-digits	10992	16	10
HRCT	1545	156	8
Covtype	581012	10	7
Ionosphere	351	33	2
Mfeat-mor	2000	6	10

We run the experiments on nine data sets, six from the UCI Machine Learning repository [25] (glass, segmentation, satellite image, letter, pen-digits, ionosphere, Mfeat-mor), one from the UCI ADD repository [2] (covtype), and a lung image data (HRCT) [9]. Since HRCT has 156 dimensions, it will take a long time for Gaussian Mixture clustering to reach convergence. Hence, when we apply Gaussian Mixture clustering to the HRCT data, we project HRCT to twenty dimensions using random projection [6], we call this data HRCT20 thereafter. In all our experiments, we set the number of clusters equal to the number of classes. In our experiments, we remove features

whose variance are smaller than 0.01 to avoid singular covariances for the mixture of Gaussian clustering. Table 1 shows the number of data instances, the number of features (after removing the low-variance features), and the number of classes for these data sets.

7.3 Experimental Results on K-means

In this section, we compare the various initialization methods for initializing K-means.

7.3.1 The Quality of the Final Clustering

Table 2 shows the MSE of the final clustering obtained when the various initialization methods are applied. Here, small MSE values are desired. Boldface indicates the initialization method which led K-means to the minimal MSE for each data set (the average MSE of random methods are used for comparison, since for random methods, we run 20 times for covtype, 50 times for HRCT and 100 times for other data sets). PCA-Part and Var-Part have comparable MSE performance, and these two perform better than the other three methods in terms of MSE. Note that in most cases PCA-Part and Var-Part show similar performance. On four out of nine data sets (glass, segment, ionosphere and mfeat-mor), both PCA-Part and Var-Part perform best, and these two methods yield MSE values close to the minimum MSE returned by the random methods. Among the five remaining data sets, PCA-Part performs best on the letter data. Var-Part performs

best on the pen-digits and the HRCT data. The refinement method is better than random seed, it results in either smaller MSE values than random seed or similar MSE values as random seed. Also, note that the deterministic methods have zero standard deviation for MSE values, because as deterministic methods, they yield the same initial points for each run. KKZ performs the worst in terms of MSE values on most data sets. In addition, KKZ results in MSE value close to or even bigger than the maximal MSE returned by random methods on segment, satellite, HRCT and mfeat-mor data. A possible reason is that KKZ is sensitive to outliers similar to cases shown in Figure 3.

We thought, what if we normalize the features so that K-means, PCA-Part and Var-Part will treat the features with equal weight? We normalize the features by linearly scaling the features to be between 0 and 1. We only need to normalize the features of data sets that have features not on the same scale (i.e., all data sets except satellite, letter, pendigits, and ionosphere data). We chose this type of normalization rather than by standardizing the features to have variance equal to one, because that will make Var-Part ineffective.

In terms of SSE, Table 3 still shows that PCA-part and Var-part get better SSE performance than the other three methods: on the glass data, Var-Part performs best; on the segment data, PCA-Part is the winner; on the covtype and mfeat-mor data, PCA-Part and Var-Part perform best and have SSE values close to the minimum SSE obtained by the random methods.

Table 2: Mean square error of K-means clustering solutions for un_normalized data. Smaller values are desired. Boldface indicates the initialization method which led K-means to the minimal average MSE for each data set. Note that the average, maximal, and the minimal values of MSE for Rand-S and refine are taken from 20 runs for covtype, 50 runs for HRCT and 100 runs for other data sets.

