DISSERTATION

EXPLOITING GEOMETRY, TOPOLOGY, AND OPTIMIZATION FOR KNOWLEDGE DISCOVERY IN BIG DATA

Submitted by

Lori Beth Ziegelmeier

Department of Mathematics

In partial fulfullment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Summer 2013

Doctoral Committee:

Advisor: Michael Kirby
Co-Advisor: Chris Peterson

Jiangguo (James) Liu
Bruce Draper

ABSTRACT

EXPLOITING GEOMETRY, TOPOLOGY, AND OPTIMIZATION FOR KNOWLEDGE DISCOVERY IN BIG DATA

In this dissertation, we consider several topics that are united by the theme of topological and geometric data analysis. First, we consider an application in landscape ecology using a well-known vector quantization algorithm to characterize and segment the color content of natural imagery. Color information in an image may be viewed naturally as clusters of pixels with similar attributes. The inherent structure and distribution of these clusters serves to quantize the information in the image and provides a basis for classification. A friendly graphical user interface called Biological Landscape Organizer and Semi-supervised Segmenting Machine (BLOSSM) was developed to aid in this classification. We consider four different choices for color space and five different metrics in which to analyze our data, and results are compared. Second, we present a novel topologically driven clustering algorithm that blends Locally Linear Embedding (LLE) and vector quantization by mapping color information to a lower dimensional space, identifying distinct color regions, and classifying pixels together based on both a proximity measure and color content. It is observed that these techniques permit a significant reduction in color resolution while maintaining the visually important features of images. Third, we develop a novel algorithm which we call Sparse LLE that leads to sparse representations in local reconstructions by using a data weighted ℓ_1 norm regularization term in the objective function of an optimization problem. It is observed that this new formulation has proven effective at automatically determining an appropriate number of

nearest neighbors for each data point. We explore various optimization techniques, namely Primal Dual Interior Point algorithms, to solve this problem, comparing the computational complexity for each. Fourth, we present a novel algorithm that can be used to determine the boundary of a data set, or the vertices of a convex hull encasing a point cloud of data, in any dimension by solving a quadratic optimization problem. In this problem, each point is written as a linear combination of its nearest neighbors where the coefficients of this linear combination are penalized if they do not construct a convex combination, revealing those points that cannot be represented in this way, the vertices of the convex hull containing the data. Finally, we exploit the relatively new tool from topological data analysis, persistent homology, and consider the use of vector bundles to re-embed data in order to improve the topological signal of a data set by embedding points sampled from a projective variety into successive Grassmannians.

ACKNOWLEDGEMENTS

This dissertation would never have come to fruition without the support and encouragement of several individuals. It is with genuine gratitude that I would like to acknowledge them.

First, I would like to thank my advisors, Drs. Michael Kirby and Chris Peterson, for their invaluable guidance and mentorship. Dr. Kirby continually introduces new concepts and a plethora of interesting data sets to consider, and Dr. Peterson's insightful ideas and way with words have been instrumental. Their dual perspectives have led to a very unique and interesting research path. In addition, I truly appreciate the opportunities to travel to a variety of conferences as well as co-teach a mini-course in Costa Rica.

I am also indebted to my outside committee member Dr. Bruce Draper who participated in several research meetings and introduced new color spaces and algorithms to me.

I cannot forget to include my Pattern Analysis Laboratory (PAL) colleagues, especially Justin Marks and Sofya Chepushtanova, who have been travel companions, idea generators, helpful resources, and indeed pals.

I would also like to thank the front office staff who encouraged me to pursue a graduate degree as well as the many wonderful instructors and mentors that I have had along the way, particularly, Drs. Dan Bates, Patrick Shipman, and James Liu.

Finally, I cannot forget to acknowledge my dear family and friends, many of whom do not quite understand how it is possible to 'research' mathematics but have supported me regardless. To my parents, Carl and Pat Ziegelmeier, who have been there for me from day one and have always encouraged me to challenge myself and strive for excellence. Last, but

certainly not least, I must thank my wonderful husband, Mike Neuberg, whose patience, love, and concern has been a great support to me for all of these years.

