THESIS

AN ADAPTATION OF K-MEANS-TYPE ALGORITHMS TO THE GRASSMANN MANIFOLD

Submitted by

Shannon J. Stiverson

Department of Mathematics

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2019

Master's Committee:

Advisor: Michael Kirby

Henry Adams
Asa Ben-Hur

ABSTRACT

AN ADAPTATION OF K-MEANS-TYPE ALGORITHMS TO THE GRASSMANN MANIFOLD

The Grassmann manifold provides a robust framework for analysis of high-dimensional data through the use of subspaces. Treating data as subspaces allows for separability between data classes that is not otherwise achieved in Euclidean space, particularly with the use of the smallest principal angle pseudometric.

Clustering algorithms focus on identifying similarities within data and highlighting the underlying structure. To exploit the properties of the Grassmannian for unsupervised data analysis, two variations of the popular K-means algorithm are adapted to perform clustering directly on the manifold. We provide the theoretical foundations needed for computations on the Grassmann manifold and detailed derivations of the key equations. Both algorithms are then thoroughly tested on toy data and two benchmark data sets from machine learning: the MNIST handwritten digit database and the AVIRIS Indian Pines hyperspectral data. Performance of algorithms is tested on manifolds of varying dimension. Unsupervised classification results on the benchmark data are compared to those currently found in the literature.

ACKNOWLEDGEMENTS

I would like to thank my advisor Michael Kirby for his insight, guidance, and support throughout this process. I would also like to thank Chris Peterson, Henry Adams, and Asa Ben-Hur for their time and feedback.

I am very grateful for all my colleagues who have been both friends and collaboraters during my time at CSU.

I thank my family, and particularly my parents, Ken and Brenda Anderson, for the unwavering support and encouragement they have provided for as long as I can remember.

I am deeply grateful to my many, many friends who have cheered me on and been a constant source of encouragement and generally contributed to keeping me sane the past few years. Your support has meant more to me than you likely realize.

I am thankful for my cats, Clover and Winston, for being constant companions and great sources of joy and humor.

Last but certainly not least, I would like to thank my husband, Dan, who has accompanied me through every step of this journey and worked tirelessly to make this possible. Thank you for your love and your patience, I could not have done this without you.

This work is based on research partially supported by the National Science Foundation under Grants No. DMS-1513633, and DMS-1322508 as well as DARPA awards N66001-17-2-4020 and D17AP00004. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or DARPA.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

The Grassmann manifold provides a robust geometric framework for analyzing data sets of high dimension. Evidence suggests that subspace representations of data are more robust to within-class variations and random noise than data points in Euclidean space [1, 2]. For example, object recognition is confounded by variations in illumination due to the sensitivity of the image representation to the illumination angle. The use of the Grassmannian greatly mitigates this issue, as shown in [1, 2]. Additionally, there are cases where classes of data that are inseparable in $\mathbb{R}^n$ become separable when represented as points on the Grassmann manifold. A simple example of this is shown in Figure 1.1 using embeddings of images of handwritten zeros and ones. In particular, experimental results suggest that the smallest angle pseudometric defined on the Grassmannian performs especially well in regards to data separation [3].

In addition to the robustness of subspace representations, another geometric property makes analysis on the Grassmannian particularly attractive. A basic example of this property is shown in the simple two-class classification problem in Figure 1.2. Here, the two classes can quite easily be visually identified as lying along two lines through the origin, but standard machine learning algorithms as well as Euclidean clustering algorithms will fail to separate the two classes. However,

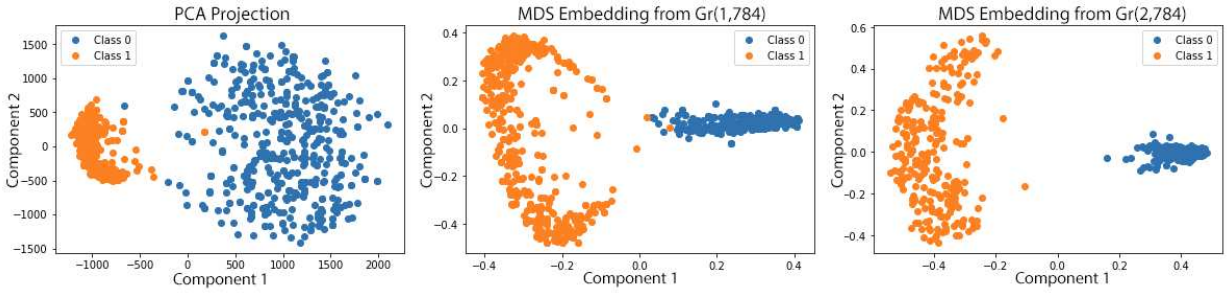

Figure 1.1: Embedding of MNIST handwritten digit data in $\mathbb{R}^2$ using Euclidean distance (left), chordal distance on $Gr(1, 784)$, and chordal distance on $Gr(2, 784)$.

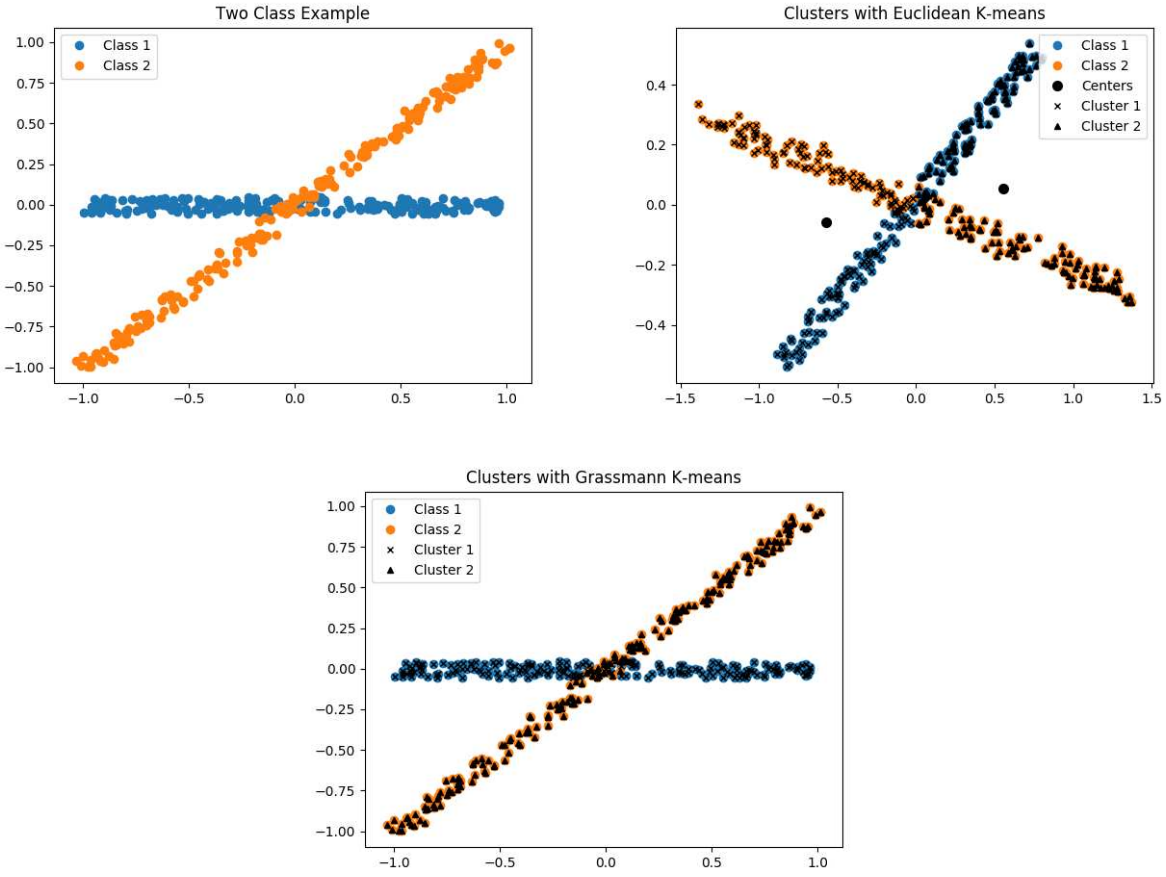

Figure 1.2: Two classes of points along lines in $\mathbb{R}^2$ with labels using both Euclidean and Grassmannian clustering.

when clustering is performed on the Grassmann manifold by treating each data point as a one-dimensional subspace in $\mathbb{R}^2$ (which can be more generally done for d dimensions in $\mathbb{R}^n$), the labelling becomes far more accurate. Overall, the benefits of analysis on the Grassmannian make it highly desirable to develop Grassmannian implementations of commonly used algorithms.

Vector quantization and data clustering have been widely used in pattern recognition and data classification for many decades [4]. Applications for these methods are found across a wide variety of fields, including speech recognition, image segmenting, sorting applications, and even biological applications [4], see Section 2.1 for additional citations. Clustering algorithms such as K-means and LBG are particularly useful for analyzing data without the use of labels and for inferring the underlying geometry of a data set. This makes clustering a powerful tool in analysis of novel

data when little other information is available. In order perform this type of clustering on the Grassmannian, we provide adaptations for two of the most common variations of the popular K-means algorithm for use directly on the Grassmann manifold.

This type of analysis is particularly useful for unsupervised classification in cases where where multiple data samples are received in a group and are known from context to be associated with one another, but the true label for the data points in unknown. The object recognition problem mentioned previously is one case where this can occur. For example, video footage captures multiple images of an individual in sequence. These video frames can be identified as a group of related images know to contain the same person, but do not inherently come with a label identifying that person. Such collections of images can be represented as a single subspace, as could similar video footage for multiple individuals. Subspace clustering would then allow for unsupervised identification of different groups of images that feature the same person. Another example of this type of problem is a biological context where multiple measurements or samples are taken from a single individual over an extended period of time. All such samples can be associated together as a subspace, with such subspaces representing presence or absence of a disease state for potentially infected subjects. The same idea can be extended to any context where multiple samples are taken from a single source and then used for classification purposes.

1.2 Overview

Here we summarize the content of this thesis. The primary contribution of this work is the adaptations of the K-means and LBG algorithms to the setting of the Grassmann manifold.

Chapter 2 reviews the background for clustering and quantization. It summarizes the various approaches to clustering and includes applications and algorithms. In particular, we provide a detailed background of the K-means family of algorithms, as well as an overview of the more widely used modifications and extensions developed for it. We then discuss previous work on clustering directly on the Grassmann manifold, as well as previous unsupervised clustering results on the benchmark data sets featured in this work.

Chapter 3 provides an in-depth exploration of the geometry of the Grassmann manifold, striving as much as possible to be self-contained. It then describes tools for analysis on the Grassmannian that are pivotal to the adaptation of Euclidean clustering algorithms to the Grassmann manifold.

Chapter 4 describes in detail the two algorithms of interest, including an analysis of computational complexity and a discussion of their relative properties.

Chapter 5 describes the adaptation of the algorithms in Chapter 4 to the Grassmann manifold using the theory developed in Chapter 3. Again, basic properties of the algorithms are compared and discussed, as is computational complexity.

Chapter 6 contains the experimental results generated over the course of this work. This includes testing algorithm functionality, comparisons between Euclidean and Grassmannian algorithms, and comparisons between the two algorithms on the Grassmann manifold.

Finally, Chapter 7 reviews the results of this thesis and discusses future avenues of exploration.

Chapter 2

Review of Literature

2.1 Overview and Applications of Clustering

Data clustering refers to the division of points in a data set into non-overlapping groups of points with intrinsic similarities. It is used to identify common characteristics in data and obtain information about the underlying data structure. Applications of clustering algorithms are found in image quantization, speech coding, document sorting, microarray and genome analysis, signal processing, and even social networking [4–10]. The prominence of clustering in data analysis has motivated the development of many approaches and algorithms [4]. Depending on the field and application, clustering data in $\mathbb{R}^n$ is also referred to as vector quantization [11].

The majority of clustering algorithms fall into one of two categories: hierarchical or partitional. Hierarchical clustering algorithms build a tree of clusters based on a proximity measure which spans the entire data set [12]. This is done either top-down by starting with a single large cluster and dividing into smaller clusters, or bottom-up by starting with each data point in its own cluster and building subsequent nested clusters based on similarities [4, 12]. Some examples of hierarchical clustering are found in [5, 13]. Partitional clustering methods seek to divide the data space into a set number of regions, with each region containing similar data points [4, 13]. The most common examples are the K-means-type algorithms derived from an algorithm initially developed by MacQueen [14]. MacQueen's K-means is discussed in more detail in Section 4.1. As the algorithms in this work both fall into the partitional category, more detailed background and theory is included in Sections 2.2 and 2.3.

Within the two primary categories of clustering algorithms, there are wide variety of different approaches to defining clusters. Density-based methods view a clusters as a regions of high density surrounded by low density regions [4]. These regions are located using information on common neighbors [15], or by applying statistical methods to identify high density regions with

some probability [16–18]. The primary drawback of this approach is its poor performance in high dimensions, where data is often very sparse. Subspace clustering is one answer to difficulties of clustering high-dimensional data. This clustering approach involves projecting points in a high-dimensional space onto one or more low dimensional subspaces for the purpose of clustering [4]. Vidal describes multiple variations of subspace clustering in [19], and Vidal and Elhamifer developed a sparse subspace clustering method in [20]. Another example of subspace clustering by Aggarwall can be found in [21]. Employing an information theoretic approach to cluster selection leads to constructing clusters in a way that minimizes entropy. Further information can be found in [22–25]. Spectral clustering is based on the graph theoretic approach. Similarity graphs are constructed based on pairwise distances given by some metric, and then the minimum cut problem is solved to divide points into clusters [4]. Laplacian eigenmaps [26] fall into this category. In some cases, multiple clustering approaches are combined to form hybrid clustering algorithms. For example, Kao and Karami employ K-means clustering in conjunction with Particle Swarm Optimization (PSO), another widely used clustering algorithm [27, 28].

For the purposes of data analysis and prediction, all clustering approaches can be further divided into three categories: supervised, unsupervised, and semi-supervised. Supervised clustering uses labeled data to build clusters, generally for use in making predictions on unlabeled data [4]. Unsupervised clustering operates on unlabeled data. Many commonly used clustering algorithms fall into this category, including K-means [4, 14]. Realistically, however, we often possess some background knowledge about the data that can be used to inform the clustering process. Semi-supervised clustering takes unlabeled data and adds constraints to how the data can be clustered [4]. Pairs of data points are given "must-link" or "cannot-link" restrictions based on this background information. Wagstaff's constrained K-means algorithm employs such a method [29].

The data clustering problem comes with many inherent difficulties. Depending on the algorithm chosen, different assumptions are made about the underlying structure of the data. Because clustering is often performed by optimizing over a cost function, the choice of distance metric used

to calculate cost adds a bias to the shape of the final clusters. Because of this, different algorithms can yield extremely different results when applied to the same data set [4].

A second difficulty is that clustering algorithms are sensitive to starting conditions and the order of data presentation. If initial conditions are poorly selected, algorithms will terminate in local minima rather than finding the true optimal partitioning. A common way of addressing this is to run repeated clustering trials on a data set to determine the true minimum, but this becomes prohibitively expensive when data sets are very large. Alternative methods for addressing large data set clustering are described in [12, 30–32].

Even quantitative comparison of two algorithms is difficult, given that the "optimal" partitioning of a data set depends both on the data itself and how clusters are defined. A variety of criteria have been used to compare and contrast clustering algorithms. Common evaluation metrics include within-cluster similarity, cluster distortion, cluster entropy, algorithm precision and recall, total run time, computational complexity, number of iterations needed for convergence, stability, and performance across different data sets [4, 33–35]. Ultimately, there is no one clustering algorithm that can be considered the "best," as the relative performance of algorithms differs based on the information desired as well as the data itself.

2.2 Partitional Clustering

Vector quantization refers specifically to the partitional clustering case where a vector space is divided into smaller discrete units, though other types of clustering are also referred to as quantization in some literature. The following definitions come from quantization theory, but the terminology is interchangeable for that of partitional clustering.

Let $\mathbf{X}$ be a metric space, x be elements of $\mathbf{X}$, and S_i be a partition unit for all i in some index set $\mathcal{I}$. A quantizer maps all $x \in \mathbf{X}$ by $q(x) = c_i$ for all $x \in S_i$, where c_i is the representative for partition unit S_i [11]. Gersho and Grey describe necessary conditions for local optimality of a partition [36]. For a partition to optimally minimize

$$\sum_{i=1}^{k} \sum_{x \in S_i} d(x, c_i)$$

for some distance metric d, the units S_i must satisfy the nearest neighbor condition defined by

$$S_i \subseteq \{x \in \mathbf{X} : d(x, c_i) \leq d(x, c_j), i \neq j)\} \tag{2.1}$$

and the representative vectors c_i must satisfy the centroid condition given by

$$c_i = \arg\min_y \{d(x, y) | x \in S_i\}. \tag{2.2}$$

In addition, the partition must satisfy

$$S_i \cap S_j = \emptyset \quad \text{for all} \quad i, j \in \mathcal{I}$$

and

$$\bigcup_{i \in I} S_i = \mathbf{X}.$$

This can be accomplished by defining the c_i to be the average of all points in S_i, where S_i is the Voronoi cell about c_i defined by

$$S_i = \{x \in \mathbf{X} : d(x, c_i) \leq d(x, c_j), i \neq j)\} \tag{2.3}$$

with points falling on the edge between two cells assigned to one or the other arbitrarily (usually based on index order) [36]. The element c_i acts as a representative for all $x \in S_i$. The set $\{c_i : i \in \mathcal{I}\}$ is referred to as a codebook for $\mathbf{X}$ [11, 36].

A partition is evaluated by defining a distortion measure that quantifies the cost of representing a data point $x \in S_i$ by c_i [11]. Many applications employ quantization to great effect, including speech recognition and image processing [9, 10, 37]. As stated previously, data clustering and vector quantization are functionally equivalent, with the cluster centroids acting as the codebook

for the data set. The same metric of distortion error can be applied to partitional clustering algorithms. As the remainder of this work will deal exclusively with partitional clustering, the terms "clustering" and "quantization" are both taken to mean the partitioning process described above.

Broadly speaking, partitional clustering algorithms handle data either as an online stream or an offline batch. Online or data streaming algorithms acquire a novel data point at each iteration and use it to update centers. MacQueen's original K-means algorithm falls into this category [14]. Other examples of such algorithms can be found in [38–40]. Offline algorithms consider the data set as a whole, and update centers using all of the data at each iteration. Examples of offline algorithms include those developed by Linde et al. and Lloyd [7, 41]. Some clustering algorithms combine the two methods by interspersing a "batch" step periodically throughout the streaming updates. In this work, we consider one batch update algorithm and one online update algorithm.

2.3 K-means-type Algorithms

A "K-means-type" algorithm refers to any of a number of algorithms that take a given set of data and partition it into k distinct clusters, each of which can be optimally represented by its associated center. In particular, the goal of these algorithms is to cluster the data in such a way that minimizes an error function defined in terms of within-cluster distortion. Given n data points $x \in \mathbf{X}$, where $\mathbf{X}$ is a metric space, and k centers c_i for $1 \leq i \leq k$, we construct clusters C_i by assigning each data point to a single center. Total cluster distortion is given by

$$D = \sum_{i=1}^{k} \sum_{x \in S_i} d(x, c_i) \tag{2.4}$$

for some distance metric $d(x, y)$ defined on $\mathbf{X}$. Initial centers are chosen and updated at each iteration of the algorithm, with the precise update method varying amongst algorithms.