Method	Glass	Segment	Satellite	Letter	Pen-digits	Hrct	Covtype	ionosphere	mfeat mor
Rand-S									
max	2.72	9187	2904.6	31.61	4909.34	161721	824889	9.18	187501
ave	1.84 ± 0.3	6609 ±1046	**2590.2** ±90	31.10 ± 0.2	4645.55 ±98	154237 ±3255	777951±16293	6.96 ± 0.4	149327 ± 5692
min	1.57	6041	2527.0	30.58	4485.26	149498	771659	6.89	147271
Refine									
max	2. 08	9113	2866.0	31.58	4787.64	164044	771984	9.18	168688
ave	1.67 ± 0.10	6358 ±876	2596.0 ±83	30.99 ± 0.2	4590.30 ±68	153700 ± 3335	**771769** ±87	6.98 ± 0.44	149539 ± 3360
min	1.56	5824	2527.05	30.64	4485.28	149539	771594	6.89	147167
KKZ	1.77 ± 0	10384± 0	2866.8 ± 0	31.35 ± 0	**4485.68** ± 0	163907 ± 0	771966 ± 0	**6.89**± 0	167093 ± 0
PCA-P	**1.57** ± 0	6010± 0	2653.8 ± 0	**30.90** ± 0	4552.88 ± 0	154655 ± 0	771804 ± 0	**6.89** ± 0	**147272** ± 0
Var-P	**1.57** ± 0	**6003** ± 0	2653.8 ± 0	31.21 ± 0	**4485.61** ± 0	**150206** ± 0	802568 ± 0	**6.89** ± 0	**147272** ± 0

Table 3 also shows that the refinement method is better than random seed and KKZ. When the features are at the same scale, they are weighted equally by the distance metric in K-means, and PCA and Var-Part captures the variance in the data based on its structure rather than on its scale. In the rest of this paper, we will only report the results for data with normalization if it is not on the same scale.

7.3.2 Speed of Convergence

In this section, we compare the various initialization methods based on the time it takes K-means to converge given the starting points provided by the various methods. Table 4 lists the average and the standard deviation of the number of iterations that K-means needs to reach convergence for the different initialization methods. The table shows that on most of the data

Table 3: Mean squared error of K-means clustering solutions for normalized data. Smaller values are desired. Boldface indicates the initialization method which led K-means to the minimum average MSE for each data set. Note that the average, maximal, and the minimal values of MSE for Rand-S and refine are taken from 20 runs for covtype, 50 runs for HRCT and 100 runs for other data sets.

Method		Glass	Segment	Hrct	Covtype	mfeat mor
Rand-S	max	17.82	435.45	4228.04	70576.40	93.98
	ave	14.11 ± 1.39	350.30 ±20.09	**4107.89** ±39.92	67255.8 ±965.95	44.57 ±16.32
	min	11.89	327.80	4042.21	66228.50	30.16
Refine	max	16.69	423.48	4271	67402.20	45.7
	ave	12.91 ± 1.15	349.89 ±17.73	4118 ±56.47	66409.1 ±347.19	32.07 ± 2.70
	min	11.49	327.80	4038	66226.00	30.16
KKZ		12.66 ± 0	390.72 ± 0	4185.74 ± 0	67328.5 ± 0	40.39 ± 0
PCA-P		12.56 ± 0	**345.37** ± 0	4114.79 ± 0	**66253.1** ± 0	**30.21** ± 0
Var-P		**12.09** ± 0	350.28 ± 0	4115.16 ± 0	**66255.0** ± 0	**30.25** ± 0

sets, Var-Part leads to the fewest number of iterations. PCA-Part, KKZ and the refinement methods have comparable performance in terms of the number of iterations needed by K-means, and they have better performance compared to random seed. Note that, as a random method, refinement can take as long as random seed in its worst case.

Although the time complexity of Var-Part is $O(ndK)$, which is more than the time complexity of random seed, observe that in most cases, using Var-Part to initialize the K-means algorithm needs less iterations to converge than using random seed. The time complexity of one iteration in K-means algorithm is $O(ndK)$. Therefore, we can conclude that generally using Var-Part to initialize the K-means algorithm requires less time than random seed, as confirmed by Table 5.