TABLE OF CONTENTS

Abstract iii

Acknowledgements v

LIST OF TABLES ix

LIST OF FIGURES x

Chapter 1. Introduction 1

Chapter 2. Background Algorithms 5

2.1. Introduction to Linde-Buzo-Gray 5

2.2. Introduction to Locally Linear Embedding 10

2.3. Metrics 31

Chapter 3. Biological Landscape Organizer and Semi-supervised Segmenting Machine (BLOSSM) 35

3.1. Introduction 35

3.2. Motivation: An Ecology Application 35

3.3. Overview 37

3.4. Color Spaces 39

3.5. Biological Landscape Organizer and Semi-supervised Segmenting Machine 43

3.6. Analysis 46

3.7. Ecology Application 58

3.8. Conclusion 63

Chapter 4. Locally Linear Embedding Clustering Algorithm 65
4.1. Introduction 65
4.2. Connecting Components in Locally Linear Embedding 68
4.3. Locally Linear Embedding Clustering 70
4.4. Implementation 78
4.5. Conclusion 84

Chapter 5. Sparse Locally Linear Embedding 86
5.1. Introduction 86
5.2. Sparse Locally Linear Embedding 89
5.3. Convex Optimization Problems 96
5.4. Solving Sparse LLE 99
5.5. The Algorithm 102
5.6. Sparsity Example 103
5.7. Swiss Roll Example 108
5.8. Fabry-Perot Data Set 110
5.9. Gene Expression Influenza Data 123
5.10. Conclusion 126

Chapter 6. Primal Dual Interior Point Method 129
6.1. Introduction 129
6.2. Central Path 129
6.3. Lagrange Multipliers 131
6.4. The Dual 132
6.5. Primal Dual Interior Point Method 134

6.6. Reducing the KKT System 138
6.7. Vanderbei PDIP Formulation 141
6.8. Complexity Analysis 147
6.9. Computational Complexity of Sparse LLE 150

Chapter 7. Detecting the Vertices of a Convex Hull Encasing a Point Cloud of Data .. 155
7.1. Introduction 155
7.2. Thought Experiment 156
7.3. Optimization Problem 158
7.4. Implementation 167
7.5. Conclusion 174

Chapter 8. Strengthening of Topological Signals in Persistent Homology through Vector Bundle Based Maps 176
8.1. Introduction 176
8.2. Background 177
8.3. Main Idea 183
8.4. Conclusion 189

Chapter 9. Conclusion 191

Bibliography 195

Appendix A. Further Analysis of Images Using BLOSSM 205

LIST OF TABLES

3.1 Table of characteristics regarding all clusters of pixels. 63

3.2 Table of characteristics regarding clusters of pixels with all green clusters combined into a single group. 63

5.1 Number of 'zero' weights as defined by those values less than the tolerance designated. 119

8.1 Persistent homology data 179

LIST OF FIGURES

2.1 Voronoi sets of 100 randomly distributed points (green) with 10 randomly identified centers (blue). 7

2.2 Sample points from the data set of 400 points in $\mathbb{R}^{400}$ created by traversing and wrapping a 10×10 black square in a 20×20 random square. 31

2.3 A plot of the embedding vectors obtained by LLE of 400 points in $\mathbb{R}^{400}$ reduced down to $\mathbb{R}^3$. 31

2.4 Illustration of metric choices with the circle representing Euclidean, the diamond representing Taxicab, the square representing Chebyshev, and the ellipse representing Mahalanobis. 34

3.1 Sample image of landscape data set. 38

3.2 Illustration of the red, green, and blue sheets associated to pixels of an image. . 40

3.3 Illustration of creating a data matrix from RGB data. 41

3.4 BLOSSM implemented on an image with 8 centers manually selected by the user to reflect distinct colors within image. 47