Variations of K-means-type algorithms are found throughout the literature. A particularly common version is the fuzzy C-means algorithm developed by Dunn [42]. This algorithm allows for points to be assigned to multiple clusters. It was further developed by Bezdek [43, 44]. Krishna

and Murty developed the genetic K-means algorithm based on principles from evolutionary biology [45]. The kernel K-means algorithm by Schölkopf et al. performs nonlinear quantization by first embedding data into a higher-dimensional kernel space [46]. A version of kernel K-means incorporating spectral clustering was developed by Dhillon et al. [47], and a hierarchical version of the K-means algorithm is described by Steinbach et al. in [33]. The K-mediod algorithm is a K-means variation that uses cluster medians to represent data rather than means [48]. ISODATA, developed by Ball and Hall, performs batch cluster updates while dynamically discarding small clusters [49]. It is widely used for pattern recognition [4].

The majority of these algorithms function as unsupervised methods of dividing or classifying data, with the only user-selected parameter being the chosen number of clusters k. Some variations on K-means propose methods for automatic selection of the parameter k. These include global K-means by Likas et al. [50], a variation by Hamerly and Elkan referred to as G-means [51], and a gap statistic method by Tibshirani et al. [52].

A known weakness of K-means-type algorithms is their sensitivity to the initial choice of centers and subsequent tendency to terminate in local minima. This problem is exacerbated in smaller data sets. Multiple methods have been developed for dealing with this issue. Pena et al. [53] provide an overview and comparisons of some commonly used methods for choosing initial centers. Pelleg and Moore developed the X-means algorithm for estimation of the number of clusters [54]. Additional methods are explored in [55–59]. The two simplest common methods of initialization are choosing centers randomly from the data itself and choosing centers randomly throughout the data space. The optimal method for choosing initial conditions that avoid local minima is highly dependant on the data set and ultimately beyond the scope of this work. To avoid the complication of empty clusters, initialization will be done by selecting centers from the data.

Variations on the K-means algorithms fall into both batch update and online update categories. Here we will deal with adapting one of each type of algorithm to the Grassmannian. Some K-means-type implementations use a combination of both batch and online updates, and though we

do not deal with these explicitly in this work, combining the given algorithms to function together on the Grassmannian should be fairly straightforward.

The reader may note that thus far we have referred to "K-means" as a general class of algorithms rather than a single specific method. This is largely due to conflicting naming conventions for K-means-type algorithms in the existing literature. A variety of different algorithms are referred to as simply "K-means," though further inspection of the methodology reveals significant differences in implementation. In particular, the name "K-means" is used in literature for both online and batch versions of the algorithm. As we consider both a batch algorithm and an online algorithm, we must establish a consistent naming convention to differentiate the two.

The online algorithm studied in this thesis is the original version of K-means published by James MacQueen in 1967 [14], discussed in more detail in Section 4.1. For the sake of clarity, "K-means" will refer only to this online algorithm for the remainder of this work. The batch algorithm explored in this work is the version developed by Linde, Buzo, and Gray [7] as a generalization of Lloyd's one-dimensional quantization algorithm [41] to function in n dimensions. The batch algorithm will hence be referred to as "LBG." Details for this version of the algorithm are found in Section 4.2.

2.4 Clustering on the Grassmannian

Though some work has been done clustering directly on the Grassmannian, most clustering algorithms require that points on the manifold first be embedded into another space. This section contains an overview of the most recent work on the Grassmann manifold.

Dong et al. provide an algorithm for constructing a subspace representation of data to act as points on the Grassmannian. Pairwise distances are then used for spectral embedding and clustering [60]. Shiraz et al. explore a kernel clustering method for points on the Grassmannian, which requires a mapping from the points on the manifold to points in a kernel space [61].

Some density based clustering methods are applied directly to the Grassmann manifold. Turaga et al. use a statistical approach based on the Karcher mean [62], and Cetingul et al. developed a mean shift algorithm that uses density estimates to iteratively update cluster modes [63].

A K-means-type clustering approach on the Grassmannian is described by Gruber and Theis in [64]. This method uses a subspace averaging approach based on theory developed by Bradley and Mangasarian in [65]. In this approach, the average subspace is calculated by minimizing the projection Frobenius norm between all subspaces and the average subspace. This reduces to an eigenvector computation on the covariance of matrix of points in the cluster [64, 66]. Though the end result appears similar to that of the flag mean method from [67] and described in Section 3.2.3, Santamaria et al. demonstrate that the solution based on the projection Frobenious norm is equivalent to the flag mean for the particular case where a d-dimensional flag is calculated using only d-dimensional subspaces, i.e., on the Grassmannian [66]. Although Gruber and Theis propose the use of the subspace mean for clustering on the Grassmannian, they only include demonstrations on toy data in low dimensions, primarily focusing on projective clustering [64]. However, they discuss the extension of partitional clustering to function on affine subspaces, which would be an intriguing avenue of future work.

2.5 Clustering on Featured Data Sets

Two benchmark data sets are used to test and compare the algorithms developed in this paper. The MNIST handwritten digit database contains approximately 70,000 samples of handwritten digits 0 through 9 [68]. All samples are reduced to 28×28 pixel resolution black and white images. These are converted into 28×28 matrices, with each coordinate containing a binary indicator of whether or not the corresponding pixel is colored. These matrices are vectorized for data analysis. The AVIRIS Indian Pines data set contains hyperspectral imaging data from a test site in Indiana [69]. The ground truth data consists of class labels based on a 145×145 pixel image of the site. Of this, 10,776 points fall into the empty class and have no associated spectral data. The remaining 10,249 points are divided into 16 classes according to the samples' features. The

hyperspectral data contains values for 220 bands of varying wavelength for each classified data point. A reduced version of the data set removes the bands corresponding to the wavelengths of water absorption, bringing the total number down to 200. The reduced version of the data set is used for all trials in this work.

To provide some metric of comparison for performance of the algorithms developed in this paper, previous unsupervised clustering results on both data sets are reviewed here.

The MNIST experiments described next assign to each cluster the label of the class with the highest probability of membership, i.e. the class corresponding to the highest fraction of points. The accuracy scores reported are the total percentage of correct labels across all clusters [70]. Springenberg applied categorical generative adversarial networks (GANs) to the MNIST data set and the best accuracy achieved was 90.30% using $k = 20$ centers [71]. The adversarial autoencoder developed by Makhazani et al. had an average accuracy of $95.90\%\pm1.13$ using $k = 30$ centers [72]. A Gaussian mixture variational autoencoder developed by Dilokthanaku et al. achieved an average classification rate of $92.77\% \pm 1.60$ by clustering using $k = 30$ centers [70]. For purposes of comparison, we will utilize the same accuracy metric as the above trials to evaluate our algorithms.

Xie et al. report classification accuracy using a slightly different metric. Their methods are evaluated by taking all data points x_i and evaluating

$$\max_m \frac{1}{n} \sum_{i=1}^{n} 1_{l_{x_i}=m(c_i)} \tag{2.5}$$

where n is the number of data points, l_i is the true label for the ith point, c_i is the center the ith point is assigned to, and m ranges through all possible one-to-one mappings between class labels and centers [73]. Here, the quantity $\{l_{x_i} = m(c_i)\}$ is set to one if the true label of x_i matches the label of its assigned center under mapping m, and is zero otherwise. Using this metric, the authors evaluated multiple clustering methods on the MNIST data set. Their Deep Embedded Clustering algorithm achieved the highest accuracy of all methods reported at 84.30% using $k = 10$ centers [73]. Notably, they also report results using MacQueen's K-means algorithm [14], and achieve an accuracy of only 53.49%.

The clustering literature for the Indian Pines data set is primarily focused on identifying relevant spectral bands for classification rather than classifying the data directly. Wu and Tsuei apply clustering to the cross-correlation matrices of the hyperspectral bands to perform dimensionality reduction on the data [74]. Tuia and Camps-Valls embed the data into a kernel space before clustering [75]. Su et al. propose a semi-supervised clustering method for band selection and dimensionality reduction [76]. Chepushtanova et al. perform supervised classify the data on the Grassmannian, using sparse support vector machines to select relevant hyperspectral bands [77]. As far as the author is aware, however, there is no literature describing unsupervised clustering of this data set directly on the Grassmannian.

Chapter 3

The Grassmann Manifold

3.1 Matrix Manifolds and the Grassmannian

A manifold of dimension d is a space $\mathcal{M}$ for which all points $x \in \mathcal{M}$ are contained in some neighborhood which can be bijectively mapped to an open subspace of $\mathbb{R}^d$ [78]. Clearly, all d-dimensional vector spaces are manifolds. Consider the set of all $n \times p$ real matrices $\mathbb{R}^{n\times p}$ with the standard addition and scalar multiplication. The matrix vectorization operation, which entails vertically stacking all p columns into a single vector, is then a bijective mapping $f : \mathbb{R}^{n\times p} \to \mathbb{R}^{np}$. Therefore $\mathbb{R}^{n\times p}$ is a manifold of dimension pn. This becomes a Euclidean manifold when equipped with the inner product defined by

$$\langle X_1, X_2 \rangle = vec(X_1)^T vec(X_2) = tr(X_1^T X_2) \tag{3.1}$$

where $X_1, X_2 \in \mathbb{R}^{n\times p}$ [78]. The topology of this manifold is equivalent to that of the canonical Euclidean topology on $\mathbb{R}^{np}$.

A matrix X is orthonormal if $X^T X = I$. The set of all orthonormal $n \times p$ matrices, called the Stiefel manifold and denoted $St(p, n)$, is an embedded submanifold of $\mathbb{R}^{n\times p}$ and is thus equipped with corresponding induced subset topology [78]. To calculate the dimension of this manifold, consider the mapping $F : \mathbb{R}^{n\times p} \to S_p$, where S_p is the set of all $p \times p$ symmetric matrices, given by

$$F(X) = X^T X - I_p. \tag{3.2}$$

This mapping is surjective, and since all matrices in $St(p, n)$ are orthonormal, $St(p, n) \subseteq F^{-1}(0)$. Conversely, $F(X) = 0$ only if $X^T X = I_p$, so $F^{-1}(X) \subseteq St(p, n)$ since all orthonormal $n \times p$ matrices must exist in the Stiefel manifold. Since $p \times p$ symmetric matrices are completely determined by the entries on and above the diagonal, the dimension of S_p is $\frac{1}{2}p(p+1)$. Additionally,

since the mapping F is surjective, we have

$$dim(\mathbb{R}^{n\times p}) = dim(S_p) + dim(F^{-1}(0)) = dim(S_p) + dim(St(p,n)) \tag{3.3}$$

and therefore the dimension of $St(p,n)$ is $pn - \frac{1}{2}p(p+1)$ [78].

The Grassmann manifold is defined as the parameterization of the set of all p-dimensional subspaces in $\mathbb{R}^n$, with $p \leq n$. Denoted $Gr(p,n)$, this manifold allows for parameterization of subspaces in $\mathbb{R}^n$. Each point on a Grassmannian is an p-dimensional subspace, which can be represented as the span of the columns of some $n \times p$ matrix. The column space of a matrix X is henceforth denoted by $[X]$. The Grassmannian is a quotient manifold of the Stiefel manifold, with $Gr(p,n) = St(p,n)/\sim$, with the equivalence relation $\sim$ is defined by

$$X \sim Y \iff [X] = [Y] \tag{3.4}$$

where X and Y are $n \times p$ matrices. The Grassmannian is then the set of all equivalence classes of $\sim$ in $St(p,n)$ and is therefore a quotient manifold of the Stiefel manifold. A function defined on $St(p,n)$ is then invariant under $\sim$ if

$$f(X) = f(Y) \iff [X] = [Y] \tag{3.5}$$

for all $X, Y \in St(p,n)$ [78]. The column space of an orthonormal $n \times p$ matrix is invariant under right multiplication by a $p \times p$ orthonormal matrix. This means that the column space of an $n \times p$ matrix is uniquely determined by $p(n-p)$ entries, therefore the dimension of $Gr(p,n)$ is $p(n-p)$ [78]. This relationship to the Stiefel manifold provides an intuitive framework for computational analysis on the Grassmannian. Points on the Grassmann manifold are represented by orthonormal matrices for the purposes of numerical computations.

3.2 Tools for Analysis on the Grassmannian

The first step in analyzing data on a Grassmannian is converting data from Euclidean points to points on the Grassmann manifold. In most cases, data is represented as a feature vector in $\mathbb{R}^n$. To construct points on the Grassmannian from euclidean data in $\mathbb{R}^n$, take p data vectors and concatenate them into an $n \times p$ matrix $\hat{X} = [x_1, x_2, ..., x_p]$. Performing QR factorization on $\hat{X}$ yields $\hat{X} = QR$, where Q is an orthonormal matrix with column space equal to that of $\hat{X}$. Thus $X = Q$ defines a matrix representation for a point in $Gr(p, n)$ defined by the p data points. Each subspace is assigned the same label as the points used to construct it. Given m total data points for a class, we obtain $\lfloor m/p \rfloor$ total subspaces, discarding m modulo p points.

3.2.1 Metrics

In order to perform computations, we must establish a notion of distance on the Grassmann manifold. Distances between two points on the Grassmannian can be defined using the principal angles between the two subspaces. Denote the principal angles between two subspaces $[X]$ and $[Y]$ generated by $X, Y \in \mathbb{R}^{n \times p}$ by θ_i for $1 \leq i \leq p$. These angles are defined recursively by

$$\theta_1 = \min\{\arccos(x^T y) |\ x \in [X],\ y \in [Y]\} \tag{3.6}$$

subject to $\|x\| = \|y\| = 1$, and more generally by

$$\theta_i = \min\{\arccos(x^T y) |\ x \in [X],\ y \in [Y]\} \tag{3.7}$$

subject to $\|x\| = \|y\| = 1,\ x^T x_j = 0$, and $y^T y_j = 0\ \forall\ j \leq i$. So θ_1 is the smallest possible angle between the two subspaces, and $\theta_i \geq \theta_j$ for all $i > j$. The corresponding unit vectors $x_i \in [X]$ and $y_i \in [Y]$ are called the principal vectors [79].

For two orthonormal matrices X and Y, the cosines of the principal angels between $[X]$ and $[Y]$ are easily obtained from the singular value decomposition (SVD) of $X^T Y$. Given the SVD of $X^T Y = U \Sigma V^T$, the singular values σ_i are by definition given by

$$\sigma_i = \max_{||u||,||v||} u^T(X^TY)v = u_i^T(X^TY)v_i \tag{3.8}$$

subject to u^Tu_j and v^Tv_j for all $j < i$. Now define $x_i = Xu_i$ and $y_i = Yv_i$. Then $x_i \in [X]$ and $y_i \in [Y]$, and $\sigma_i = u_i^T(X^TY)v_i = x_i^Ty_i$, which is the cosine of the angle between x_i and y_i. Using this with equations (3.7) and (3.8), we obtain $\sigma_i = \cos(\theta_i)$ and therefore $\theta_i = \arccos(\sigma_i)$ [79]. Thus, the SVD provides a relatively low cost computational method for quickly obtaining principal angles from orthonormal bases for subspaces. Note that all principal angles θ must exist in the interval $[0, \frac{\pi}{2}]$.

The Grassmannian can be endowed with many distinct orthogonally invariant geometric structures written in terms of principal angles; see [80] for details. The projection Frobenius norm provides a distance metric, called chordal distance, on $Gr(p, n)$ that can be expressed in terms of the sines of principal angles. If $[X], [Y] \in Gr(p, n)$ then

$$d_c([X], [Y]) = \left(\sum_{i=1}^p (\sin\theta_i)^2\right)^{1/2} = \|\sin\theta\|_c. \tag{3.9}$$

Using chordal distances, any set of points on the Grassmannian can be isometrically embedded into Euclidean space using multi-dimensional scaling (MDS) [80]. A second norm, referred to as geodesic distance, is given by

$$d_g([X], [Y]) = \left(\sum_{i=1}^p \theta_i^2\right)^{1/2} = \|\theta\|_g. \tag{3.10}$$

It is important to note that because $\sin(\theta)$ is monotonically increasing for $\theta \in [0, \frac{\pi}{2}]$, geodesic distances can be related to chordal distances via a retraction mapping. However, geodesic distances cannot be used to obtain an isometric embedding of points using MDS. Finally, the smallest principal angle defines a pseudo-metric on the Grassmannian, with

$$d_p([X], [Y]) = \min_i \theta_i \tag{3.11}$$

This also cannot be isometrically embedded into Euclidean space and does not define a true norm on the Grassmannian, since $d_p([X],[Y]) = 0$ does not guarantee $[X] = [Y]$. Nevertheless, use of this pseudo-metric has been shown to result in better classification than the standard metrics, in addition to being much simpler computationally [3].

3.2.2 Geodesics and Parameterization

The shortest path distance between two points on a manifold is known as a geodesic curve. In order to update centers in the K-means algorithm, we require a way to move one subspace a specified distance towards another subspace. We acheive this by parameterizing the geodesic curve between two points $[X]$ and $[Y]$ on the Grassmannian. This is accomplished by using the quotient geometry of the Grassmannian to derive a formula for a geodesic curve [80].

Consider the set of all $n \times n$ orthonormal matrices, denoted by $O(n)$. $O(n)$ is a manifold in Euclidean space. Now suppose some $\tilde{Q} \in O(n)$ lies on a smooth curve $X(t)$ with $X(0) = \tilde{Q}$. Since $X(t) \in O(n)$, we must have $X(t)^T X(t) = I$ for all t. Differentiating with respect to t yields

$$\dot{X}(t)^T X(t) + X(t)^T \dot{X}(t) = 0 \tag{3.12}$$

and plugging in the initial condition yields

$$\dot{X}(t)^T \tilde{Q} = -\big(\dot{X}(t)^T \tilde{Q}\big)^T. \tag{3.13}$$

This means that for any smooth curve through $\tilde{Q}$, we must have $\dot{X}(t) = \tilde{Q}A$, where A is an $n \times n$ skew symmetric matrix [81]. Therefore the tangent space to $O(n)$ at $\tilde{Q}$ is given by

$$T_{\tilde{Q}} O(n) = \{\tilde{Q}A | A = -A^T\}. \tag{3.14}$$

Now define $\tilde{Q}(t)$ to be a geodesic curve through $\tilde{Q}$. Using the above equations, we find $\tilde{Q}(t) = \tilde{Q}e^{At}$ as the formula for a geodesic curve through $\tilde{Q}$ on $O(n)$ [80].

To find the geodesic formula on $Gr(p,n)$, we define the Grassmannian as a quotient of $O(n)$ based on the equivalence relation

$$[\tilde{Q}] = \left\{ \tilde{Q} \begin{bmatrix} Q_p & 0 \\ 0 & Q_{(n-p)} \end{bmatrix}, \quad Q_p \in O(p), Q_{(n-p)} \in O(n-p) \right\}. \tag{3.15}$$

Note that there exists a bijective mapping f between the equivalence class in equation (3.15) and the point $[Q] \in Gr(p,n)$ with $f([\tilde{Q}]) = [Q]$ and $f^{-1}([Q]) = [\tilde{Q}]$, where

$$Q = \begin{bmatrix} Q_p \\ Q(n-p) \end{bmatrix},$$

so $(O(n)/\sim) \cong Gr(p,n)$. Thus $Gr(p,n)$ has the quotient geometry from $O(n)$, and therefore $T_{[\tilde{Q}]}O(n)$ can be decomposed into orthogonal horizontal and vertical tangent spaces H_Q and V_Q, respectively, with $H_Q = T_{[Q]}Gr(p,n)$ [80]. The formula for the Grassmannian geodesic will therefore be developed on $O(n)/\sim$ and subsequently mapped to $Gr(p,n)$.