In Table 5, we present the total time in seconds for the computation of the initial clusters up to the time it takes K-means to converge. These re-

sults show that Var-Part, as an initialization method, requires less time than PCA-Part or one run of random seed. Var-Part and sub-sampling with refinement result in the best time performance, followed by random seed and KKZ. The worse time performance is by PCA-Part. Note that this table shows the average running time for one random seed run. In practice, to escape from getting stuck at a local minimum, one typically applies r random seed restarts (i.e., run random seed and K-means r times). Then, select the final clustering from the r runs by choosing the groupings that gave the minimum SSE. Typically random seed is run $r = 10$ times. Random restart thus on average takes ten times the time it takes one random seed to run shown in the table. Thus, Var-Part and even PCA-Part provide reasonable alternatives to initializing K-means. In addition, in these experiments, we simply applied the off-the-shelf code "the Template Numerical Toolkit (TNT)" [29] to compute PCA, which uses the Householder-QL algorithm to compute all the eigenvectors and eigenvalues. As we explained in section 3.2, using the power method can be faster since only the first eigenvector is needed. Yet, PCA-Part in our experiments seems to provide comparable time performance with the other initialization techniques presented here.

7.4 Experimental Results for Mixture of Gaussians

In this subsection, we present the results of the different initialization methods for initializing mixture of Gaussian clustering on the glass, segment, satellite, letter, pendigits, ionosphere, hrct20, covtype and mfeat-mor data.

Table 4: The number of iterations it takes K-means to reach convergence. Boldface indicates the initialization method which led K-means to reach convergence with the fewest number of iterations for each data set.

Method	Glass	Segment	Satellite	Letter	Pendigits	Hrct	Covtype	ionosphere	mfeat mor
Rand-S	12.01 ± 4.90	16.81 ± 7.00	21.53 ± 7.83	33.38 ± 9.28	20.75 ± 8.24	19.16 ± 6.38	18.97 ± 6.28	6.16 ± 1.76	15.48 ± 6.15
Refine	5.27 ± 2.78	11.78 ± 6.76	14.46 ± 7.39	28.52 ± 9.24	12.00 ± 5.73	14.42 ± 5.50	**7.17** ± 3.02	4.9 ± 1.67	9.32 ± 3.02
KKZ	**4.00** ± 0.00	30 ± 0.00	**7.0** ± 0.00	29.0 ± 0.00	12.0 ± 0.00	23.00 ± 0.0	13.00 ± 0.00	5.00 ± 0.00	19.00 ± 0.00
PCA-P	6.00 ± 0.00	10 ± 0.00	17.0 ± 0.00	49.0 ± 0.00	11.0 ± 0.00	25.00 ± 0.00	18.00 ± 0.00	2.00 ± 0.00	**6.00** ± 0.00
Var-P	6.00 ± 0.00	**8.00** ± 0.00	22.0 ± 0.00	**22.0** ± 0.00	**10.0** ± 0.00	**14.00** ± 0.00	12.00 ± 0.00	**2.00** ± 0.00	8.00 ± 0.00

Table 5: The running time it takes to initialize and to run K-means measured in seconds. Boldface indicates the initialization method which led K-means to reach convergence fastest for each data set. This table shows the average running time for one random seed run. Typically one applies 10 random seed restarts.

Method	Glass	Segment	Satellite	Letter	Pendigits	Hrct	Covtype	ionosphere	mfeat mor
Rand-S	0.01	0.14	0.94	11.10	1.11	1.10	28.53	0.01	0.06
Refine	0.01	0.12	0.71	7.14	**0.74**	1.14	**20.20**	0.01	0.07
KKZ	0.01	0.29	**0.45**	10.15	0.91	1.67	28.00	0.01	0.11
PCA-P	0.01	0.26	2.75	14.00	1.53	9.68	71.00	0.01	0.07
Var-P	0.01	**0.10**	1.17	**5.60**	**0.75**	**1.03**	28.00	0.01	**0.05**

7.4.1 The Quality of the Final Clustering

Table 6 shows the maximum log-likelihood values of the final clustering obtained when the various initialization methods are applied. Here, higher log-likelihood values are desired. PCA-Part shows the best performance, it results in the highest average log-likelihood for six out of the nine data sets (glass, segment,letter, covtype, ionosphere, hrct20), PCA-Part is followed closely by sub-sampling with refinement and Var-Part. KKZ performs the worst on all data sets except pendigits and covtype. KKZ obtains significantly smaller log-likelihood values compared to the other methods on five data sets (glass, satellite, hrct20, ionosphere and mfeat-mor). The effect of outliers on KKZ's performance is more pronounced in mixture of Gaussian clustering compared to that in K-means. The results here show that PCA-Part and Var-Part usually lead Gaussian mixture clustering to likelihood values bigger than the average performance of random seed, and in some cases close to the maximum likelihood values attained by the random methods.