3.5 Reconstructions using the LBG algorithm with fixed color space RGB and varying the metric used for still life image. 49

3.6 Distortion errors using the LBG algorithm with fixed color space RGB and varying the metric used for still life image. 50

3.7 Reconstructions using the LBG algorithm with fixed color space Quantized RGB and varying the metric used for still life image. 51

3.8 Distortion errors using the LBG algorithm with fixed color space Quantized RGB and varying the metric used for still life image. 52
3.9 Reconstructions using the LBG algorithm with fixed color space Named Color and varying the metric used for still life image. 54
3.10 Distortion errors using the LBG algorithm with fixed color space Named Color and varying the metric used for still life image. 55
3.11 Reconstructions using the LBG algorithm with fixed color space CIELab and varying the metric used for still life image. 56
3.12 Distortion errors using the LBG algorithm with fixed color space CIELab and varying the metric used for still life image. 57
3.13 Characteristic GUI, displaying the reconstruction image and final centers found after clustering. 59
3.14 Distribution of clusters. 59
3.15 Individual clusters. 60
3.16 Characteristics regarding the yellow cluster of pixels. 62

4.1 Images generated by Pattern Analysis Laboratory at Colorado State University where an individual remains motionless and the illumination of the surrounding area varies. 72
4.2 Plots of 3D data points generated by extracting the RGB values of a single pixel for each of 200 images and their corresponding embedding vectors as reconstructed by the LLE algorithm using k=8 nearest neighbors, and k=4 nearest neighbors with 'connected' data points. 73
4.3 Illustration of subspace segmentation algorithm. 76

4.4 Subspace segmentation of 2D reconstruction data using LLEC with accuracy tolerances of approximately 0.4 by initializing $\mathbf{y}^*$ as the point whose b^{th} nearest neighbor has the smallest distance. 77

4.5 Reconstruction images of LLEC with variances tolerances. 81

4.6 Reconstruction images after quantizing the color space of original image with LBG using indicated method to determine the centers. Note that the respective distortion errors of each implementation with 15 iterations are: 140.0250, 342.6351, 219.0756, and 146.7013. 83

4.7 Quantizing the color space of the original image with LBG using indicated method to determine the centers. Note that the respective distortion errors of these two implementations with 15 iterations are: (1st Original) 210.3490 and 210.6900, (2nd Original) 140.0250 and 146.7013, (3rd Original) 172.5580 and 170.7743. 85

5.1 If K is too large, points in oval A could potentially be represented by nearest neighbors in oval B. 87

5.2 Original data in $\mathbb{R}^3$ 88

5.3 LLE reconstruction plots in $\mathbb{R}^2$ using various choices for the number of nearest neighbors K 88

5.4 Considering weights associated to the central (black) pixel. Using $K = 20$. Nearest neighbor pixels associated to nonzero weights are colored gray and those associated to zero weights are colored white. 105

5.5 Plot of weights associated to central pixel, allowing $K = 20$ nearest neighbors and varying λ. 106

5.6 Plot of the number of nonzero weights associated to nearest neighbors of the central pixel versus λ. 106

5.7 Plot of the reconstruction error versus λ. 107

5.8 Plot of the sparsity term, the 'risk', $\sum_{j \in N_i} |w_j| f(d(\mathbf{x}_i, \mathbf{x}_j))$ versus the reconstruction error, the 'reward', $\|\mathbf{x}_i - \sum_{j \in N_i} w_j \mathbf{x}_j\|_2^2$. Note that blue corresponds to smaller λ values while red corresponds to larger λ values. 107

5.9 2000 randomly generated points along the swiss roll in $\mathbb{R}^3$. 108

5.10 Points in $\mathbb{R}^3$ sampled from the swiss roll embedded into $\mathbb{R}^2$ using indicated algorithms and parameters, where Full indicates using the full $K \times p$ matrix W and Zeroed indicates zeroing those entries less than 10^{-4} in W. 110