We derive a formula for the geodesic on H_Q by first defining V_Q. Let

$$V(t) = \tilde{Q} \begin{bmatrix} Q_p t & 0 \\ 0 & Q_{(n-p)} t \end{bmatrix} \tag{3.16}$$

be a curve in $O(n)$ along $[\tilde{Q}]$ with $V(0) = \tilde{Q}$. As before, we must have $V(t)^T V(t) = I$ for all t [81]. Taking the derivative with respect to t and plugging in the initial conditions yields

$$\dot{Q}_p^T Q_p + Q_k^T \dot{Q}_p = 0$$

and

$$\dot{Q}_{(n-p)}^T Q_{(n-p)} + Q_{(n-p)}^T \dot{Q}_{(n-p)} = 0.$$

The same reasoning as before yields

$$V_Q = \left\{ \tilde{Q} \begin{bmatrix} C & 0 \\ 0 & D \end{bmatrix} \right\}$$

where C is $p \times p$ skew symmetric and D is $(n-p) \times (n-p)$ skew symmetric [81]. Since H_Q must be orthogonal to V_Q with all elements of H_Q skew symmetric, it follows that

$$H_Q = \left\{ \tilde{Q}\tilde{A} \right\}$$

where

$$\tilde{A} = \begin{bmatrix} 0 & -B \\ B & 0 \end{bmatrix}$$

The geodesic on H_Q is then

$$\tilde{Q}(t) = \tilde{Q}e^{\tilde{A}t},$$

and therefore the matrix representation for a geodesic on $Gr(p,n)$ through $[Q]$ is

$$Q(t) = \tilde{Q}e^{\tilde{A}t} \begin{bmatrix} I_p \\ I_{(n-p)} \end{bmatrix}.$$

Because only the first p columns of $\tilde{Q}$ are important for numerical computation, we can rewrite the geodesic curve as

$$\Phi(t) = \tilde{Q}e^{\tilde{A}t} \begin{bmatrix} I_p \\ 0 \end{bmatrix}.$$

From Theorem 1 in [81], given $\Phi(0) = X$ and $\dot{\Phi}(0) = H$, we can write the geodesic curve using the compact SVD of the velocity matrix $H = U\Sigma V^T$ as

$$\Phi(t) = XV\cos(\Sigma t)V^T + U\sin(\Sigma t)V^T. \tag{3.17}$$

This provides a formula for a geodesic through $[X] \in Gr(p,n)$ given a known velocity matrix H, but the algorithms developed in this paper require a method to find H given two points $[X]$ and $[Y]$ in $Gr(p,n)$ such that $\Phi(0) = X$ and $\Phi(1) = YD \in [Y]$ (D being any $p \times p$ orthogonal matrix). This computation is found in [81], and it is reiterated here for the purpose of thoroughness.

At $t = 1$, the above requires that

$$YD = XV\cos(\Sigma t)V^T + U\sin(\Sigma t)V^T. \tag{3.18}$$

Left multiplication by X^T yields

$$X^TYD = V\cos(\Sigma)V^T \tag{3.19}$$

since $X^TU = 0$. Substituting X^TYD back into equation (3.18) yields

$$YD = XX^TYD + U\sin(\Sigma)V^T, \tag{3.20}$$

and combining equations (3.19) and (3.20) yields

$$U\sin(\Sigma)V^T(V\cos(\Sigma)V^T)^{-1} = U\tan(\Sigma)V^T = (I - XX^T)Y(X^TY)^{-1}. \tag{3.21}$$

Hence $U\Theta V^T$ is the SVD of H, with $\Theta = \arctan(\Sigma)$ and

$$H = (I - XX^T)Y(X^TY)^{-1}. \tag{3.22}$$

Using this formula for H and the SVD, equation (3.17) yields

$$\Phi(t) = XV\cos(\Theta t) + U\sin(\Theta t). \tag{3.23}$$

Together, equations (3.22) and (3.23) paramaterize a curve between any two points $[X]$ and $[Y]$ in $Gr(p, n)$.

3.2.3 Averaging Subspaces

The flag mean is an algorithm for computing averages of points on Grassmannians [67, 82, 83]. One can use such an algorithm to determine common attributes of a set of points on the Grassmannian as a set of nested subspaces, called a flag [82]. The algorithm, summarized below, is at the heart of the Grassmannian LBG procedure.

A flag is a nested sequence of subspaces. Given a finite collection of subspaces, the flag mean algorithm computes the best flag representation of the collection. The flag mean can be calculated for any dimension $r \leq p$ [67], and thus it is possible to consider a lower dimensional representation for subspaces of mixed dimensions, but that will not be the focus of this work.

Denote the flag by $\{[u_1], [u_2], ..., [u_r]\}$, where the u_i are orthogonal unit vectors with $r \leq p$. Let $\{[X_i]\}_{i=1}^m$ be a set of points in $Gr(p, n)$ and $\{X_i\}$ be their corresponding matrix representations. To construct the flag mean, iteratively solve the following optimization problem:

$$[u_j] = \underset{[u] \in Gr(1,n)}{\arg\min} \sum_{i=1}^{m} d_c([u], [X_i])^2, \quad \text{subject to } [u] \perp [u_l] \text{ for all } l < j \tag{3.24}$$

for $[u_1], ..., [u_r]$ [67]. From equation (3.9),

$$\underset{[u]}{\arg\min} \sum_{i=1}^{m} d_c([u], [X_i])^2 = \underset{[u]}{\arg\min} \sum_{i=1}^{m} (\sin(\theta_i))^2, \tag{3.25}$$

and the optimization problem is then equivalent to

$$[u_j] = \underset{[u] \in Gr(1,n)}{\arg\max} \sum_{i=1}^{m} (\cos(\theta_i))^2 \tag{3.26}$$

with the same constraints on the $[u_i]$ [67]. Consider the thin SVD of $u^T X_i$. Using the SVD,

$$u^T X_i X_i^T u = U\Sigma V^T V \Sigma^T U^T = \cos^2(\theta_i). \tag{3.27}$$

Combining equations (3.26) and (3.27) yields

$$[u_j] = \underset{[u]\in Gr(1,n)}{\arg\max}\, u^T \left(\sum_{i=1}^{m} X_i X_i^T \right) u, \quad \text{subject to } [u] \perp [u_l] \text{ for all } l < j. \tag{3.28}$$

For simplicity, let $A = \sum_{i=1}^{m} X_i X_i^T$. To find an optimal u_j, take the Lagrangian

$$\mathcal{L}(u, \lambda, \lambda_1, ..., \lambda_{j-1}) = u^T A u - \lambda(u^T u - 1) - \sum_{l=1}^{j-1} \lambda_l (u^T u_l) \tag{3.29}$$

and its partial derivatives. Setting the partials equal to zero yields the optimality conditions

$$Au = \lambda u, \quad u^T u = 1, \quad u^T u_l = 0, \tag{3.30}$$

and the problem is reduced to an eigenvector computation [67]. Given that in most cases we have $p << n$, it is desirable to find more efficient method than standard eigenvector computations that have complexity $\mathcal{O}(n^3)$. Define the concatenation of the matrix representations of the $\{[X_i]\}$ by $\mathbf{X} = [X_1, ..., X_m]$. Then $\mathbf{X}$ is an $n \times (mp)$ matrix, and $\mathbf{X}\mathbf{X}^T = \sum_{i=1}^{m} X_i X_i^T = A$. The thin SVD of $\mathbf{X} = U\Sigma V^T$ then yields $\mathbf{X}\mathbf{X}^T = U\Sigma\Sigma^T U^T$, and the columns of U are therefore the eigenvalues of $\mathbf{X}\mathbf{X}^T$ with corresponding eigenvectors σ_i^2 [67]. Using the SVD for this calculation reduces the order to $\mathcal{O}(nm^2p^2)$ and is therefore more efficient whenever $mp < n$, which is quite often the case.

The flag mean acts as a representative for the subspaces $[X_i]$. Figure 3.1 depicts an MDS embedding of subspaces generated data from using the MNIST handwritten digit database [68]. Here, points in $Gr(5, 784)$ were constructed from images of handwritten "2"s using the method described in Section 3.1. The center is the 5-dimensional flag mean calculated according to Equation 3.24. An interesting property of the flag mean is its ability to concisely capture information about not only the average, but the most common variations within data. Figure 3.2 shows, in order, the five component vectors of the flag mean reshaped and colored as images. Clearly, the first

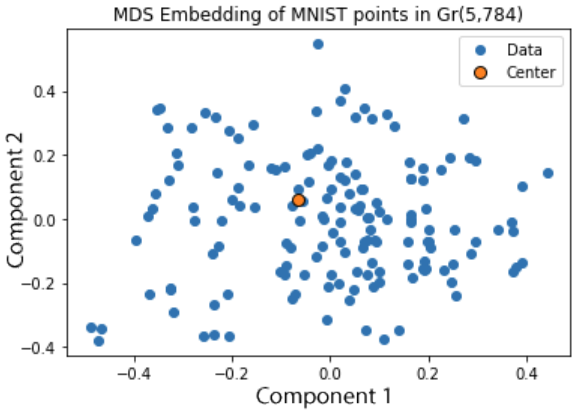

Figure 3.1: MDS embedding of handwritten digit "2" from $Gr(5, 784)$ and corresponding flag mean center.

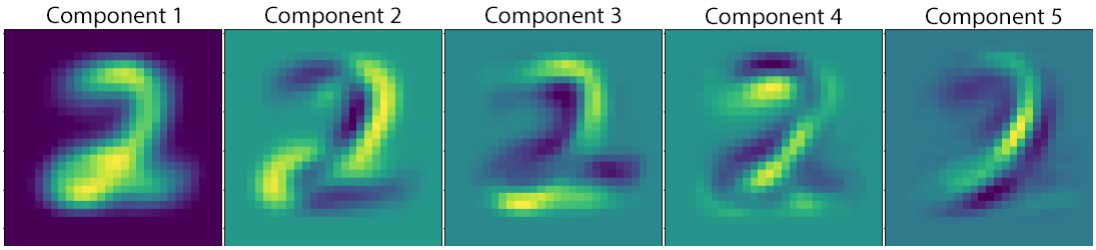

Figure 3.2: The five ordered component vectors from the flag mean, reshaped and visualized.

component depicts the average (i.e. most common) variant of the handwritten digit. The remaining components depict, in order from most to least frequent, common variations from the average. In this way, the flag mean captures visual information about within-cluster variations that can provide additional insight into data.

Chapter 4

Euclidean Algorithms

Here we provide an overview of the two partitional clustering algorithms central to this thesis. Both are derived from the optimization strategies discussed in Sections 2.2 and 2.3 and define clusters according to Voronoi cells about each center as defined in equation (2.3).

4.1 K-means

MacQueen's K-means algorithm operates on an incoming stream of data, assigning each data point to its nearest center [14]. The chosen center is then updated in the direction of the new data point.

Let x_{t+1} be the $(t+1)^{st}$ data point assigned to a center c. The center at time t, denoted c_t, is then updated by

$$c_{t+1} = c_t + \frac{1}{t}(x - c_t) = \frac{tc_t + x_{t+1}}{t+1}. \tag{4.1}$$

An illustration of the update process is depicted in Figure 4.1, and Algorithm 1 describes the steps for performing K-means on a data set.

Note that if the initial center c_0 is updated using the first point x_1, then

$$c_1 = \frac{0c_0 + x_1}{1}$$

so each center is automatically set to the first point it sees. A second update yields

$$c_2 = \frac{c_1 + x_2}{2} = \frac{x_1 + x_2}{2} = \frac{1}{2}\sum_i = 1^2 x_i.$$

More generally, if $c_t = \frac{1}{t}\sum_{i=1}^{t} x_i$, then

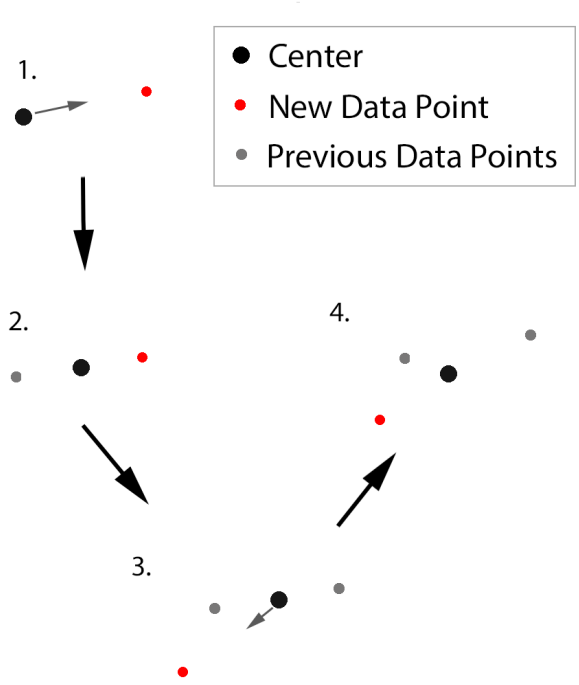

Figure 4.1: Illustration of center updates in the K-means algorithm.

$$c_{t_1} = \frac{t\frac{1}{t}\sum_{i=1}^{t} x_i + x_{t+1}}{t+1} = \frac{1}{t+1}\sum_{i=1}^{t+1} x_i$$

Each center is thus the average of all points previously assigned to it. Because K-means updates centers without seeing all the data, the final centers are not necessarily the average of the data points contained in their Voronoi set, depending on how far the center moved from its original location. Algorithm 1 describes the steps for iterating through data points and updating centers.

Algorithm 1 Euclidean K-means

1: Initialize k centers in the data space.
2: Set the count for each center to zero.
3: **for** each data point **do**
4: Select the center nearest to the data point.
5: Add 1 to the count for that center.
6: Update the center according to equation (4.1) with t = current count.

Because data is received as a stream, K-means is especially sensitive to starting conditions and the order the data is presented. This can be mitigated somewhat by iterating through all data points repeatedly until the cluster distortion stabilizes. Though this is certainly impractical for online analysis of incoming data, we will implement this method in order to provide more meaningful comparisons with the LBG algorithm. A single pass of K-means through all data points is referred to as an epoch. Algorithm 2 describes the steps for the full epoch version of K-means.

Algorithm 2 Euclidean K-means with Epochs

1: Initialize cluster distortion to infinity.
2: Set termination threshold.
3: Initialize k centers in the data space.
4: Set the count for each center to zero.
5: **for** each data point **do**
6: Select the center nearest to the data point.
7: Add 1 to the count for that center.
8: Update the center according to equation (4.1) with t = current count.
9: **end for**
10: Calculate new cluster distortion.
11: Calculate change in cluster distortion.
12: Compare distortion change to specified threshold.
13: **if** distortion change is greater than threshold **then** repeat steps 5-13.

4.2 LBG

LBG is a K-means-type algorithm initially developed for performing vector quantization by Linde, Buzo, and Gray [7]. LBG differs from the MacQueen K-means algorithm in that it considers the entire data set as a whole and performs batch updates rather than updating based on one point at a time. This algorithm uses Euclidean squared distances to calculate and update the Voronoi cells. Figure 4.2 illustrates a single update step for a pair of clusters associated with two centers. Algorithm 3 describes the process.

As with K-means, LBG suffers a sensitivity to starting conditions. The original LBG paper presents a method for optimally initializing centers [7], and additional methods are explored in

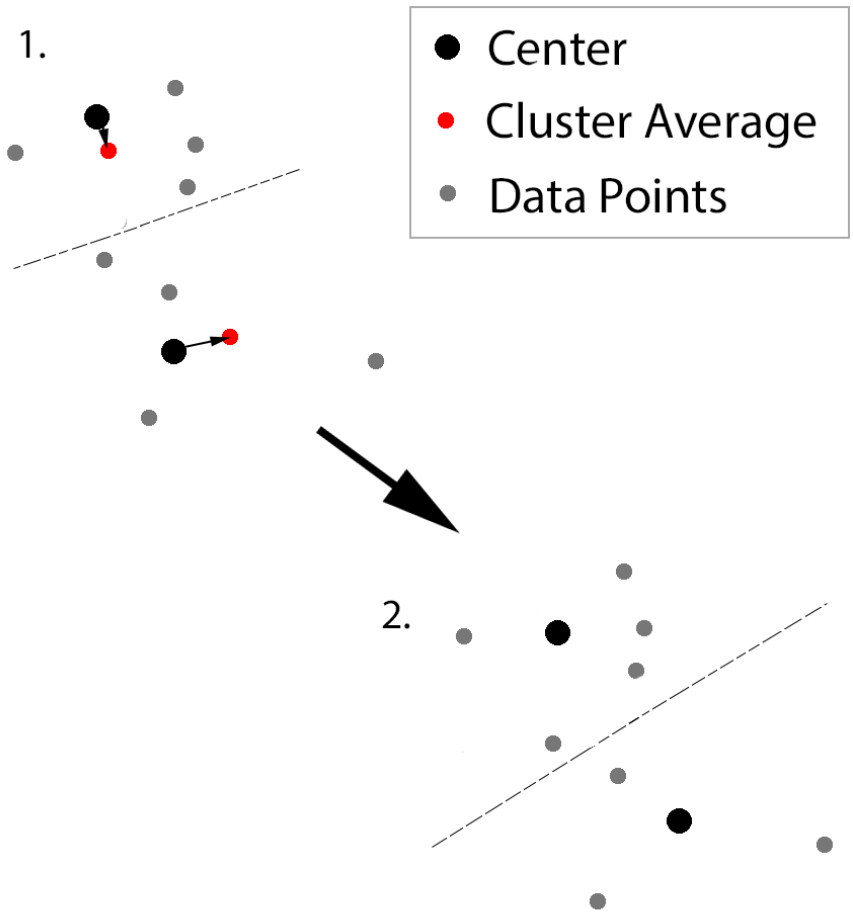

Figure 4.2: An illustration of cluster updates in LBG.

Algorithm 3 Euclidean LBG

1: Initialize cluster distortion to infinity.
2: Select termination threshold.
3: Initialize k centers in the data space.
4: Assign each data point to a center by constructing Voronoi cells S_i about each center.
5: Update each center c_i by averaging the data in S_i.
6: Reassign all data points to Voronoi cells defined by updated centers.
7: Calculate new cluster distortion.
8: Calculate change in cluster distortion.
9: Compare distortion to specified threshold.
10: **if** distortion change is greater than threshold **then** repeat steps 5-10.

[84, 85]. As with K-means, these additions to the algorithm are beyond the scope of this work. To account for the possibility of poor starting conditions, each experiment for both methods is run multiple times, with ending distortion error used to choose the best possible results. Using the Euclidean metric to quantize implicitly assumes that the cells will be isotropic; however, this may not be the case, especially in high dimensional spaces with noise [86].