7.4.2 Speed of Convergence

Table 7 lists the average and the standard deviation of the number of iterations that Gaussian mixture clustering needs to reach convergence for the different initialization methods. Random seed in general leads Gaussian mixture clustering to the slowest convergence compared to the other four methods. Sub-sampling with refinement, KKZ, PCA-Part and Var-Part

Table 6: The log-likelihood of Gaussian mixture clustering solutions. Higher log-likelihood values are desired. Boldface indicates the initialization method which led Gaussian mixture clustering to the maximal average log-likelihood value for each data set. Note that the average, maximal, and the minimal values of log-likelihood for Rand-S and refine are taken from 20 runs for covtype, 50 runs for HRCT and 100 runs for other data sets.

Method	Glass	Segment	Satellite	Letter	Pendigits	Hrct20	covtype	Ionosphere	mfeat mor
Rand-S									
max	2995	106355	-621598	-413108	-574293	44947	8.055e6	2185	34646
ave	2820 ±110	103621 ±2353	-623814 ±1413	-430578 ±9e3	-593142 ±1e5	44437 ± 244	8.042e6 ± 17e3	-1271 ± 2016	32101 ±1624
min	2489	93126	-626093	-455861	-629680	43873	7.981e6	-4685	27413
Refine									
max	3047	106383	-621737	-406117	-578474	44818	8.053e6	2127	34388
ave	2871 ± 101	105342 ±1102	-623569 ± 1317	-431057 ± 9e3	-594198 ± 9e3	44514 ± 170	**8.046e6** ± 8489	178 ± 1381	**32808** ± 1224
min	2597	100847	-627144	-456345	-618398	44150	8.030e6	-3051	29744.9
KKZ	2662 ± 0	103291 ± 0	-627218 ± 0	-446703± 0	**-582924** ± 0	43748 ± 0	8.044e6 ± 0	-3343	29681 ± 0
PCA-P	**3044** ± 0	**106246** ± 0	-625695 ± 0	**-416903** ± 0	-592619 ± 0	**44749** ± 0	**8.046e6** ± 0	**1155** ± 0	32599 ± 0
Var-P	2840 ± 0	104561 ± 0	**-622886** ± 0	-429976 ± 0	-594864 ± 0	44729 ± 0	8.042e6 ± 0	1149 ± 0	32599 ± 0

have comparable performance.

In Table 8, we present the total time in seconds for the computation of the initial clusters up to the time it takes Gaussian mixture clustering to converge. These results show that the capability of PCA-Part to lead Gaussian mixture clustering to fast convergence compensates for its longer computational overhead of computing for the principal components. The running time for all the initialization methods are comparable for all the data sets. Again, note that in practice, to escape from getting stuck at a local minimum, one typically applies r random seed restarts (i.e., run random seed and kmeans r times). Then, one selects the final clustering from the r runs by choosing the groupings that gave the minimum SSE. Typically random seed is run $r = 10$ times. Random restart, thus, on average takes ten times longer than the other methods.

Table 7: The average number of iterations it takes for Gaussian mixture clustering to converge. Boldface indicates the initialization method which led Gaussian mixture clustering to reach convergence with the fewest number of iterations for each data set.