5.11 Histograms of the number of nonzero weights associated to each point where the Sparse LLE weight matrix W had entries less than 10^{-4} zeroed out. 111

5.12 Number of nonzero weights associated to nearest neighbors as both λ and K are varied. 112

5.13 One sheet of a sample data cube. 113

5.14 Illustration of considering a single pixel in $\mathbb{R}^{20}$ through time. 114

5.15 All 20 sheets of a pre-processed sample data cube. 115

5.16 Frame 70 of GAA clustered using LBG. 116

5.17 Reducing the 20-dimensional Fabry-Perot data down to $\mathbb{R}^2$ using standard LLE with varying K values. 117

5.18 Reducing the 20-dimensional Fabry-Perot data down to $\mathbb{R}^2$ using Sparse LLE with fixed K values and varying λ. 118

5.19 Reducing the 20-dimensional Fabry-Perot data down to $\mathbb{R}^2$ using Sparse LLE with varying K values and a fixed $\lambda = 0.05$. 119

5.20 Reducing the 20-dimensional Fabry-Perot data down to $\mathbb{R}^2$ using Sparse LLE with varying K values and a fixed $\lambda = 0.1$. 120

5.21 Reducing the 20-dimensional Fabry-Perot data down to $\mathbb{R}^2$ using standard LLE and Sparse LLE with fixed $K = 40$ and $\lambda = 0.1$, but varying the cut off of zero, i.e. any $w_{ij} < tol$ will be zeroed out. 121

5.22 Number of nonzero weights associated to nearest neighbors as both λ and K are varied. 122

5.23 Reducing the 12023-dimensional data down to $\mathbb{R}^2$ using LLE while varying K. . 125

5.24 Reducing the 12023-dimensional data down to $\mathbb{R}^3$ using LLE with $K = 6$. By visual inspection, this choice of K appears to provide the best separation between sick and healthy individuals. 126

5.25 Reducing the 12023-dimensional data down to $\mathbb{R}^2$ using Sparse LLE while varying K and λ values. 127

5.26 Number of nonzero weights associated to nearest neighbors as both λ and K are varied. 128

6.1 Average number of iterations for 100 trials, varying the size of system solves to be $m \times 2m$, with derivations as described in Section 6.6. 149

6.2 Average CPU time for 100 trials, varying the size of system solved to be $m \times 2m$, with derivations as described in Section 6.6. 150

6.3 Average number of iterations for 100 trials, varying the size of system solved to be $m \times 2m$, with derivations as described in Section 6.7. 151

6.4 Average CPU time for 100 trials, varying the size of system solved to be $m \times 2m$, with derivations as described in Section 6.7. 152

6.5 Average number of iterations and CPU time for 100 trials, varying the size of system solved to be $m \times 2m$. Comparing all derivations discussed in Sections 6.6 and 6.7. 153

7.1 Illustration of reconstructing an interior point versus a vertex by a set of nearest neighbors. 158

7.2 The data set consists of an initial 50 random points augmented by an additional 10 boundary points (magenta). The vertices (red) are determined by the Quickhull algorithm. 168

7.3 Implementation of the algorithm with various parameter choices. Cyan points have a negative weight for the given parameter choices. 169

7.4 Data points colored according to magnitude of the two norm where blue corresponds to the smallest values and red corresponds to the largest values. . . 171

7.5 Plot of 2000 points randomly selected points with the 8 additional vertices of the cube added. Cyan points indicate negative weight, varying parameter λ. . . . 172

7.6 Plot of Euclidean norm of weights associated to each point with parameters $\gamma = 10^{-5}$ and $\lambda = 0.025$. 172

7.7 Plot of distance to the boundary of the cube versus the Euclidean norm of weights associated to each point with parameters $\gamma = 10^{-5}$ and $\lambda = 0.025$. 173