4.3 Properties and Complexity

Clearly, for a single cluster of m data points, the end result of one epoch of K-means is equivalent to one iteration of LBG since both algorithms will return the average of all points in the cluster. The primary difference between the two algorithms is only apparent in the presence of more than one cluster. After the final iteration of LBG, all centers will be exactly the means of their assigned data points, and the optimality conditions described in Section 2.2 are satisfied. After one epoch of K-means, however, the centers are not guaranteed to be the average of the data points currently assigned to them. Consider the case where a point x_i is initially assigned to center c_1, but as the algorithm updates, c_1 moves away from x_i and c_2 moves nearer to x_i. If, at the end of the epoch, c_2 is closer to x_i than c_1, x_i will be assigned to c_2 even though it did not contribute to the updates of that center. Since each center is the average of all points that it sees during an epoch, c_1 will be the average of some subset of data points including x_i, and c_2 will be the average of a distinct set of data points that does not include x_i. As a result, the optimality conditions given by Gersho and Gray are not necessarily satisfied. This can be remedied by setting each center to the centroid of its corresponding cell after the K-means algorithm terminates, but this is not practical in cases where data is received in real time. Thus the centers selected by LBG and K-means for multiple clusters on the same data set may differ depending on the starting conditions for the K-means algorithm.

It is also worth noting that the optimality conditions given by Gersho and Gray only guarantee local optimality of a partition, not global [36]. Therefore, depending on starting conditions, it is possible for both algorithms to terminate in a local minimum that is not the global minimum, so repeated trials are necessary to accurately divide a data set.

Given k total centers and m total data points, a single iteration of LBG requires km pairwise distance calculations to assign data points to centers, and then another m calculations to compute the new centers. A single epoch of K-means requires k pairwise distance calculations for every data point, totalling km. Center updates require 3 operations per data point, for a total of $3m$. Given that a single pairwise Euclidean squared distance calculation in $\mathbb{R}^n$ requires $n^2(n-1)$ floating point operations (FLOPS), this step has the highest computational cost and thus both the LBG iteration and the K-means epoch have complexity $\mathcal{O}(kmn^2(n-1))$. Therefore the cost of a single iteration of LBG is equivalent to that of a single epoch of K-means.

Chapter 5

Grassmannian Algorithms

The algorithms in this section are developed by translating each step of the Euclidean algorithms to the Grassmann manifold.

5.1 Grassmannian K-means

To adapt the K-means algorithm to function on Grassmann manifolds, we use the parameterization of the geodesic in equation (3.23) to update each center in the direction of novel data points. Given data as a collection of points in $\mathbb{R}^n$, we construct matrix representations of p-dimensional subspaces as discussed in Section 3.2. We then use the adapted K-means algorithm to partition the data into k clusters in $Gr(p, n)$. Cluster distortion is calculated according to equation (2.4) using the chordal distance metric, though it is also possible to substitute geodesic distances or the smallest principal angle pseudometric. To update a center in the direction of a new point, we use equation (3.23) and set t equal to the inverse of the total number of points assigned to that center, including the newest one. Steps are detailed in Algorithm 4.

Algorithm 4 Grassmannian K-means

1: Construct matrix representations for data in $Gr(p, n)$.
2: Initialize cluster distortion to infinity.
3: Select termination threshold.
4: Initialize k centers on the manifold.
5: Set the count for each center to zero.
6: **for** each data point **do**
7: Select the center nearest to the data point according to a Grassmannian metric.
8: Add 1 to the count for that center.
9: Update the center according to equation (3.23) with $t = 1$ / (current count).
10: **end for**
11: Calculate new cluster distortion using chosen Grassmannian metric.
12: Calculate change in cluster distortion.
13: Compare change in distortion to specified threshold.
14: **if** distortion change is greater than threshold **then** repeat steps 6-14.

5.2 Grassmannian LBG

To adapt the LBG algorithm for use on the Grassmannian, we replace the traditional Euclidean average with the flag mean from equation (3.24) to average all subspaces contained in each Voronoi set. As with Grassmann K-means, data in $\mathbb{R}^n$ is used to construct matrix representations for subspaces in $Gr(p, n)$ as previously discussed in Section 3.1. Algorithm 5 outlines the procedure.

Algorithm 5 Grassmannian LBG

1: Construct matrix representations for data in $Gr(p, n)$.
2: Initialize cluster distortion to infinity.
3: Select termination threshold.
4: Initialize k centers on the manifold.
5: Assign each data point a center by constructing Voronoi cells S_i about each center using a Grassmannian distance metric.
6: Update each center c_i by calculating the flag mean of S_i as in equation (3.24).
7: Reassign all data points to Voronoi cells defined by updated centers.
8: Calculate cluster distortion using chosen metric.
9: Calculate change in cluster distortion.
10: Compare change in distortion to specified threshold.
11: **if** distortion change is greater than threshold **then** repeat steps 6-11.

5.3 Properties and Complexity

Figure 3.2 in Section 3.2.3 demonstrated the ability of the flag mean to capture information in an ordered fashion. Because all centers in Grassmann LBG are calculated as flag means of Voronoi cells, this property is also present in centers selected by this algorithm. To test whether this is also the case in the Grassmann K-means algorithm, the algorithm was run on the same set of handwritten digits in $Gr(5, 784)$ pictured in Figure 3.1. The five components were then visualized in order and compared to the LBG components. Figure 5.1 shows the results. Although K-means captures much of the same information, it does so much less cleanly than LBG, and there is no apparent order to the components.

Note that component 5 of the K-means center has an inverted color scheme from that of the other pictured centers. This is because the matrix representation for the center is scaled by a factor of -1 relative to the matrix representations for the others. However, because we are considering subspaces as points, this representation is equivalent to the one resulting in "standard" coloration.

To compare the computational complexity of each algorithm, one K-means epoch is compared to a single LBG iteration. A K-means epoch includes iteration through m data points in $Gr(p, n)$, with each iteration requiring a pairwise distance calculation between a single data point and the k centers, label and count updates, a center update using geodesic parameterization from equation (3.23), and a QR-factorization of the updated center to restore orthonormality. For LBG, one iteration consists of a pairwise distance calculation between the k centers and all m data points, label updates for each point, and k flag mean calculations that each include varying numbers of data points. For both algorithms, the cluster distortion check is eliminated from the calculation since it is identical regardless of the selected algorithm and not considered part of the update step.

Let m be the number of data points in $Gr(p, n)$ with k the number of centers used in the algorithm. The complexity of pairwise distance is required for both algorithms, and will thus be

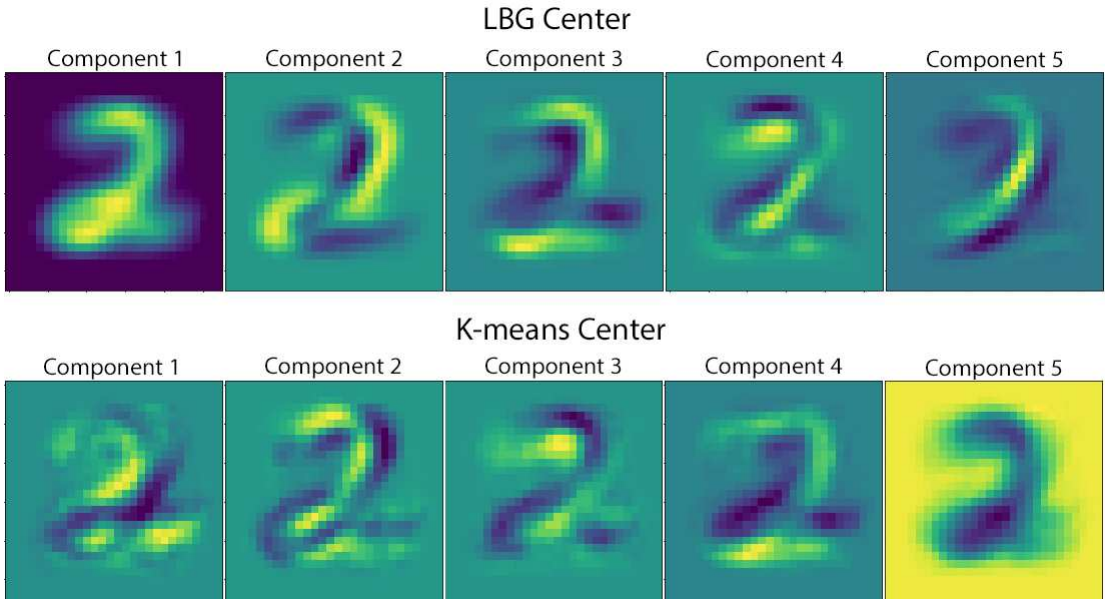

Figure 5.1: Comparison of components of centers from LBG and K-means.

calculated prior to analysis of the algorithms for all three distance measures used in this work: squared chordal distance, squared geodesic distance, and smallest principal angle distance.

Pairwise chordal distance calculations take as an input a set of k subspaces $\{[X_i]\}$ and a set of m subspaces $\{[Y_j]\}$. The distance for a single pair $[X_i], [Y_j]$ is given by

$$d_{ij} = \| \sin\theta \|_c^2 \sum_{q=1}^{p} \left(1 - \cos^2(\theta_q) \right) = \|$$

where the $\cos(\theta_q)$ are the p singular values obtained from the SVD of $X_i^T Y_j$. For the purposes of these calculations, this SVD is assumed to be a full SVD. The highest cost step of this pairwise distance calculation is the matrix multiplication and subsequent SVD. Since both X_i and Y_j are $p \times n$ matrices, the multiplication step totals $p^2(2n-1)$ FLOPS. A full SVD on an $n \times m$ matrix has $\mathcal{O}(n^2m + nm^2 + m^3)$ FLOPS. Because $X_i^T Y_j$ results in a $p \times p$ matrix, this simplifies to $\mathcal{O}(p^3)$. The remaining calculation of d_{ij} has complexity of $2p$, and is therefore disregarded. Repeating the calculation km times results in a complexity of $\mathcal{O}(knmp^2 + kmp^3)$, plus lower order terms.

The geodesic pairwise distance calculation takes the same sets of k and m subspaces. The distance for a single pair is given by

$$d_{ij} = \sum_{q=1}^{p} \theta_q^2$$

where the $\theta_q = \arccos(\cos(\theta_q))$ are again obtained from the SVD of $X_i^T Y_j$. The same multiplication and SVD steps found in the chordal distance again total $p^2(2n-1)$ and $\mathcal{O}(p^3)$, respectively. Both the $\arccos$ and remaining additions have lower relative costs, so the computational complexity is again $\mathcal{O}(knmp^2 + kmp^3)$.

The smallest principal angle pseudometric only requires the first angle of the SVD of $X_i^T Y_j$. Again, multiplication and the SVD yield the same number of steps. The only remaining operation is taking the $\arccos$ of the smallest angle, so overall order again reduces to $\mathcal{O}(knmp^2 + kmp^3)$.

Because the highest cost computational steps for each distance measure are the same, the overall cost for both Grassmannian LBG and Grassmannian K-means is not affected by the choice of metric.

The two high cost steps in a Grassmannian K-means epoch are calculating chordal distances and updating the center via geodesic parameterization. The calculation of the velocity matrix H requires inversion of a $p \times p$ matrix, n^2 subtractions, a multiplication of a $p \times n$ matrix and an $n \times p$ matrix, another multiplication of a $n \times p$ and a $p \times n$ matrix, and multiplications of $n \times n$, $n \times p$, and $p \times p$ matrices. Assuming the inversion step is done by Gaussian elimination, the complexity of this step is $\mathcal{O}(p^3)$. The four matrix multiplication steps require $2np^2 - p^2 + 2n^2p - n^2 + 2np^2 - np + 2n^2p - np$ total FLOPS, which can be reduced to $\mathcal{O}(np^2 + n^2p - n^2 - p^2)$ by eliminating lower order terms. Combining this with the inversion and subraction operation counts and eliminating terms yields $\mathcal{O}(p^3 + pn^2 + np^2)$ required FLOPs for this step. Next, a truncated SVD is performed on the $n \times p$ matrix H, which requires $\mathcal{O}(np^2 + p^3)$ operations. The final step requires multiplication of $n \times p$ and $p \times n$ matrices and an $n \times p$ subtraction operation. After eliminating lower order terms, the complexity of this step is $\mathcal{O}(np^2 + p^3)$. The final QR decomposition of an $n \times p$ matrix adds an additional $2np^2$ FLOPs. Combining all the above steps and eliminating low order terms yields a total complexity of $\mathcal{O}(p^3 + np^2)$ for the center update. Since chordal distance is only calculated using a single data point, the complexity of that step becomes $\mathcal{O}(kp^2n + kp^3)$. Repeating these steps for m data points gives a total computational complexity of $\mathcal{O}(mp^3 + mnp^2 + mkp^3 + nmkp^2)$ for a single epoch of Grassmann K-means with m data points and k centers in $Gr(p, n)$.

For Grassmann LBG, the high cost steps are calculation of pairwise distances between the k centers and m data points and the k required flag mean calculations. The flag mean step introduces the added complication of generalizing a computational cost for averaging a varying number of data points, but this can be addressed. First, suppose we are averaging m subspaces in $Gr(p, n)$. The only required calculation is a truncated SVD of the horizontally concatenated $n \times pm$ matrices. The cost for a thin SVD of an $n \times pm$ matrix is $\mathcal{O}(nm^2p^2 + m^3p^3)$. Now suppose we have two centers with l and q points respectively. Then the computational cost for both centers is

$$\mathcal{O}(nl^2p^2 + l^3p^3 + nq^2p^2 + q^3p^3) = \mathcal{O}(m(l^2 + q^2)p^2 + (l^3 + q^3)p^3) \leq \mathcal{O}(m(l + q)^2p^2 + (l + q)^3p^3)$$

so for m data points, the k flag mean calculations can have a complexity of at most $\mathcal{O}(nm^2p^2 + m^3p^3)$. Adding this to the pairwise chordal distance cost yields a cost of $\mathcal{O}(nm^2p^2 + m^3p^3 + nmkp^2 + mkp^3)$ for a single iteration of Grassmann LBG with m data points and k centers.

In general, we can assume that $p << n$ for most problems. Since K-means is linear in n, m, and k while LBG contains an m^3 term, K-means will be far less computationally expensive than LBG, especially for large data sets. However, K-means does not provide the same data characteristics that LBG does. Complexity of steps in both algorithms could be further optimized in future work.

Because the optimality conditions are defined for any metric space, conditions for optimality in Euclidean algorithms also apply to algorithms on the Grassmann manifold.

Chapter 6

Experimental Results

Here, we provide both a qualitative demonstration and a more formal evaluation of Grassmannian K-means and Grassmannian LBG.

We use two measures to evaluate unsupervised classification accuracy. The first is evaluation of the average purity of all clusters upon termination of the algorithm. Let $y(x) = v$ be the true label for a point x_i. The average purity for k clusters C_i, $1 \leq i \leq k$, given class labels L is

$$P = \frac{1}{k} \sum_{i=1}^{k} \left(\frac{1}{|C_i|} \max_{x \in C_i, v \in L} |y(x) = v| \right). \tag{6.1}$$

This measure describes how cleanly centers partition distinct groups of data by quantifying the fraction of each cluster occupied by points from a single class. However, the average across all clusters can be heavily skewed by clusters with a very small number of points. To account for this, we use the classification accuracy described in Section 2.5 that assigns each cluster the label corresponding to the largest fraction of points in the cluster as our second measure. Accuracy is then determined according to the

$$A = \frac{1}{m} \sum_{i=1}^{k} \sum_{x \in C_i} 1_{y(x)=y(C_i)} \tag{6.2}$$

where m is the total number of data points, 1 is the indicator function, and $y(C_i)$ is the label assigned to cluster C_i. This gives the total fraction of the data that was assigned the correct label, and large discrepancies this and cluster purity allow for quick identification of skewed clustering. This accuracy is used to compare results from Grassmannian K-means and Grassmannian LBG on the MNIST 10-class clustering problem to results found in the literature.

In addition to the previous measures, total cluster distortion at algorithm termination is also reported. Although this is useful for comparing results from several trials within a single experiment

or results from two algorithms applied to the same data, it is not suitable for direct comparison of trials that are performed on different manifolds or that use different metrics. Distortion is primarily reported here for direct comparison of Grassmannian LBG and Grassmannian K-means when applied to the same experiment with identical k values. In this situation, it provides useful information about which algorithm terminates nearest the true minimum and how they compare in stability across multiple trials.

Cluster purity and classification accuracy are reported as a percentage. Distortion errors are reported as raw numbers calculated from the distance metric.

6.1 Testing on Toy Data

Prior to more rigorous comparison on the benchmark data sets, both Grassmannian LBG and Grassmannian K-means were tested on simple, highly separable clusters on various data manifolds. These tests serve to demonstrate the functionality of both algorithms and provide some qualitative insight into cluster structure on the Grassmannian.

6.1.1 Epsilon Balls on Gr(1, 3)

The Grassmannian $Gr(1,3)$ is the parameterization of the set of all lines in $\mathbb{R}^3$ passing through the origin. It is also referred to as the real projective space $\mathbb{RP}^2$. It is one of three Grassmannians that can be explicitly visualized in three dimensions. This is accomplished by projecting each line onto the upper hemisphere of the unit circle and plotting them as points. Because of this useful visualization method, $Gr(1,3)$ is used to test the Grassmannian LBG and K-means algorithms and provide insight into the manifold structure.

To test the functionality and effectiveness of the algorithms, clusters were generated in $Gr(1,3)$ by randomly selecting a single point on the manifold, then constructing an ϵ-ball about this point using a Gaussian normal distribution. The value of ϵ was varied, along with size and number of clusters, to thoroughly test the robustness of these algorithms. Although both K-means and LBG are known to be sensitive to initial starting conditions, that issue did not appear in any of these

tests. Here, three variations of the ϵ-ball trial are reviewed. All clusters are visualized on the upper unit hemisphere in $\mathbb{R}^3$, and then both data and centers are isometrically embedded into $\mathbb{R}^2$ using chordal distances.

The first trial used three clusters containing ten points each in $Gr(1,3)$ with $\epsilon = 0.1$. Both algorithms were applied to the data using $k = 3$, and both successfully identified the centers. The clusters for each algorithm are seen in Figures 6.1 and 6.2 along with MDS embeddings of the clusters and selected centers. A second trial tested the two Grassmannian algorithms on four distinct clusters with 100 points each, generated using $\epsilon = 0.1$. All clusters were separated with 100% accuracy by both algorithms using $k = 4$. Clusters on the unit hemisphere and MDS embeddings into $\mathbb{R}^2$ are found in Figures 6.3 and 6.4. The final trial tested two large clusters

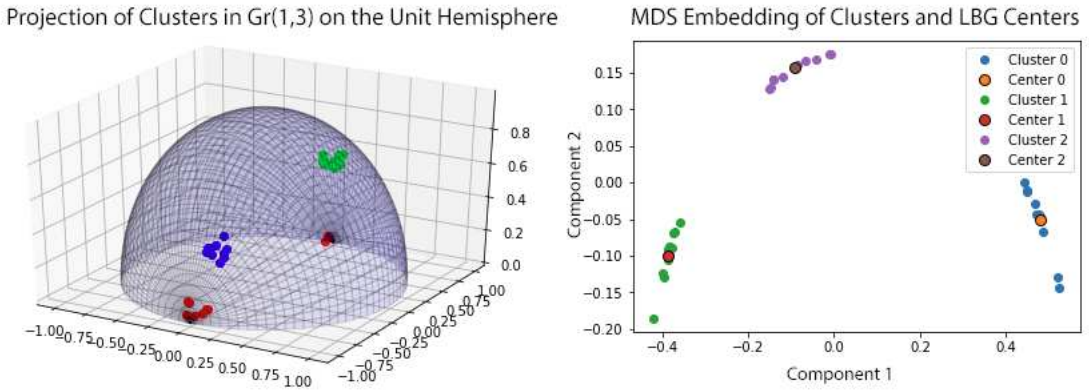

Figure 6.1: Clusters in $Gr(1,3)$ and their two-dimensional MDS embeddings with centers selected using Grassmannian LBG.