Method	Glass	Segment	Satellite	Letter	Pendigits	HRCT20	Covtype	Ionosphere	mfeat mor
Rand-S	17.90 ± 9.35	22.52 ± 8.24	34.76 ±15.08	73.64 ±22.55	45.22 ±16.37	47.16 ± 18.77	84.3 ± 34.8	13.36 ± 12	34.18 ±18.8
Refine	16.32 ± 8.32	23.70 ± 8.76	33.52 ±17.41	65.26 ± 21.82	45.6 ± 17.03	45.36 ± 13.36	**52.8** ± 30.2	24.64 ± 8	26.16 ± 14.5
KKZ	25.00 ± 0.00	20.00 ± 0.00	**15.00** ± 0.00	88.00 ± 0.00	44.00 ± 0.00	68.00 ± 0.00	62.0 ± 0.0	**8.00** ± 0	**23.00** ± 0.0
PCA-P	13.00 ± 0.00	**14.00** ± 0.00	34.00 ± 0.00	89.00 ± 0.00	**30.00** ± 0.00	**32.00** ± 0.00	61.0 ± 0.0	30.00 ± 0	29.00 ± 0.0
Var-P	**11.00** ± 0.00	76.00 ± 0.00	38.00 ± 0.00	**66.00** ± 0.00	35.00 ± 0.00	42.00 ± 0.00	60.0 ± 0.0	32.00 ± 0	26.00 ± 0.0

Table 8: The running time it takes to initialize and to perform Gaussian mixture clustering. Boldface indicates the initialization method which led Gaussian mixture clustering to reach convergence fastest for each data set. This table shows the average running time for one random seed run. Typically one applies 10 random seed restarts.

Method	Glass	Segment	Satellite	Letter	Pendigits	Hrct20	Covtype	Ionosphere	mfeat mor
Rand-S	0.22	18.04	312.46	1871.92	237.30	39.90	11904	1.76	5.26
Refine	0.32	18.98	407.06	2068.70	269.44	42.16	**8219**	3.46	6.08
KKZ	0.33	16.00	**163.00**	2219.00	239.00	56.00	9069	**1.08**	**3.82**
PCA-P	0.19	**11.00**	314.00	2254.00	**154.00**	**26.00**	8910	4.21	4.41
Var-P	**0.15**	59.00	354.00	**1707.00**	187.00	35.00	8556	4.51	4.04

8 Conclusion

The performance of K-means and Gaussian mixture clustering depends on the initial starting conditions. Typically these clustering algorithms are initialized with random methods. In this paper, we examined two deterministic divisive hierarchical initialization methods for clustering: PCA-Part and Var-Part. We provided a theoretical motivation behind PCA based partitioning methods and show its strengths and limitations. PCA-Part is computationally intensive, because it needs to find the largest principal component at every stage. To ameliorate this expensive calculation, we introduced Var-Part, which approximates PCA-Part assuming diagonal covariance matrices.

We performed a comparative study between these two deterministic methods against two random methods: random seed and random sub-sampling with refinement. We also compared them against a deterministic method called KKZ. The experiments show that usually Var-Part and PCA-Part led K-means to have smaller MSE values compared to the average performance of random seed, with values close to the minimum value obtained by the random seeds over tens of runs for large data sets or hundred runs for small and medium sized data sets. In the same way, Var-Part and PCA-Part also led Gaussian mixture clustering to have higher log-likelihood values compared to the average performance of random seed. Both Var-Part and PCA-Part have better performance than KKZ in our experiments, and are equal or slightly better than random sub-sampling with refinement. Var-Part has computa-

tional complexity equal to only one iteration of K-means. Experiments on running time show that Var-Part achieves good clustering performance at speeds equivalent to one random seed run.

The results of this paper are encouraging. In case one cannot afford several random start runs, our deterministic initialization methods provide reasonable alternatives. PCA-Part and Var-Part present some promise in initializing at intelligent starting points for both K-means and Gaussian mixture clustering for reaching values close to the optimum, instead, of just random start. Moreover, deterministic methods ensure repeatability of experiments. This work suggests research directions, such as exploring other ways of partitioning the sample space (e.g., "pie"-slices). When time complexity is not crucial, one may apply different non-random intelligent restarts (capturing different possible data configuration scenarios), apply PCA-Part*, or combine random and non-random restarts for initializing K-means and Gaussian mixture clustering. This way, we are assured that at least one of the clustering runs (due to the intelligent starting points) would lead to good clustering results in terms of SSE or log-likelihood.