7.8 Plot of each point in the cube colored according to magnitude of the corresponding weight vector with $0 \leq ||\mathbf{w}||_2^2 \leq 0.25$ orange, $0.25 \leq ||\mathbf{w}||_2^2 \leq 0.5$

yellow, $0.5 \leq ||\mathbf{w}||_2^2 \leq 0.75$ green, $0.75 \leq ||\mathbf{w}||_2^2 \leq 1$ cyan, $1 \leq ||\mathbf{w}||_2^2 \leq 1.25$ light blue, $1.25 \leq ||\mathbf{w}||_2^2 \leq 1.5$. 173

7.9 Implementation of algorithm with various parameter choices on convex combination created using chemical signatures. Points represented in cyan have a negative weight. 175

8.1 A sequence of Vietoris-Rips simplicial complexes shown geometrically and abstractly along with their maximal faces. 180

8.2 Barcodes corresponding to Figure 8.1 181

8.3 Betti-1 barcodes for each of the four specified embeddings. 188

8.4 Average ratio of the sum of the longest two barcode lengths to the sum of the lengths of all barcodes for $k = 1, \ldots, 10$. 189

8.5 Average ratio of the second longest barcode to the third longest barcode for $k = 1, \ldots, 10$. 189

A.1 BLOSSM implemented on an image with 9 centers manually selected by the user to reflect distinct colors within image. 206

A.2 BLOSSM implemented on an image with 14 centers manually selected by the user to reflect distinct colors within image. 206

A.3 Reconstructions using the LBG algorithm with fixed color space RGB and varying the metric used for car image. 208

A.4 Distortion errors using the LBG algorithm with fixed color space RGB and varying the metric used for car image. 209

A.5 Reconstructions using the LBG algorithm with fixed color space Quantized RGB and varying the metric used for car image. 210

A.6 Distortion errors using the LBG algorithm with fixed color space Quantized RGB and varying the metric used for car image. 211
A.7 Reconstructions using the LBG algorithm with fixed color space Named Color and varying the metric used for car image. 212
A.8 Distortion errors using the LBG algorithm with fixed color space Named Color and varying the metric used for car image. 213
A.9 Reconstructions using the LBG algorithm with fixed color space CIELab and varying the metric used for car image. 215
A.10 Distortion errors using the LBG algorithm with fixed color space CIELab and varying the metric used for car image. 216
A.11 Reconstructions using the LBG algorithm with fixed color space RGB and varying the metric used for still life image. 217
A.12 Distortion errors using the LBG algorithm with fixed color space RGB and varying the metric used for still life image. 218
A.13 Reconstructions using the LBG algorithm with fixed color space Quantized RGB and varying the metric used for still life image. 220
A.14 Distortion errors using the LBG algorithm with fixed color space Quantized RGB and varying the metric used for still life image. 221
A.15 Reconstructions using the LBG algorithm with fixed color space Named Color and varying the metric used for still life image. 222
A.16 Distortion errors using the LBG algorithm with fixed color space Named Color and varying the metric used for still life image. 223

A.17Reconstructions using the LBG algorithm with fixed color space CIELab and varying the metric used for still life image. 224
A.18Distortion errors using the LBG algorithm with fixed color space CIELab and varying the metric used for still life image. 225

CHAPTER 1

Introduction

Geometric and topological structure has been widely observed in the context of massive data sets, giving rise to the need for mathematically-based algorithms. Knowledge discovery of these large data sets is an area of mathematics at the intersection of linear algebra, geometry, topology, computing, data mining, statistics, optimization, and signal processing. With computational power now available, knowledge discovery of large data sets may be uncovered using mathematical techniques such as manifold learning. Manifold learning is a data analysis approach that assumes that the observations, taken as a whole, possess some geometric structure.

One problem in manifold learning is characterizing the color content of natural imagery. As natural images are not random, there exists correlation between pixels that gives rise to structure. It has been said that correlation is the footprint of low dimensionality, and thus, exploiting this correlation reveals structure within a data set. The color content of images may be viewed naturally as clusters of pixels in color space that are correlated, and the inherent structure and distribution of these clusters affords a quantization of the information in the image.