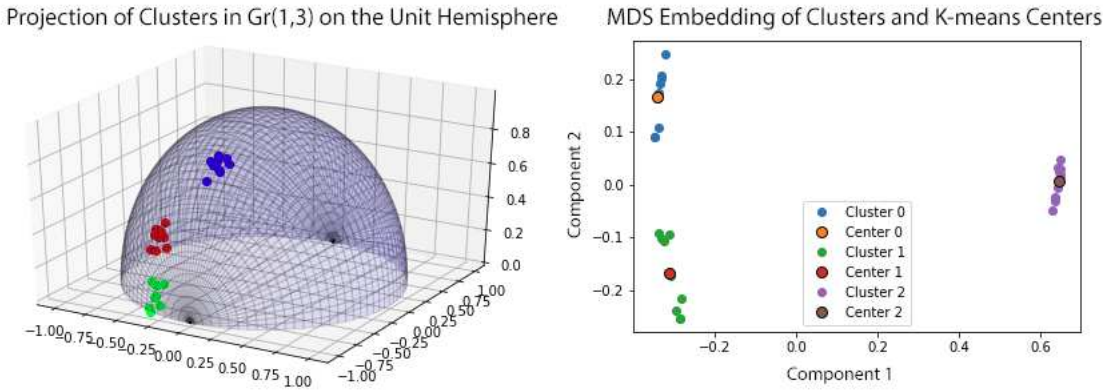

Figure 6.2: Clusters in $Gr(1,3)$ and their two-dimensional chordal distance MDS embeddings with centers selected using Grassmannian K-means.

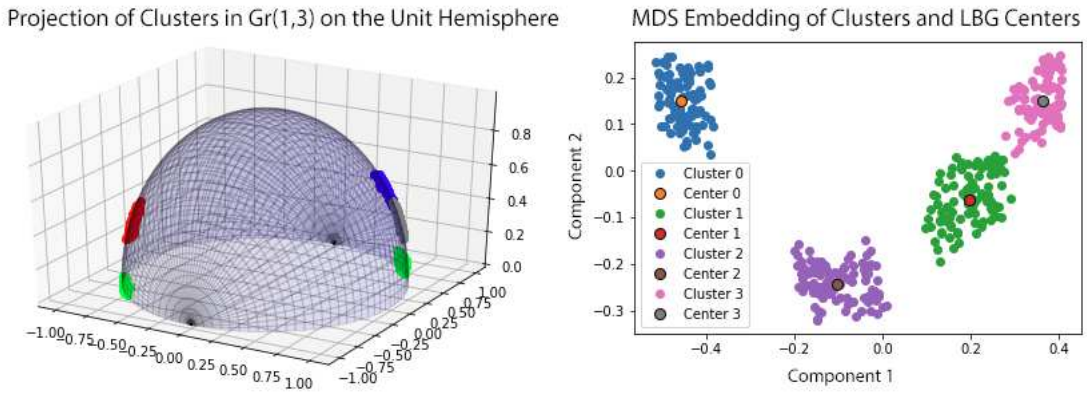

Figure 6.3: Larger clusters in $Gr(1,3)$ and their two-dimensional chordal distance MDS embeddings with centers selected using Grassmannian LBG.

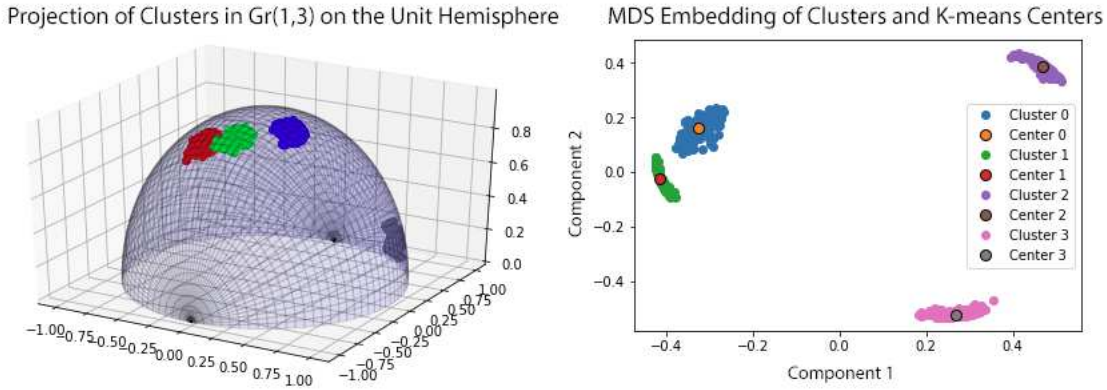

Figure 6.4: Larger clusters in $Gr(1,3)$ and their two-dimensional chordal distance MDS embeddings with centers selected using Grassmannian K-means.

using 100 points generated with $\epsilon = 0.5$ so that each cluster covered a large portion of the unit hemisphere while still being completely separable. Figures 6.5 and 6.6 show the representations of clusters in $Gr(1,3)$ along with the two-dimensional embedding of the clusters and centers selected by each Grassmannian algorithm. Again, both algorithms separated the clusters perfectly using $k = 2$.

It is, however, interesting to observe that as the clusters spread to cover more of the unit hemisphere, their centers appear to move closer to the outside edges of the MDS embedding plot. This phenomenon is further explored in the next section.

As an aside, the left-most images in Figures 6.1, 6.3, and 6.6 show an interesting geometric property of the Grassmannian. Because all points along a line are identified with each other by

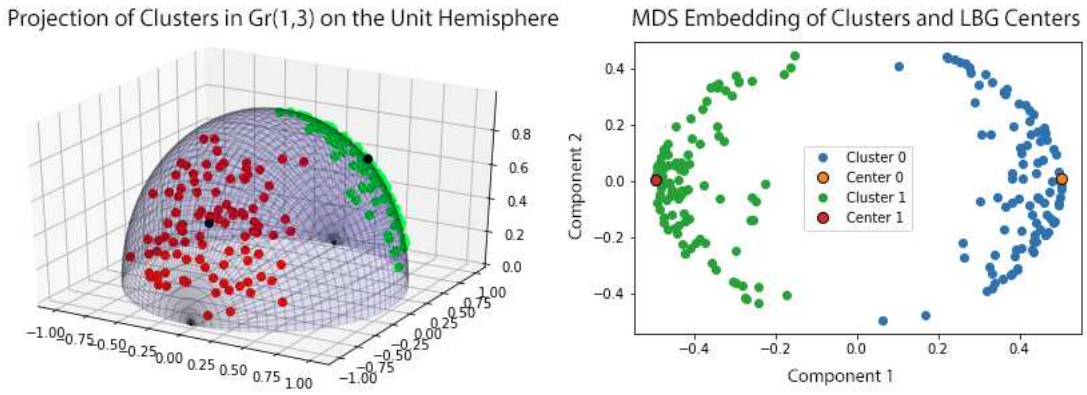

Figure 6.5: Two large ϵ-balls in $Gr(1,3)$ and their two-dimensional chordal distance MDS embeddings with centers selected using Grassmannian LBG.

the equivalence relation, points on opposite sides of the base of the unit hemisphere are considered identical. In these figures, a single cluster appears to be split across the hemisphere, but the identification of points on the circumference of the base means that the two "pieces" are still very close together. This is evident in the corresponding MDS embeddings of these clusters, since the pairwise distances are based on angles between the lines in $\mathbb{R}^3$.

6.1.2 Epsilon Balls on Grassmannians of Higher Dimension

The manifold $Gr(1,3)$ has dimension $d = 2$, so embedding into $\mathbb{R}^2$ for visualization is fairly trivial. Higher dimensional Grassmannians are more difficult to accurately visualize. Additionally, experimentation suggests that as the dimension of the Grassmannian grows, cluster centers move

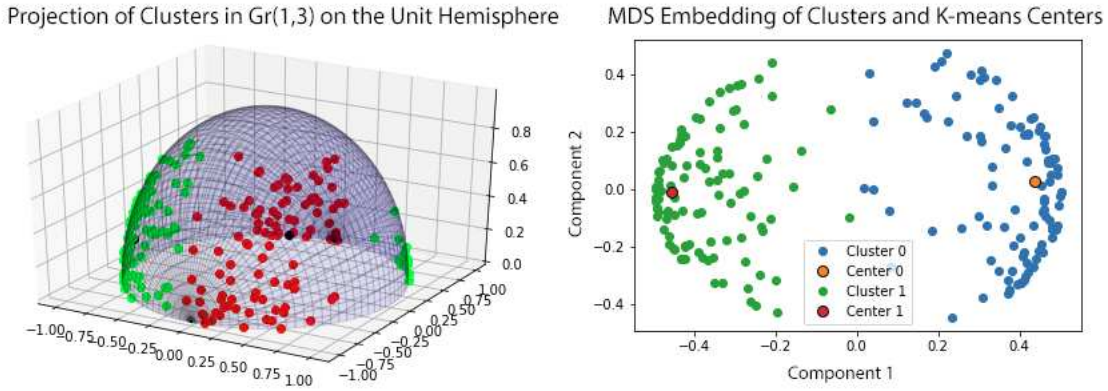

Figure 6.6: Two large ϵ-balls in $Gr(1,3)$ and their two-dimensional chordal distance MDS embeddings with centers selected using Grassmannian K-means.

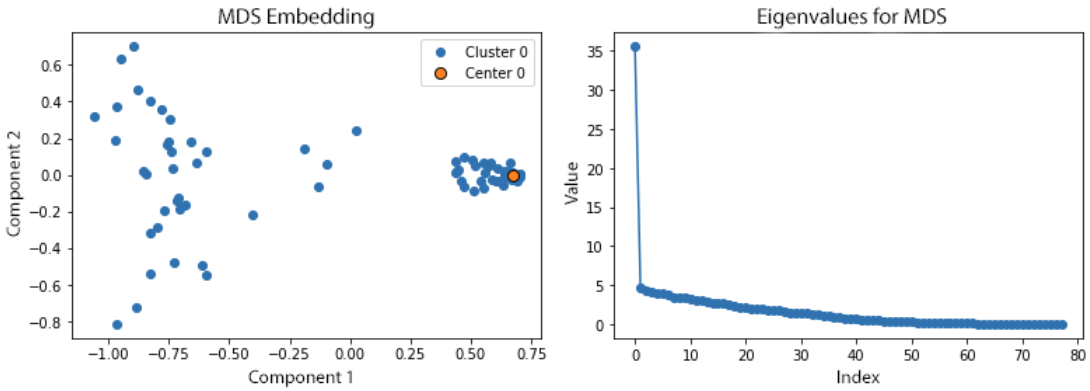

Figure 6.7: Chordal distnace MDS embedding of single ϵ-ball in $Gr(2, 10)$ with its center plotted.

progressively further from the center of the embedding. This appears to be less of an issue in localized clusters (i.e. generated by smaller ϵ values), but Figure 6.5 shows a clear example of centers sitting at the outer edges of the embedding when the clusters spread further across the manifold. In this section, we explore this using clusters in higher dimensional Grassmannians and test both algorithms for robustness in high dimensions. All embeddings are based on the chordal distance metric to preserve the data structure.

Figure 6.7 depicts the isometric MDS embedding of a single ϵ-ball in $Gr(2, 10)$. This Grassmannian has dimension $d = 16$. The ball was generated by varying ϵ from 0.01 to 1 in order to cover the majority of the manifold while maintaining a denser cluster of points around the center. The ball contains a total of 76 points. The eigenvalues of the MDS embedding indicate that most of the variation in the cluster is captured by the first MDS component, therefore the embedding is a good representation of the relative shape of the cluster. Here, the true center is plotted, and appears near the edge of the dense cluster of points on the right hand side of the embedding. The same strategy was used to generate an ϵ-ball in $Gr(20, 200)$, which has $d = 3600$. This embedding and the eigenvalues from MDS are depicted in Figure 6.8. Again, points are more tightly clustered near the center, and the center itself appears to be at the extreme edge of the cluster.

Figure 6.9 contains four separate ϵ-balls in $Gr(20, 200)$. These were generated by varying ϵ between 0.01 and 0.1, which still allows for distinct separate clusters. Again, the centers of each algorithm reside at the "end" of the embedding for each cluster. In Figure 6.10, points were added

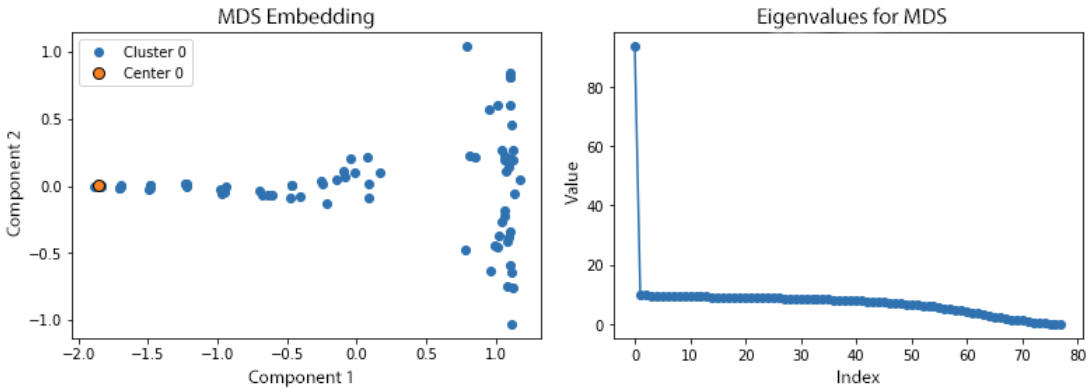

Figure 6.8: Chordal distance MDS embedding of single ϵ-ball in $Gr(20, 200)$ with its center plotted.

to the balls in Figure 6.9 using ϵ values up to 1, which resulted in all four clusters extending far enough to overlap one another on the manifold. However, the linear appearance of each ball remains everywhere in the embedding except for the area of overlap. The eigenvalues for this MDS embedding now indicate that most of the variance information is captured in the first three components. Adjusting the viewing angle in three dimensions shows that the centers of the four clusters sit in a tetrahedral shape when embedded in $\mathbb{R}^3$. Figure 6.10 includes tests of both Grassmannian K-means and Grassmannian LBG on the four clusters. Here, both algorithms closely approximate the true centers, and it is easily seen that these centers lie at the extreme outer edges of their corresponding clusters.

Figure 6.11 contains four clusters in $Gr(2, 10)$ generated in the same manner as those in Figure 6.10. The eigenvalues of MDS suggest once again that most of the variation is captured in the first

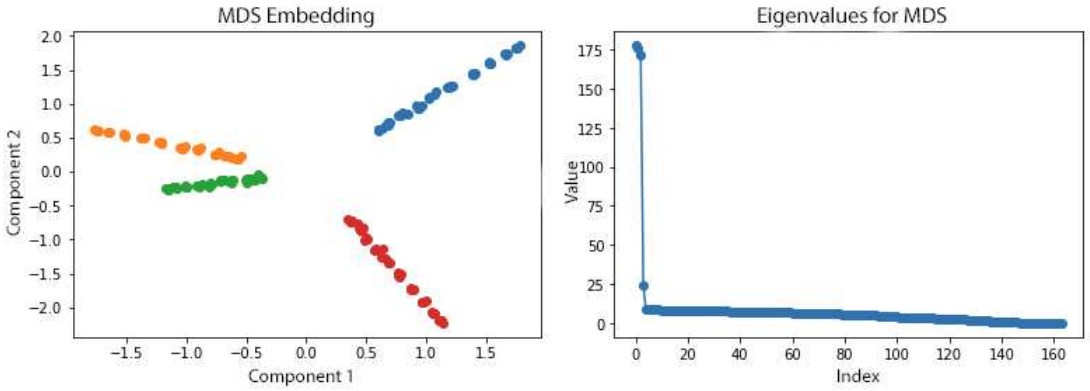

Figure 6.9: Chordal distance MDS embedding of four ϵ-balls in $Gr(20, 200)$ with $\epsilon \leq 0.1$.

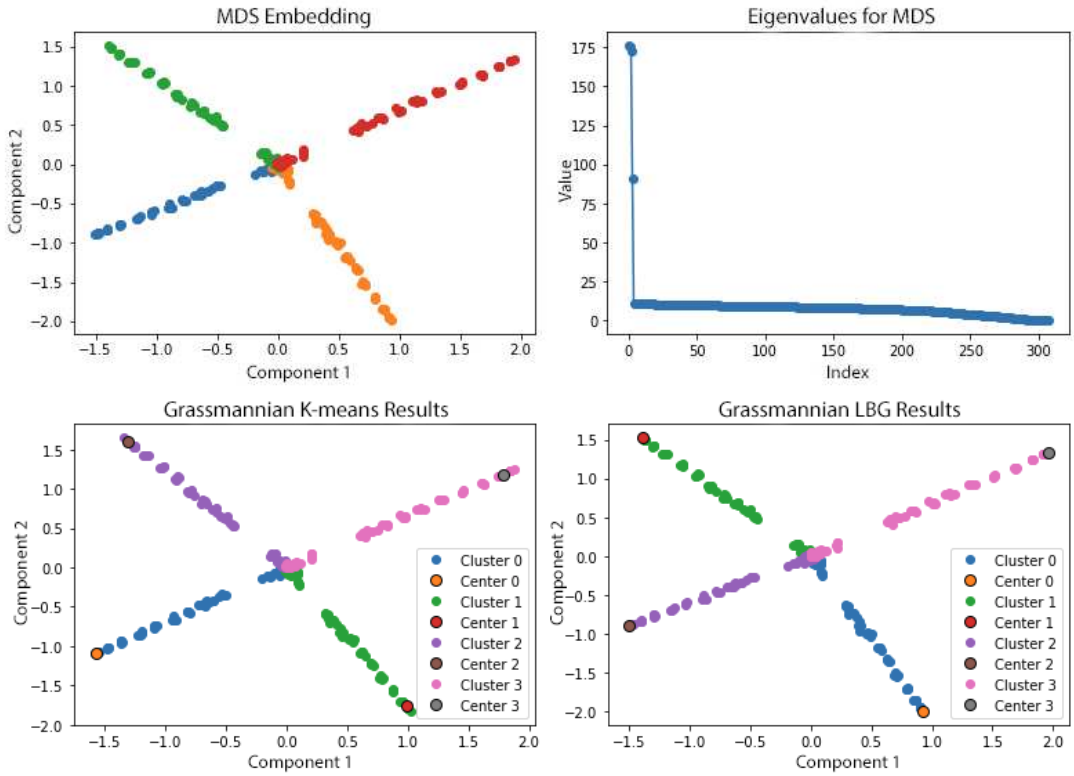

Figure 6.10: Chordal distance MDS embedding of four ϵ-balls in $Gr(20, 200)$ with $\epsilon \leq 1$.