References

[1] M. R. Anderberg. *Cluster Analysis for Applications*. Academic Press, NewYork, NY, 1973.

[2] S. D. Bay. The UCI KDD archive, 1999.

[3] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[4] D. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[5] P. S. Bradley and U. M. Fayyad. Refining initial points for K-means clustering. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 91–99, San Francisco, CA, 1998. Morgan Kaufmann.

[6] S. Dasgupta. Experiments with random projection. In *Proceedings of the 16th Conference on Uncertainty in Artificial In telligence*, pages 143–151. Morgan Kaufmann Publishers Inc., 2000.

[7] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[8] O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley-Interscience, NewYork, NY., 2000.

[9] J. G. Dy, C. E. Brodley, A. Kak, L. S. Broderick, and A. M. Aisen. Unsupervised feature selection applied to content-based retrieval of lung images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(3):373–378, 2003.

[10] U. M. Fayyad, C. Reina, and P. S. Bradley. Initialization of iterative refinement clustering algorithms. In *Knowledge Discovery and Data Mining*, pages 194–198, 1998.

[11] E. Forgy. Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. *Biometrics*, 21:768, 1965.

[12] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, Boston, MA, 1990.

[13] G. Golub and C. V. Loan. *Matrix Computations*. John Hopkins University Press, 1996.

[14] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[15] J. He, M. Lan, C.-L. Tanz, S.-Y. Sungz, and H.-B. Low. Initialization of cluster refinement algorithms:. In *Proceedings 2004 IEEE International Conference on Neural Networks*, pages 297–302, 2004.

[16] C.-M. Huang and R. Harris. A comparison of several vector quantization codebook generation approaches. *IEEE Trans. on Image Processing*, 2(1):108–112, 1993.

[17] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[18] I. Jolliffe. *Principal Component Analysis*. Spring-Verlag, New York, 1986.

[19] J.T.Tou and R.C.Gonzalez. *Pattern Recognition Principles*. Addison-Wesley, Massachusetts, 1974.

[20] I. Katsavounidis, C.-C. J. Kuo, and Z. Zhang. A new initialization technique for generalized Lloyd iteration. *IEEE Signal Processing Letters*, 1(10):144–146, 1994.

[21] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Morgan Kaufmann, San Francisco, 1995.

[22] J. Macqueen. Some methods for classfications and analysis of multivariate observations. *Proc. Symp. Mathmetic Statistics and Probability, 5th, Berkeley*, 1:281–297, 1967.

[23] G. McLachlan and D. Peel. *Finite Mixture Models*. Wiley, New York, 2000.

[24] M. Meilă and D. Heckerman. An experimental comparison of several clustering and initialization methods. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 386–395, San Francisco, CA, 1998. Morgan Kaufmann.

[25] C. J. Merz, P. Murphy, and D. Aha. UCI repository of machine learning databases, 1996.

[26] G. Milligan. An examination of the effect of six types of error perturbation on fifteen clustering algorithms. *Psychometrika*, 45(21):325–342, 1980.

[27] S. A. R. Moh B. Al-Daoud. New methods for the initialisation of clusters. *Pattern Recognition Letters*, 17:451–455, 1996.

[28] J. Pena, J. Lozano, and P. Larranaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, 20:1027–1040, 1999.

[29] R. Pozo. The template numerical toolkit (tnt).

[30] W. H. Press, W. T. Vetterling, S. A. Teukolsky, and B. P. Flannery. *Numerical Recipes in C++: the Art of Scientific Computing*. Cambridge University Press, Cambridge, UK, 2002.

[31] J. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[32] R.L.Burden and J.D.Faires. *Numerical Analysis*. Weber & Schmidt, Boston, MA, 1985.

[33] A. Strehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *Journal on Machine Learning Research*, 3:583–617, December 2002.

[34] J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58:236–244, 1963.