In this thesis, we discuss two algorithms for color content quantization, one a well-known algorithm applied to a new area and the other a way to combine existing algorithms into a novel technique. We also discuss a new derivation of a manifold learning algorithm that better reflects the topological structure of a data set and a method to determine the convex

hull from a point cloud of data of any dimension. Finally, we discuss a multiscale algorithm that looks for structure in data sets at different scales to observe which features persist in each scale and a way to improve the topological signal in each of these scales.

In Chapter 3, we discuss an application of a well-known clustering algorithm to landscape ecology. A common problem in landscape ecology is to ascertain the flower cover of a particular type of flower within a region and understand spatial relationships of foliage and flowers within the region. Using the well-studied Linde-Buzo-Gray vector quantization algorithm [54], we have developed a friendly graphical user interface dubbed BLOSSM (Biological Landscape Organizer and Semi-Supervised Segmenting Machine) that quantizes the pixels of an image and thus, the color space. We consider this algorithm's performance using four choices of color space (RGB, Quantized RGB, Named Color, and CIELab) and five similarity measures (ℓ_1, ℓ_2, ℓ_∞, Mahalanobis distance, and spectral angle). Analysis on a sample landscape image, comparing the choice of color space and metric on the clustering. Important features pertaining to a cluster associated to a single color such as number of pixels, the percent of the total area, the number of contiguous regions, and the average number of pixels in each contiguous region can be determined. The entropy of a given reconstruction is also uncovered. Analysis of two additional images can be found in Appendix A.

As the transition of the color values of neighboring pixels in pixel space of a given hue is typically smooth, we can associate these smooth variations of pixel colors of a given hue to be lying on a manifold, at least approximately. Blending the topologically-driven manifold learning Locally Linear Embedding algorithm [71], Principal Component Analysis [50], and the Linde-Buzo-Gray algorithm [54], we have developed a novel clustering algorithm that reveals the geometric structure associated with smooth variations in the pixel distribution of

images and identifies underlying submanifolds which allow for segmentation and quantization of the color space of images. This technique permits a significant reduction in color resolution while maintaining the visually important features of images. This new algorithm, called Locally Linear Embedding Clustering is discussed in Chapter 4.

In Chapter 5, we discuss a new modified version of the LLE algorithm. In the standard LLE algorithm, the error of the squared Euclidean distance between a data point and its reconstruction is minimized. We consider a reformulation of this problem by considering this error with an ℓ_1 regularization term added to the objective function in the second step to determine the weights, in order to penalize weights of nearest neighbors that are not actually very similar. The sparsity property induced by the ℓ_1 norm is exploited in this regularization. LLE has the artifact that if K nearest neighbors are allowed, then all of the weights associated to these nearest neighbors will be nonzero. In our reformulation, many weights are driven to zero, allowing points to automatically be constructed by an appropriate number of nearest neighbors. We call this modified problem Sparse LLE.

We discuss in Chapter 6 a method to solve Sparse LLE known as the Primal Dual Interior Point algorithm. A variety of formulations is presented and the computational complexity of each formulation is explored.

A new algorithm to determine boundaries of a data set is discussed in Chapter 7. This algorithm arose from reconstructing data points by a choice of nearest neighbors and preferring those that fall within a convex hull of its neighbors. Those points that are not able to fall within a convex hull of its neighbors (i.e. cannot be represented as a convex combination) are deemed to be points lying on the boundary, or vertices of the convex hull containing all data points.

Finally, Chapter 8 discusses the relatively new tool from topological data analysis, persistent homology. For many, the way data sets (and the information contained in those sets) are viewed have been changed by persistent homology. It is derived directly from techniques in computational homology but has the added feature that it is able to capture structure at multiple scales. One way that this multi-scale information can be presented is through a barcode. A barcode consists of a collection of line segments each representing the range of parameter values over which a generator of a homology group persists. A segment's length relative to the length of other segments is an indication of the strength of a corresponding topological signal. In this paper, we consider how vector bundles may be used to re-embed data as a means to improve the topological signal. As an illustrative example, we construct maps of tori to a sequence of Grassmannians of increasing dimension. We equip the Grassmannian with the geodesic metric and observe an improvement in barcode signal strength as the dimension of the Grassmannians increase.