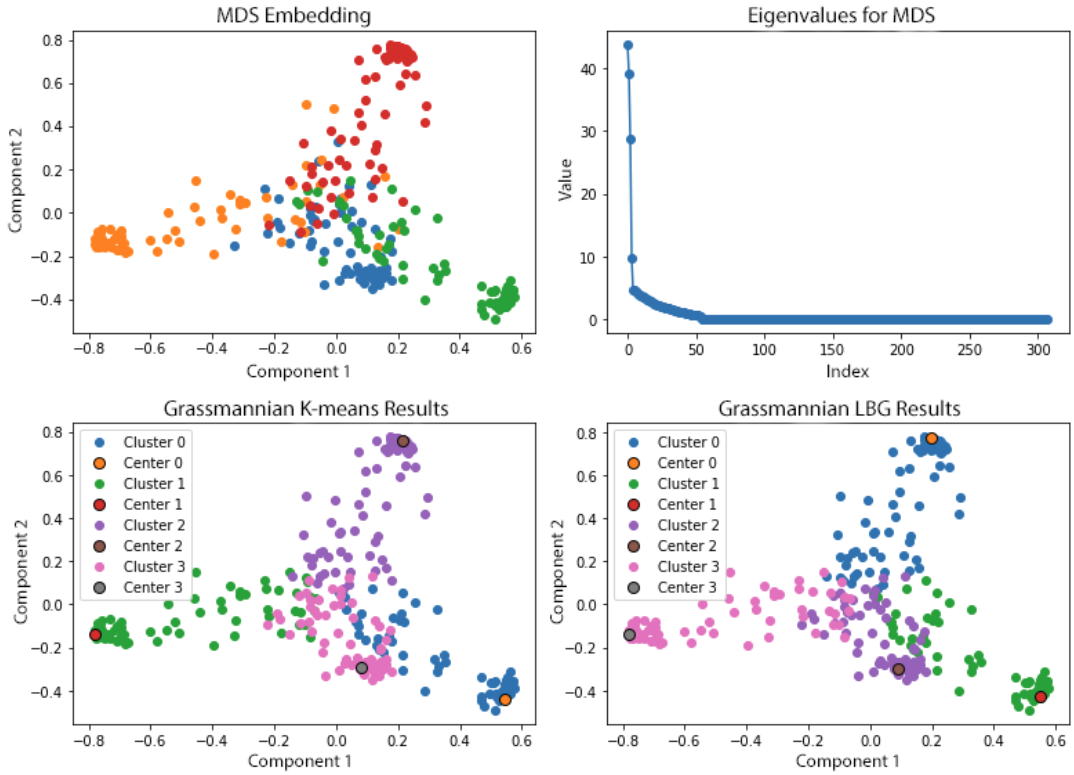

Figure 6.11: Chordal distance MDS embedding of four ϵ-balls in $Gr(2, 10)$ with $\epsilon \leq 1$.

few dimensions, though the distinction is not as extreme as in the $Gr(20, 200)$ plot. This is seen in the way each cluster is more spread throughout the embedding space. Again, Grassmannian K-means and Grassmannian LBG were evaluated on the data and successfully approximated each true center. However, repeated trials of this and the previous test on the higher dimensional manifold suggest that the Grassmannian K-means algorithm is much more susceptible to terminating in local minima than the Grassmannian LBG is, especially in higher dimensions.

6.2 MNIST Trials

The MNIST handwritten digit data set [68] is used to evaluate Grassmannian K-means and Grassmannian LBG and provide a basis for comparison with the Euclidean clustering methods described in Section 2.5. For every trial described, five test runs were performed, and all metrics are reported as average $\pm$ variance.

6.2.1 Three Class Trial

The first classification task performed was differentiating between three handwritten digits in $Gr(5, 784)$. The digits chosen were 5, 3, and 6, with 500 data points randomly selected for each digit. Subspaces of dimension five were constructed for each digit, representing a classification task where samples are received in groups of five and can be associated with each other. After the five dimensional subspace representations were constructed, each class contained 100 data points.

Table 6.1 contains the results from this experiment. Trials were run with both Grassmannian K-means and Grassmannian LBG using $k = 3$ and $k = 6$. Both algorithms perfectly classified the

Table 6.1: Results from the three-class MNIST experiment.

MNIST 3 Class Trials						
Manifold	k	**Metric**	**Algorithm**	**Avg. Purity**	**Avg. Accuracy**	**Avg. Distortion**
$Gr(5, 784)$	3	Chordal	K-means	99.68 ± 0.00	99.67 ± 0.00	874.23 ± 90.13
$Gr(5, 784)$	3	Chordal	LBG	87.11 ± 1.73	79.93 ± 3.36	882.81 ± 471.55
$Gr(5, 784)$	6	Chordal	K-means	100.00 ± 0.00	100.00 ± 0.00	856.58 ± 81.75
$Gr(5, 784)$	6	Chordal	LBG	100.00 ± 0.00	100.00 ± 0.00	834.84 ± 3.23

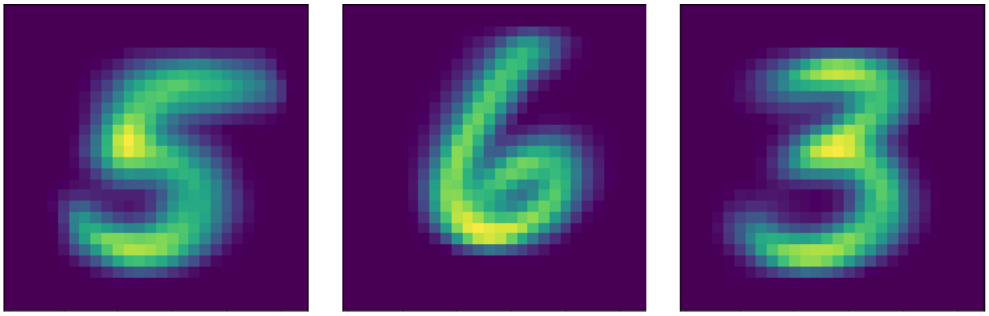

Figure 6.12: The first flag vector from each of the 3 LBG centers selected in the best 3 center experiment.

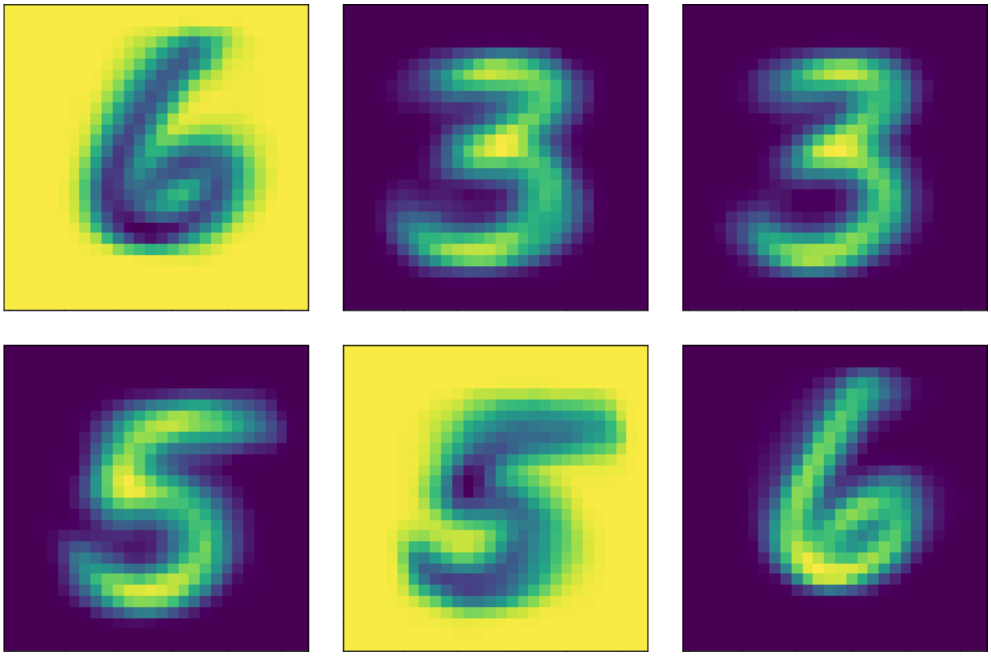

Figure 6.13: The first flag vector from each of the 6 LBG centers selected in the best 6 center experiment.

data using six centers. The first flag vector from each LBG center for a single three center run is visualized in Figure 6.12, and the flag vectors for a single six vector run are in Figure 6.13.

Interestingly, Grassmannian LBG performed much more poorly using $k = 3$ than Grassmannian K-means did. This is primarily because Grassmannian LBG frequently grouped digits 3 and 6 together, resulting in low accuracy. As suggested by the flag vectors in Figure 6.12, the best Grassmannian LBG trial did separate the three digits properly, but in other trials the algorithm terminated in a local minimum instead. This minimum occurs when a single center is positioned between the clusters for digits 3 and 6, with the remaining two centers assigned to digit 5. Figure 6.14 shows the first flag vector from one of these mixed centers, which, upon close inspection,

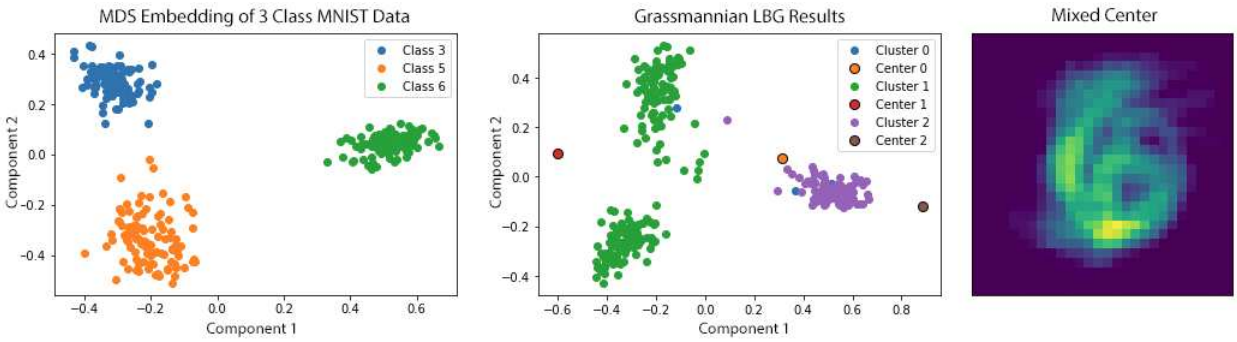

Figure 6.14: An MDS embedding with true labels for the three class problem, the embedding of centers and labels from a low accuracy LBG run, and the first flag vector from the red center.

appears to be a 6 overlayed with a noisy 3. The MDS embedding of the data and all centers from the same trial is also pictured in Figure 6.14.

Even though both algorithms perfectly separate the three classes with $k = 6$, the Grassmannian LBG algorithm has a lower average cluster distortion with less variance than that of Grassmannian K-means. This is likely because a local minimum is guaranteed when a center is the average of all points in its Voronoi cell, but this is not always the case with K-means (see Section 4.3 for more discussion). This could be addressed after the final epoch of K-means by averaging each cluster and setting each center equal to its cluster mean, but in this added complexity is not necessary for correct classifications since cluster membership would not be changed.

6.2.2 Ten Class Trial

The primary classification test used for comparing Grassmannian LBG and Grassmannian K-means with other clustering algorithms was the 10 class MNIST experiment. This experiment tests the ability of algorithms to distinguish between all digits from 0 to 9. All clustering is performed on $Gr(5, 784)$, and 500 data points were chosen from each class and used to make 100 subspaces of dimension 5 per digit for a total of 1000 points. This again represents a scenario where handwriting samples for a digit are received in groups of five, but the digit featured is unknown.

Table 6.2 shows results comparing performance of Grassmannian LBG and Grassmannian K-means using different distance metrics on $Gr(5, 784)$. Each algorithm and metric pair were tested

Table 6.2: Results from MNIST 10 class comparison of algorithm performance using different distance metrics on $Gr(5, 200)$.

MNIST 10 Class Trials						
Manifold	k	**Metric**	**Algorithm**	**Avg. Purity**	**Avg. Accuracy**	**Avg. Distortion**
$Gr(5, 784)$	10	Chordal	K-means	89.57 ± 0.07	80.60 ± 0.43	2923.33 ± 342.68
$Gr(5, 784)$	10	Chordal	LBG	91.64 ± 0.08	85.38 ± 0.28	2865.84 ± 1432.53
$Gr(5, 784)$	20	Chordal	K-means	96.89 ± 0.06	95.58 ± 0.19	2799.84 ± 342.46
$Gr(5, 784)$	20	Chordal	LBG	98.60 ± 0.03	99.60 ± 0.00	2721.22 ± 170.98
$Gr(5, 784)$	10	Geo.	K-means	84.14 ± 0.11	75.78 ± 0.30	4863.10 ± 1541.13
$Gr(5, 784)$	10	Geo.	LBG	90.94 ± 0.06	83.14 ± 0.26	4766.52 ± 2447.49
$Gr(5, 784)$	20	Geo.	K-means	95.45 ± 0.02	94.58 ± 0.08	4527.74 ± 2484.65
$Gr(5, 784)$	20	Geo.	LBG	95.92 ± 0.02	96.86 ± 0.00	4466.68 ± 920.20
$Gr(5, 784)$	10	Pseudo.	K-means	93.53 ± 0.31	86.86 ± 1.21	282.79 ± 32.41
$Gr(5, 784)$	10	Pseudo.	LBG	91.61 ± 0.08	83.94 ± 0.30	282.79 ± 32.41
$Gr(5, 784)$	20	Pseudo.	K-means	99.92 ± 0.00	99.92 ± 0.00	265.97 ± 1.07
$Gr(5, 784)$	20	Pseudo.	LBG	99.54 ± 0.00	99.84 ± 0.00	257.35 ± 2.68

for both $k = 10$ and $k = 20$. All methods had average cluster purity greater than 84%, and only geodesic Grassmannian K-means had accuracy fall under 80%.

The most accurate distance measure for clustering all ten digits was the first principal angle pseudometric, with both accuracy and purity averaging over 99% with very little variance. Figure 6.15 contains cluster assignment results from all five Grassmannian K-means trials, and Figure 6.16 contains results from the five Grassmannian LBG trials. In particular, the results for the second Grassmannian K-means trial show that only two points out of 1000 were incorrectly classified. The unsupervised classification accuracy achieved by both algorithms is higher than the other previously discussed unsupervised results on this data set, clearly demonstrating the benefit of using subspaces for sample groupings.

6.3 Indian Pines Trials

The AVIRIS Indian Pines data set [69] was used to further test both algorithms on problems with far less data available. Recall that this data set consists of 16 classes varying in size from 20 points to 2455 points. The bands corresponding to the wavelengths of water absorption were removed for all trials performed on this data.

Figure 6.15: Grassmannian K-means center assignments from the five 10 class MNIST trials using the first principal angle psuedometric.

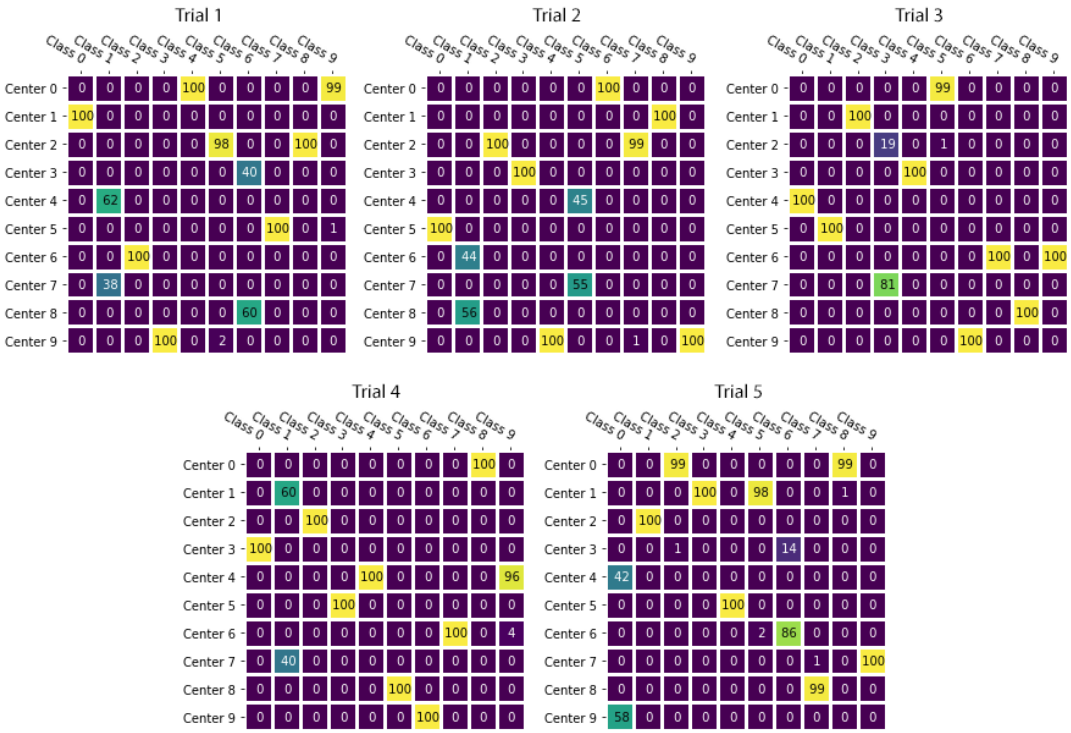

Figure 6.16: Grassmannian K-means center assignments from the five 10 class MNIST trials using the first principal angle psuedometric.

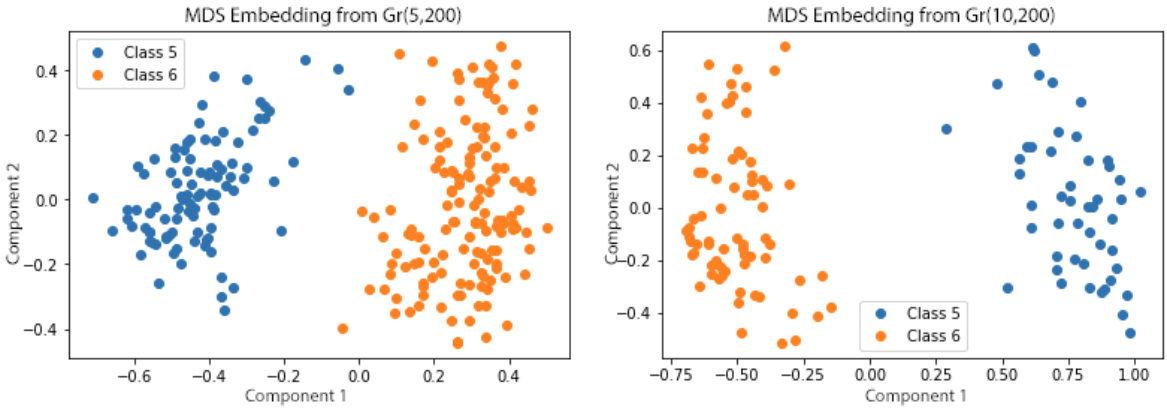

Figure 6.17: MDS embedding of the *pasture* (class 5) and *trees* (class 6) classes.

6.3.1 Pasture vs. Trees

Both Grassmannian LBG and Grasmannian K-means were tested on the classes *grass-pasture* (class 5) and *grass-trees* (class 6). For simplicity, these two classes are referred to as *pasture* and *trees*, respectively. These classes were chosen due to their similarity and relatively even sample sizes.

Algorithms were tested using points on both $Gr(5, 200)$ and $Gr(10, 200)$. Since the class *pasture* contains 483 total data points, moving the data to $Gr(5, 200)$ according to the method described in Section 3.1 reduced the class size to 96, and moving data to $Gr(10, 200)$ reduced the number of points to 48. The class *trees* contains 730 data points, which reduces to 146 on $Gr(5, 200)$ and 73 on $Gr(10, 200)$. For MDS embeddings of the two classes from both manifolds, see Figure 6.17. The chordal distance metric was used for all algorithms and embeddings. Trials on $Gr(5, 200)$ were run using $k = 2$, $k = 4$, and $k = 6$. Trials on $Gr(10, 200)$ were run using $k = 2$ and $k = 4$. All experiments were repeated five times, and all results are reported as averages of the five runs.

Table 6.3 contains the results for all trials. Both algorithms performed very well in all experiments, which is unsurprising given that the MDS embedding of the data shows separability for both manifolds. In particular, both algorithms performed perfectly in all trials on $Gr(10, 200)$.

Table 6.3: Results from AVIRIS Indian Pines *pasture* vs. *trees* experiments.

Indian Pines Pasture vs. Trees						
Manifold	k	**Metric**	**Algorithm**	**Avg. Purity**	**Avg. Accuracy**	**Avg. Distortion**
$Gr(5, 200)$	2	Chordal	K-means	98.87 ± 0.00	98.76 ± 0.00	491.91 ± 6.05
$Gr(5, 200)$	2	Chordal	LBG	98.61 ± 0.00	98.53 ± 0.00	478.57 ± 0.00
$Gr(5, 200)$	4	Chordal	K-means	98.92 ± 0.01	99.01 ± 0.00	464.59 ± 8.39
$Gr(5, 200)$	4	Chordal	LBG	97.88 ± 0.06	98.35 ± 0.01	452.00 ± 19.00
$Gr(5, 200)$	6	Chordal	K-means	99.15 ± 0.00	99.17 ± 0.00	451.90 ± 36.94
$Gr(5, 200)$	6	Chordal	LBG	98.02 ± 0.00	98.48 ± 0.00	434.01 ± 25.54
$Gr(10, 200)$	2	Chordal	K-means	100.00 ± 0.00	100.00 ± 0.00	448.95 ± 1.21
$Gr(10, 200)$	2	Chordal	LBG	100.00 ± 0.00	100.00 ± 0.00	435.98 ± 0.00
$Gr(10, 200)$	4	Chordal	K-means	100.00 ± 0.00	100.00 ± 0.00	439.01 ± 0.84
$Gr(10, 200)$	4	Chordal	LBG	100.00 ± 0.00	100.00 ± 0.00	420.37 ± 4.87

6.3.2 Corn vs. Alfalfa

The second experiment on the Indian Pines data set utilized the classes *alfalfa* (class 1) and *corn* (class 4). These classes were selected for their unbalanced sizes and overall fewer data points in order to test the robustness of Grassmannian K-means and Grassmannian LBG on a small data set.

Both algorithms were tested on $Gr(5, 200)$, $Gr(10, 200)$, and $Gr(15, 200)$. The *alfalfa* class contains only 46 points, which subspace construction reduces to 9 on $Gr(5, 200)$, 4 on $Gr(10, 200)$, and 3 on $Gr(15, 200)$. This class specifically was chosen to evaluate algorithm performance on extremely small classes. The *corn* class contains 237 data points, reducing to 47 on $Gr(5, 200)$, 22 on $Gr(10, 200)$, and 15 on $Gr(5, 200)$, making it approximately five times the size of *alfalfa* on all manifolds. Figure 6.18 contains MDS embeddings of the two classes from each manifold.

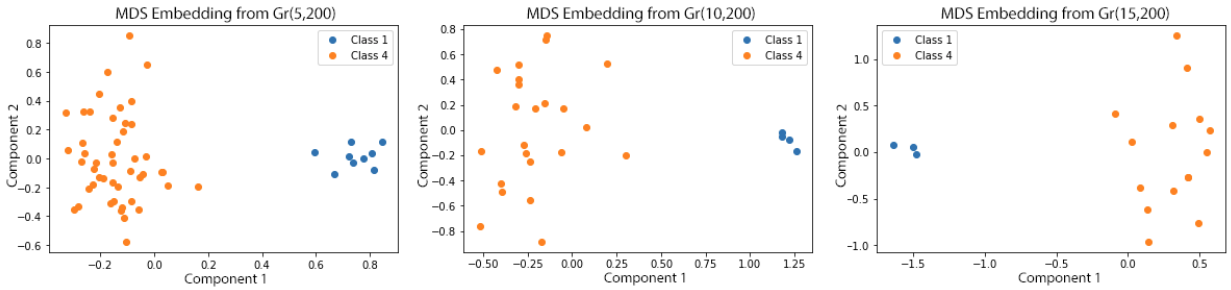

Figure 6.18: MDS embeddings of *alfalfa* (class 1) and *corn* (class 4) using the chordal metric.

Table 6.4: Results from AVIRIS Indian Pines *corn* vs. *alfalfa* experiments.

Indian Pines Corn vs. Alfalfa						
Manifold	k	**Metric**	**Algorithm**	**Avg. Purity**	**Avg. Accuracy**	**Avg. Distortion**
$Gr(5, 200)$	2	Chordal	K-means	100.00 ± 0.00	100.00 ± 0.00	114.54 ± 0.00
$Gr(5, 200)$	2	Chordal	LBG	93.78 ± 0.73	93.57 ± 0.77	108.60 ± 2.30
$Gr(5, 200)$	4	Chordal	K-means	100.00 ± 0.00	100.00 ± 0.00	102.66 ± 2.14
$Gr(5, 200)$	4	Chordal	LBG	94.59 ± 0.55	93.57 ± 0.77	101.16 ± 6.82
$Gr(5, 200)$	6	Chordal	K-means	100.00 ± 0.00	100.00 ± 0.00	96.38 ± 1.07
$Gr(5, 200)$	6	Chordal	LBG	100.00 ± 0.00	100.00 ± 0.00	92.08 ± 2.36
$Gr(10, 200)$	2	Chordal	K-means	100.00 ± 0.00	100.00 ± 0.00	98.69 ± 0.00
$Gr(10, 200)$	2	Chordal	LBG	90.00 ± 1.18	92.59 ± 0.55	97.01 ± 11.86
$Gr(10, 200)$	4	Chordal	K-means	100.00 ± 0.00	100.00 ± 0.00	88.69 ± 0.52
$Gr(10, 200)$	4	Chordal	LBG	91.15 ± 0.56	90.39 ± 0.99	87.03 ± 1.62
$Gr(10, 200)$	6	Chordal	K-means	100.00 ± 0.00	100.00 ± 0.00	80.36 ± 0.57
$Gr(10, 200)$	6	Chordal	LBG	94.22 ± 0.34	92.43 ± 0.66	77.01 ± 1.41
$Gr(15, 200)$	2	Chordal	K-means	100.00 ± 0.00	100.00 ± 0.00	91.75 ± 0.00
$Gr(15, 200)$	2	Chordal	LBG	87.83 ± 1.46	87.78 ± 2.69	89.52 ± 4.18
$Gr(15, 200)$	4	Chordal	K-means	100.00 ± 0.00	100.00 ± 0.00	78.75 ± 0.11
$Gr(15, 200)$	4	Chordal	LBG	95.50 ± 1.01	96.67 ± 0.56	75.61 ± 3.33

Results for these experiments are in Table 6.4. Surprisingly, Grassmannian K-means perfectly separated the two clusters in every single trial. Grassmannian LBG, however, was less successful in classification, even though in most cases the average distortion at termination was lower than for Grassmannian K-means. Again, this is likely due to the fact that the final Grassmannian K-means centers are not the true averages of their clusters. It may also be a case where the optimal partition for classification purposes is not necessarily the global minimum of the cost function. Regardless, Grassmanian LBG was very sensitive to starting conditions with these small, unbalanced clusters.

6.3.3 Soybeans

The final Indian Pines experiment was performed on the three soybean classes: *soybean-notill* (class 10), *soybean-mintill* (class 11), and *soybean-clean* (class 12), which will be referred to by their numbers for simplicity. These three classes are inseparable in Euclidean space, but become separable when converted to points on the Grassmannian, specifically by using the smallest princi-

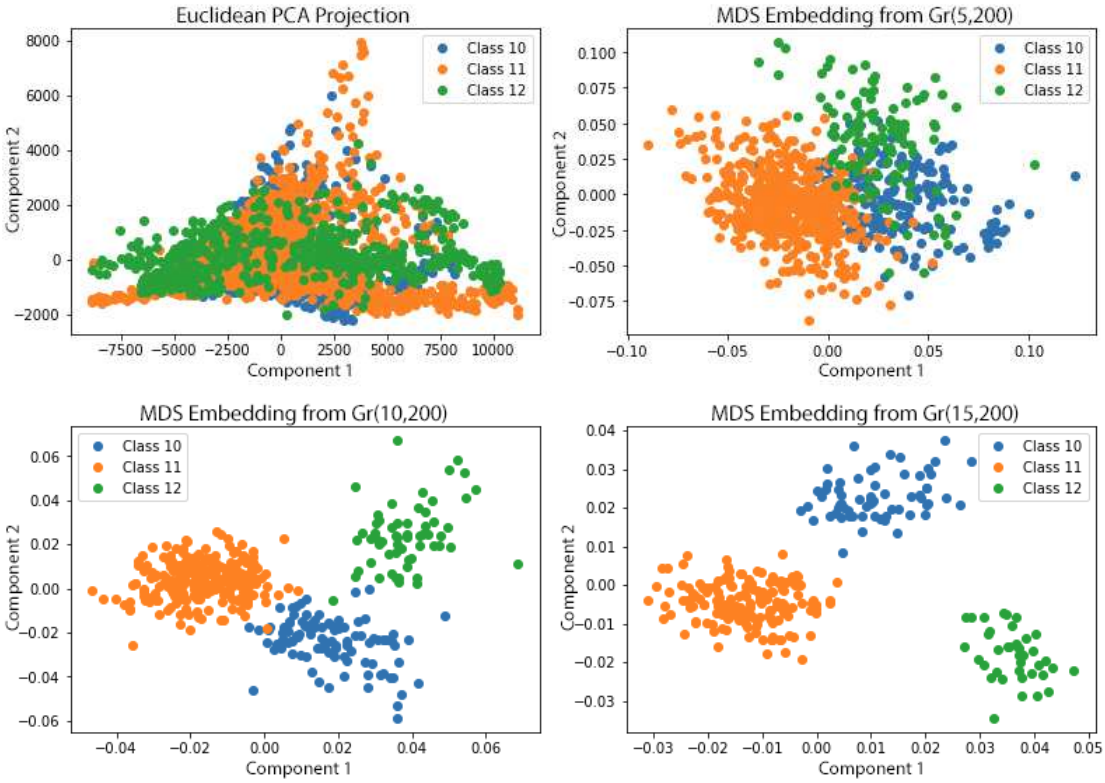

Figure 6.19: MDS embedding of the three soybean classes using Euclidean distance and smallest angle pseudometric on $Gr(5, 200)$, $Gr(10, 200)$, and $Gr(15, 200)$.

pal angle pseudometric. Trials performed on the three soybean classes in this data set particularly illustrate the benefits of Grassmannian clustering methods.

As in the *corn* vs. *alfalfa* trials, the classes in this experiment are not balanced. Class 10 contains 972 data points, which reduces to 194 on $Gr(5, 200)$, 96 on $Gr(10, 200)$, and 64 on $Gr(15, 200)$. Class 11 contains 2455 data points, which then reduces to 491 on $Gr(5, 200)$, 245 on $Gr(10, 200)$, and 163 on $Gr(15, 200)$. Class 12 contains 593 data points and reduces to 118 on $Gr(5, 200)$, 59 on $Gr(10, 200)$, and 39 on $Gr(15, 200)$. Due to the reduction in the number of points available for clustering, no subspaces of dimension higher than 15 were used. Figure 6.19 depicts several MDS embeddings of these classes. Clearly, the data is not at all separable in Euclidean space. However, when it is transferred to a Grassmannian and pairwise distances are calculated using the smallest principal angle pseudometric, separability increases with the dimension of the manifold.

Table 6.5: Comparison of three approaches to clustering.

Soybeans Method Comparison						
Method	**Manifold**	k	**Metric**	**Algorithm**	**Avg. Purity**	**Avg. Accuracy**
Euclidean	$\mathbb{R}^{200}$	3	Euclidean	K-means	56.95 ± 0.00	61.07 ± 0.00
Euclidean	$\mathbb{R}^{200}$	3	Euclidean	LBG	56.73 ± 0.00	61.07 ± 0.00
Embedded	$Gr(5, 200)$	3	Pseudometric	K-means	84.09 ± 0.05	73.70 ± 0.75
Embedded	$Gr(5, 200)$	3	Pseudometric	LBG	85.34 ± 0.67	86.90 ± 0.57
Grassmannian	$Gr(5, 200)$	3	Pseudometric	K-means	85.26 ± 0.10	85.32 ± 0.27
Grassmannian	$Gr(5, 200)$	3	Pseudometric	LBG	73.49 ± 0.18	78.74 ± 0.07

The first set of experiments depicted in Table 6.5 takes a different approach than previous trials. Here, two possible approaches to utilizing the Grassmannian in clustering are compared. In the first approach, points are transferred to $Gr(5, 200)$, the pseudometric is used to construct a pairwise distance matrix, and then MDS is performed on the distance matrix to embed the clusters back into Euclidean space, where the clustering task is performed. The second method is transferring data to $Gr(5, 200)$ and applying either Grassmannian LBG or Grassmannian K-means on the manifold itself, using the pseudometric to compute distances. This provides a basis for comparison of the developed algorithms with another strategy that utilizes the Grassmannian, but does not require computations on the manifold itself. Trials using the standard Euclidean K-means and LBG are also included as a baseline comparison for algorithm performance. All experiments are done using $k = 3$ clusters and are repeated 10 times.

Table 6.5 contains the averages for classification accuracy and cluster purity over all trials. Not surprisingly, the Euclidean algorithms performed poorly on the classification task. Clustering on embedded data performed similarly to clustering directly on the manifold itself. In particular, the LBG algorithm seems to benefit slightly from clustering on embedded data, whereas the K-means algorithm performs slightly better on the Grassmannian. The classification accuracy for both algorithms had higher variance on the embedded data than on the Grassmannian data, suggesting that clustering directly on the Grassmannian benefits more from subspace robustness. Overall, it appears that the best method of clustering may be dependant both on the chosen algorithm and the data itself.

Table 6.6: Comparison of algorithm performance on different manifolds using the smallest principal angle pseudometric.

Indian Pines Soybeans Trials Table 1						
Manifold	k	**Metric**	**Algorithm**	**Avg. Purity**	**Avg. Accuracy**	**Avg. Distortion**
$Gr(5, 200)$	3	Pseudo	K-means	85.26 ± 0.10	85.32 ± 0.27	5.38 ± 0.00
$Gr(5, 200)$	3	Pseudo	LBG	73.49 ± 0.18	78.74 ± 0.07	5.21 ± 0.00
$Gr(5, 200)$	6	Pseudo	K-means	82.34 ± 0.13	83.09 ± 0.06	5.03 ± 0.00
$Gr(5, 200)$	6	Pseudo	LBG	83.31 ± 0.02	84.11 ± 0.01	4.77 ± 0.00
$Gr(5, 200)$	9	Pseudo	K-means	82.53 ± 0.17	84.63 ± 0.06	4.90 ± 0.00
$Gr(5, 200)$	9	Pseudo	LBG	87.89 ± 0.01	88.04 ± 0.01	4.58 ± 0.00
$Gr(10, 200)$	3	Pseudo	K-means	92.67 ± 0.14	88.48 ± 0.25	1.23 ± 0.00
$Gr(10, 200)$	3	Pseudo	LBG	95.92 ± 0.04	95.46 ± 0.43	1.11 ± 0.00
$Gr(10, 200)$	6	Pseudo	K-means	95.82 ± 0.13	93.12 ± 0.16	1.17 ± 0.00
$Gr(10, 200)$	6	Pseudo	LBG	97.62 ± 0.00	96.82 ± 0.00	1.08 ± 0.00
$Gr(10, 200)$	9	Pseudo	K-means	97.89 ± 0.04	94.16 ± 0.05	1.15 ± 0.00
$Gr(10, 200)$	9	Pseudo	LBG	98.13 ± 0.00	96.66 ± 0.01	1.07 ± 0.00
$Gr(15, 200)$	3	Pseudo	K-means	98.88 ± 0.59	90.00 ± 0.43	0.51 ± 0.00
$Gr(15, 200)$	3	Pseudo	LBG	97.36 ± 0.07	94.21 ± 0.54	0.46 ± 0.00
$Gr(15, 200)$	6	Pseudo	K-means	98.68 ± 0.02	94.66 ± 0.40	0.50 ± 0.00
$Gr(15, 200)$	6	Pseudo	LBG	98.41 ± 0.03	99.10 ± 0.00	0.45 ± 0.00

More experiments were run on the soybean data to compare the performance of Grassmannian LBG and Grassmannian K-means using different values of k, different manifolds, and different distance measures. For these experiments, ten trials were run and reported results are averaged across these trials.

Each of the following tests were performed with chordal distance, geodesic distance, and smallest principal angle distance. Soybean data on $Gr(5, 200)$ was clustered with both algorithms using $k = 3$, $k = 6$, and $k = 9$. Data on $Gr(10, 200)$ was also clustered using $k = 3$, $k = 6$, and $k = 9$. Finally, data on $Gr(15, 200)$ was clustered using $k = 3$ and $k = 6$. Table 6.6 contains all results using the pseudometric, Table 6.7 contains results using geodesic distance, and Table 6.8 contains results obtained using chordal distance.

Examination of the purity and accuracy averages from all tests reveals that the smallest principal angle pseudometric was the best choice of distance for classification by a wide margin, with only Grassmannian LBG on $Gr(5, 200)$ with $k = 3$ yielding results below 80%. For all met-

Table 6.7: Comparison of algorithm performance on different manifolds using the geodesic metric.

Indian Pines Soybeans Trials Table 2						
Manifold	k	**Metric**	**Algorithm**	**Avg. Purity**	**Avg. Accuracy**	**Avg. Distortion**
$Gr(5, 200)$	3	Geo.	K-means	62.74 ± 0.02	61.80 ± 0.01	2288.83 ± 269.03
$Gr(5, 200)$	3	Geo.	LBG	62.19 ± 0.01	62.22 ± 0.06	2262.79 ± 55.98
$Gr(5, 200)$	6	Geo.	K-means	65.91 ± 0.01	66.23 ± 0.04	2122.27 ± 689.13
$Gr(5, 200)$	6	Geo.	LBG	69.32 ± 0.03	67.20 ± 0.01	2093.66 ± 216.15
$Gr(5, 200)$	9	Geo.	K-means	69.44 ± 0.16	68.44 ± 0.01	2045.19 ± 1391.15
$Gr(5, 200)$	9	Geo.	LBG	71.35 ± 0.04	71.51 ± 0.04	2002.03 ± 80.90
$Gr(10, 200)$	3	Geo.	K-means	76.63 ± 0.08	68.68 ± 0.26	2363.03 ± 2381.32
$Gr(10, 200)$	3	Geo.	LBG	68.96 ± 0.03	70.17 ± 0.08	2324.61 ± 82.66
$Gr(10, 200)$	6	Geo.	K-means	80.70 ± 0.23	70.67 ± 0.18	2270.32 ± 376.49
$Gr(10, 200)$	6	Geo.	LBG	76.96 ± 0.10	77.16 ± 0.10	2197.80 ± 50.84
$Gr(10, 200)$	9	Geo.	K-means	80.96 ± 0.31	72.87 ± 0.15	2202.25 ± 593.92
$Gr(10, 200)$	9	Geo.	LBG	78.70 ± 0.07	78.20 ± 0.07	2125.10 ± 83.35
$Gr(15, 200)$	3	Geo.	K-means	81.14 ± 0.28	69.25 ± 0.82	2291.72 ± 2152.87
$Gr(15, 200)$	3	Geo.	LBG	72.45 ± 0.12	71.96 ± 0.22	2226.14 ± 31.57
$Gr(15, 200)$	6	Geo.	K-means	87.35 ± 0.14	72.18 ± 0.54	2190.26 ± 2192.28
$Gr(15, 200)$	6	Geo.	LBG	83.82 ± 0.19	84.21 ± 0.04	2109.24 ± 65.82

Table 6.8: Comparison of algorithm performance on different manifolds using the chordal metric.

Indian Pines Soybeans Trials Table 3						
Manifold	k	**Metric**	**Algorithm**	**Avg. Purity**	**Avg. Accuracy**	**Avg. Distortion**
$Gr(5, 200)$	3	Chordal	K-means	66.75 ± 0.09	65.68 ± 0.05	1450.04 ± 159.12
$Gr(5, 200)$	3	Chordal	LBG	65.02 ± 0.07	64.39 ± 0.07	1429.36 ± 12.64
$Gr(5, 200)$	6	Chordal	K-means	73.95 ± 0.35	67.40 ± 0.05	1412.00 ± 116.26
$Gr(5, 200)$	6	Chordal	LBG	69.19 ± 0.03	68.77 ± 0.02	1358.46 ± 12.49
$Gr(5, 200)$	9	Chordal	K-means	74.61 ± 0.16	68.84 ± 0.01	1382.48 ± 112.23
$Gr(5, 200)$	9	Chordal	LBG	73.75 ± 0.06	70.29 ± 0.06	1326.17 ± 48.03
$Gr(10, 200)$	3	Chordal	K-means	81.18 ± 0.56	67.83 ± 0.23	1496.59 ± 607.39
$Gr(10, 200)$	3	Chordal	LBG	73.59 ± 0.15	74.21 ± 0.14	1413.21 ± 52.68
$Gr(10, 200)$	6	Chordal	K-means	89.95 ± 0.42	75.76 ± 0.30	1433.90 ± 543.02
$Gr(10, 200)$	6	Chordal	LBG	84.35 ± 0.04	83.54 ± 0.04	1360.31 ± 38.85
$Gr(10, 200)$	9	Chordal	K-means	91.63 ± 0.29	73.42 ± 0.16	1425.33 ± 605.53
$Gr(10, 200)$	9	Chordal	LBG	85.15 ± 0.22	84.14 ± 0.06	1334.35 ± 14.09
$Gr(15, 200)$	3	Chordal	K-means	86.12 ± 0.05	73.99 ± 1.03	1417.47 ± 572.63
$Gr(15, 200)$	3	Chordal	LBG	72.22 ± 0.78	74.96 ± 0.83	1344.13 ± 48.31
$Gr(15, 200)$	6	Chordal	K-means	93.51 ± 0.02	70.98 ± 0.86	1387.86 ± 682.66
$Gr(15, 200)$	6	Chordal	LBG	82.25 ± 0.16	83.01 ± 0.20	1297.41 ± 50.17

rics, classification accuracy and purity increased with both the dimension of the manifold and the number of clusters used.

Chapter 7

Conclusion

The primary contribution of this thesis is the adaptation of the LBG and K-means algorithms to the Grassmann manifold. We test both Grassmannian K-means and Grassmannian LBG rigorously on well-known data sets, demonstrating robustness and providing a baseline for comparison with other work on the Grassmannian. The theory behind the adaptations is described in depth to make the work as self contained as possible and to create a detailed, coherent reference for Grassmannian clustering in general.

In Chapter 2, we provided an overview of clustering algorithms and their applications. We described in further depth the partitional clustering algorithms of interest and their variations. We also summarized previous work on Grassmannian clustering, including the connection between the flag mean and the projection Frobenius norm eigenvalue problem found in subspace averaging.

Chapter 3 provided an in-depth look at the Grassmann manifold and its geometry. We describe the relationship between matrix manifolds in real space and the real Grassmann manifold. We also included full derivations for computing geodesic curves and averaging subspaces, as well as a discussion of metrics and pseudometrics on the Grassmannian.

Chapter 4 provided details on the Euclidean versions of the K-means and LBG algorithms, which were then adapted to the Grassmann manifold in Chapter 5. Each chapter contains thorough discussions of computational complexity for their respective algorithms.

Chapter 6 contained experimental results on a toy data set and two benchmark data sets. The toy data set was used to visualize and explore the geometry of high-dimensional Grassmannians along with verifying the functionality of both Grassmannian algorithms. The benchmark data sets provided an avenue of comparison with previously reported clustering results on both the Grassmannian and in Euclidean space.

The results in this thesis lend themselves to several avenues of future work. Due to the fact that the Grassmannian LBG updates are performed using a flag mean calculation, this algorithm

could be adapted to function on flag manifolds. This would allow for clustering on subspaces of different dimensions, and would allow for data to be converted into subspaces without omitting any points. The Grassmannian algorithms could potentially be adapted to operate on collections of affine subspaces as well. Additionally, since we now possess both a batch and an online clustering algorithm on the Grassmannian, these two approaches could be combined to yield potentially better clustering results. Both algorithms show some sensitivity to the initial choice of centers, so further investigation into selecting optimal starting conditions on the manifold is needed. Methods for automatic selection of the parameter k could easily be translated to the Grassmannian as well. Besides further development of algorithms, more exploration is needed into the geometry of embeddings of high-dimensional Grassmannians. In particular, it appears the locality of points on the manifold drastically affects the appearance of the embedding, even when the embedding itself is of low dimension. Overall, we look forward to expanding on the results from this thesis.

Bibliography

[1] Jen-Mei Chang, Michael Kirby, Holger Kley, Chris Peterson, Bruce Draper, and J. Ross Beveridge. Recognition of digital images of the human face at ultra low resolution via illumination spaces. In *Asian Conference on Computer Vision*, pages 733–743. Springer, 2007.

[2] J. Ross Beveridge, Bruce Draper, Jen-Mei Chang, Michael Kirby, Holger Kley, and Chris Peterson. Principal angles separate subject illumination spaces in YDB and CMU-PIE. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):351–363, 2009.

[3] Sofya Chepushtanova and Michael Kirby. Classification of hyperspectral imagery on embedded grassmannians. *arXiv preprint arXiv:1502.00946*, 2015.

[4] Anil K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[5] Ronald L. Breiger, Scott A. Boorman, and Phipps Arabie. An algorithm for clustering relational data with applications to social network analysis and comparison with multidimensional scaling. *Journal of Mathematical Psychology*, 12(3):328–383, 1975.

[6] Paul Scheunders. A comparison of clustering algorithms applied to color image quantization. *Pattern Recognition Letters*, 18(11-13):1379–1384, 1997.

[7] Yoseph Linde, Andres Buzo, and Robert Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95, 1980.

[8] Alexander Sturn, John Quackenbush, and Zlatko Trajanoski. Genesis: Cluster analysis of microarray data. *Bioinformatics*, 18(1):207–208, 2002.

[9] Andrés Buzo, A. Gray, R. Gray, and John Markel. Speech coding based upon vector quantization. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(5):562–574, 1980.

[10] John Makhoul, Salim Roucos, and Herbert Gish. Vector quantization in speech coding. *Proceedings of the IEEE*, 73(11):1551–1588, 1985.

[11] Robert M. Gray and David L. Neuhoff. Quantization. *IEEE transactions on Information Theory*, 44(6):2325–2383, 1998.

[12] Adil Fahad, Najlaa Alshatri, Zahir Tari, Abdullah Alamri, Ibrahim Khalil, Albert Y. Zomaya, Sebti Foufou, and Abdelaziz Bouras. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE Transactions on Emerging Topics in Computing*, 2(3):267–279, 2014.

[13] Alan P. Reynolds, Graeme Richards, Beatriz de la Iglesia, and Victor J. Rayward-Smith. Clustering rules: A comparison of partitioning and hierarchical clustering algorithms. *Journal of Mathematical Modelling and Algorithms*, 5(4):475–504, 2006.

[14] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. Statistical Laboratory of the University of California, Berkeley, 1967.

[15] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, volume 96, pages 226–231. AIII, 1996.

[16] Chibiao Chen, Eric Durand, Florence Forbes, and Olivier François. Bayesian clustering algorithms ascertaining spatial population structure: A new computer program and a comparison study. *Molecular Ecology Notes*, 7(5):747–756, 2007.

[17] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[18] Wei Li and Andrew McCallum. Pachinko allocation: DAG-structured mixture models of topic correlations. In *Proceedings of the 23rd international Conference on Machine Learning*, pages 577–584. ACM, 2006.

[19] René Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68, 2011.

[20] Ehsan Elhamifar and Rene Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2765–2781, 2013.

[21] Charu C. Aggarwal, Joel L. Wolf, Philip S. Yu, Cecilia Procopiuc, and Jong Soo Park. Fast algorithms for projected clustering. In *ACM SIGMOD Record*, volume 28, pages 61–72. ACM, 1999.

[22] Liping Jing, Michael K. Ng, and Joshua Zhexue Huang. An entropy weighting K-means algorithm for subspace clustering of high-dimensional sparse data. *IEEE Transactions on Knowledge & Data Engineering*, (8):1026–1041, 2007.

[23] Philip A. Chou, Tom Lookabaugh, and Robert M. Gray. Entropy-constrained vector quantization. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(1):31–42, 1989.

[24] Haifeng Li, Keshu Zhang, and Tao Jiang. Minimum entropy clustering and applications to gene expression analysis. In *Proceedings. 2004 IEEE Computational Systems Bioinformatics Conference*, pages 142–151. IEEE, 2004.

[25] Stephen J. Roberts, Richard Everson, and Iead Rezek. Minimum entropy data partitioning. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*. IET, 1999.

[26] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[27] Amin Karami and Manel Guerrero-Zapata. A fuzzy anomaly detection system based on hybrid PSO-Kmeans algorithm in content-centric networks. *Neurocomputing*, 149:1253–1269, 2015.

[28] Yi-Tung Kao, Erwie Zahara, and I-Wei Kao. A hybridized approach to data clustering. *Expert Systems with Applications*, 34(3):1754–1762, 2008.

[29] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained K-means clustering with background knowledge. In *ICML '01 Proceedings of the Eighteenth International Conference on Machine Learning*, volume 1, pages 577–584. Morgan Kaufmann Publishers Inc., 2001.

[30] Paul S. Bradley, Usama M. Fayyad, and Cory Reina. Scaling clustering algorithms to large databases. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 9–15. AIII, 1998.

[31] Rong Zhang and Alexander I. Rudnicky. A large scale clustering scheme for kernel K-means. In *Object Recognition Supported by User Interaction for Service Robots*, volume 4, pages 289–292. IEEE, 2002.

[32] Zhexue Huang. Extensions to the K-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, 1998.

[33] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge, Discovery, and Data Mining*, 2000.

[34] Manish Verma, Mauly Srivastava, Neha Chack, Atul Kumar Diswar, and Nidhi Gupta. A comparative study of various clustering algorithms in data mining. *International Journal of Engineering Research and Applications*, 2(3):1379–1384, 2012.

[35] Osama Abu Abbas. Comparisons between data clustering algorithms. *International Arab Journal of Information Technology (IAJIT)*, 5(3), 2008.

[36] Allen Gersho and Robert M. Gray. *Vector Quantization and Signal Compression*, volume 159. Springer Science & Business Media, 2012.

[37] Nasser M. Nasrabadi and Robert A. King. Image coding using vector quantization: A review. *IEEE Transactions on Communications*, 36(8):957–971, 1988.

[38] Sudipto Guha, Nina Mishra, Rajeev Motwani, and Liadan o'Callaghan. Clustering data streams. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 359–366. IEEE, 2000.

[39] Liadan O'callaghan, Nina Mishra, Adam Meyerson, Sudipto Guha, and Rajeev Motwani. Streaming-data algorithms for high-quality clustering. In *Proceedings of the 18th International Conference on Data Engineering*, pages 685–694. IEEE, 2002.

[40] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th International Conference on Very Large Data Bases*, pages 81–92. VLDB Endowment, Morgan Kaufmann Publishers Inc., 2003.

[41] Stuart Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[42] Joseph C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

[43] James C. Bezdek. *Pattern Recognition With Fuzzy Objective Function Algorithms*. Springer Science & Business Media, 2013.

[44] James C. Bezdek, Robert Ehrlich, and William Full. FCM: The fuzzy C-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.

[45] K. Krishna and Narasimha M. Murty. Genetic K-means algorithm. *IEEE Transactions on Systems, Man, And Cybernetics-Part B: Cybernetics*, 29(3):433–439, 1999.

[46] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[47] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Kernel K-means: Spectral clustering and normalized cuts. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 551–556. ACM, 2004.

[48] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*, volume 344. John Wiley & Sons, 2009.

[49] Geoffrey H. Ball and David J. Hall. ISODATA, a novel method of data analysis and pattern classification. Technical report, Stanford Research Institute, 1965.

[50] Aristidis Likas, Nikos Vlassis, and Jakob J. Verbeek. The global K-means clustering algorithm. *Pattern Recognition*, 36(2):451–461, 2003.

[51] Greg Hamerly and Charles Elkan. Learning the k in K-means. In *Advances in Neural Information Processing Systems*, volume 17, pages 281–288. NIPS, 2004.

[52] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[53] José M. Pena, Jose Antonio Lozano, and Pedro Larranaga. An empirical comparison of four initialization methods for the K-means algorithm. *Pattern Recognition Letters*, 20(10):1027–1040, 1999.

[54] Dan Pelleg and Andrew W. Moore. X-means: Extending K-means with efficient estimation of the number of clusters. In *Proceedings of the 17th International Conf. on Machine Learning*, volume 1, pages 727–734. Morgan Kaufmann, 2000.

[55] John A. Hartigan and Manchek A. Wong. Algorithm AS 136: A K-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[56] Shehroz S. Khan and Amir Ahmad. Cluster center initialization algorithm for K-means clustering. *Pattern Recognition Letters*, 25(11):1293–1302, 2004.

[57] G. Phanendra Babu and M. Narasimha Murty. A near-optimal initial seed value selection in K-means means algorithm using a genetic algorithm. *Pattern Recognition Letters*, 14(10):763–769, 1993.

[58] Kohei Arai and Ali Ridho Barakbah. Hierarchical K-means: an algorithm for centroids initialization for K-means. *Reports of the Faculty of Science and Engineering*, 36(1):25–31, 2007.

[59] Paul S. Bradley and Usama M. Fayyad. Refining initial points for K-means clustering. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 91–99. Morgan Kaufmann Publishers Inc., 1998.

[60] Xiaowen Dong, Pascal Frossard, Pierre Vandergheynst, and Nikolai Nefedov. Clustering on multi-layer graphs via subspace analysis on Grassmann manifolds. *IEEE Transactions on Signal Processing*, 62(4):905–918, 2014.

[61] Sareh Shirazi, Mehrtash T. Harandi, Conrad Sanderson, Azadeh Alavi, and Brian C. Lovell. Clustering on Grassmann manifolds via kernel embedding with application to action analysis. In *2012 19th IEEE International Conference on Image Processing*, pages 781–784. IEEE, 2012.

[62] Pavan Turaga, Ashok Veeraraghavan, Anuj Srivastava, and Rama Chellappa. Statistical computations on Grassmann and Stiefel manifolds for image and video-based recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2273–2286, 2011.

[63] Hasan Ertan Cetingul and René Vidal. Intrinsic mean shift for clustering on Stiefel and Grassmann manifolds. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1896–1902. IEEE, 2009.

[64] Peter Gruber and Fabian J. Theis. Grassmann clustering. In *EUSIPCO 2006: 14th European Signal Processing Conference*, pages 1–5. IEEE, 2006.

[65] Paul S. Bradley and Olvi L. Mangasarian. K-plane clustering. *Journal of Global Optimization*, 16(1):23–32, 2000.

[66] Ignacio Santamaria, Javier Vía, Michael Kirby, Tim Marrinan, Chris Peterson, and Louis Scharf. Constrained subspace estimation via convex optimization. In *EUSIPCO 2017: 25th European Signal Processing Conference*, pages 1200–1204. IEEE, 2017.

[67] Bruce Draper, Michael Kirby, Justin Marks, Tim Marrinan, and Chris Peterson. A flag representation for finite collections of subspaces of mixed dimensions. *Linear Algebra and its Applications*, 451:15–32, 2014.

[68] Dan Claudiu Cireşan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22(12):3207–3220, 2010.

[69] Marion F. Baumgardner, Larry L. Biehl, and David A. Landgrebe. 220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3, Sep 2015.

[70] Nat Dilokthanakul, Pedro A. M. Mediano, Marta Garnelo, Matthew C. H. Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with Gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*, 2016.

[71] Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

[72] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[73] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 478–487. PMLR, 2016.

[74] Jee Cheng Wu and Gwo Chyang Tsuei. Unsupervised cluster-based band selection for hyperspectral image classification. In *Proceedings of the 2013 International Conference on Advanced Computer Science and Electronics Information (ICACSEI 2013)*. Atlantis Press, 2013.

[75] Devis Tuia and Gustavo Camps-Valls. Semisupervised remote sensing image classification with cluster kernels. *IEEE Geoscience and Remote Sensing Letters*, 6(2):224–228, 2009.

[76] Hongjun Su, He Yang, Qian Du, and Yehua Sheng. Semisupervised band clustering for dimensionality reduction of hyperspectral imagery. *IEEE Geoscience and Remote Sensing Letters*, 8(6):1135–1139, 2011.

[77] Sofya Chepushtanova, Christopher Gittins, and Michael Kirby. Band selection in hyperspectral imagery using sparse support vector machines. In *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XX*, volume 9088, 2014.

[78] P.-A. Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009.

[79] Ake Björck and Gene H. Golub. Numerical methods for computing angles between linear subspaces. *Mathematics of Computation*, 27(123):579–594, 1973.

[80] Alan Edelman, Tomás A. Arias, and Steven T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

[81] Xiaofeng Ma, Michael Kirby, Chris Peterson, and Louis Scharf. Self-organizing mappings on the Grassmannian with applications to data analysis in high-dimensions. *Neural Computing and Applications*, submitted 2018.

[82] Tim Marrinan, J. Ross Beveridge, Bruce Draper, Michael Kirby, and Chris Peterson. Flag manifolds for the characterization of geometric structure in large data sets. In *Numerical Mathematics and Advanced Applications-ENUMATH 2013*, pages 457–465. Springer, 2015.

[83] Tim Marrinan, Bruce Draper, J. Ross Beveridge, Michael Kirby, and Chris Peterson. Finding the subspace mean or median to fit your need. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1082–1089. IEEE, 2014.

[84] Arup Kumar Pal and Anup Sar. An efficient codebook initialization approach for LBG algorithm. *arXiv preprint arXiv:1109.0090*, 2011.

[85] Christophe Rosenberger and Kacem Chehdi. Unsupervised clustering method with optimal estimation of the number of clusters: Application to image segmentation. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 1, pages 656–659. IEEE, 2000.

[86] Petra Schneider, Michael Biehl, and Barbara Hammer. Distance learning in discriminative vector quantization. *Neural Computation*, 21(10):2942–2969, 2009.