First though, in Chapter 2, we will present a couple of background algorithms that will be used in a variety of places throughout this dissertation.

CHAPTER 2

Background Algorithms

In this chapter, we will discuss some background algorithms and concepts that will be used throughout this disseratation. This material is included to make this a self-contained document but is not part of the original contribution. We will first describe the well-known vector quantization algorithm known as the Linde-Buzo-Gray algorithm which is similar to K-means. Next, we will discuss in detail the Locally Linear Embedding algorithm, including the derivation, proofs of invariance relationships, methods for determining parameter values, and an example of nonlinear dimensionality reduction. Finally, we describe a few metrics that will be used throughout this dissertation.

2.1. Introduction to Linde-Buzo-Gray

The Linde-Buzo-Gray (LBG) algorithm [54] is a vector quantization, competitive learning algorithm. The idea of competitive learning is that a set of classification vectors "compete" for the privilege to react to a subset of the input data in an effort to distribute vectors in a way that reflects the probability distribution of the input signals. The LBG algorithm in its standard form is unsupervised, meaning that it can identify major characteristics or patterns without any information from the user such as data labels.

In essence, the algorithm determines all points that fall within a region of a classification vector which we call a center and label $\mathbf{c}_i$, calculates the mean of all points falling within this

region, and then updates the center of this set to be equal to the mean. The entire process is then iterated until all data points are quantized to an appropriate degree of accuracy.

To make this idea more precise, we need to introduce the idea of a Voronoi set. Voronoi sets are determined by identifying a set of vectors called centers in which to cluster the data. Then, each point within the data set is compared to each of the centers by computing the distance according to some metric between each point and each center. Each point is identified with the center it is closest to, i.e. the center with the smallest distance.

Definition 1. *A **Voronoi set** S_i is the subset of points of a data set X for which center $\boldsymbol{c}_i$ is the winner, meaning*

$$S_i = \{\boldsymbol{x} \in X \mid \|\boldsymbol{x} - \boldsymbol{c}_i\| \leq \|\boldsymbol{x} - \boldsymbol{c}_j\| \ \forall \ j \neq i\}$$

Here, we define $\|\mathbf{x} - \mathbf{c}_i\|$ to be the standard Euclidean distance i.e. if $\mathbf{x} = (x_1, x_2, x_3, \ldots, x_n)$ and $\mathbf{c}_i = (c_1^i, c_2^i, c_3^i, \ldots, c_n^i)$, then

$$\|\mathbf{x} - \mathbf{c}_i\| = \sqrt{(x_1 - c_1^i)^2 + (x_2 - c_2^i)^2 + (x_3 - c_3^i)^2 + \ldots + (x_n - c_n^i)^2}.$$

Therefore, the Voronoi set S_1 is the set of all points satisfying $\|\mathbf{x} - \mathbf{c}_1\| \leq \|\mathbf{x} - \mathbf{c}_j\|$ for all $j \neq 1$, S_2 is the set of all points satisfying $\|\mathbf{x} - \mathbf{c}_2\| \leq \|\mathbf{x} - \mathbf{c}_j\|$ for all $j \neq 2$, etc. Furthermore, the Voronoi set S_1 can be thought of the set of all points closer to center $\mathbf{c}_1$ than to all other centers, the Voronoi set S_2 can be thought of the set of all points closer to center $\mathbf{c}_2$ than to all other centers, etc. Do note that any distance metric could be used as well.

The partitioning of data points within an input space is known as a Voronoi tessellation. Refer to Figure 2.1 to observe 100 randomly distributed points (green) partitioned into 10 Voronoi sets associated to each of the randomly generated centers (blue).

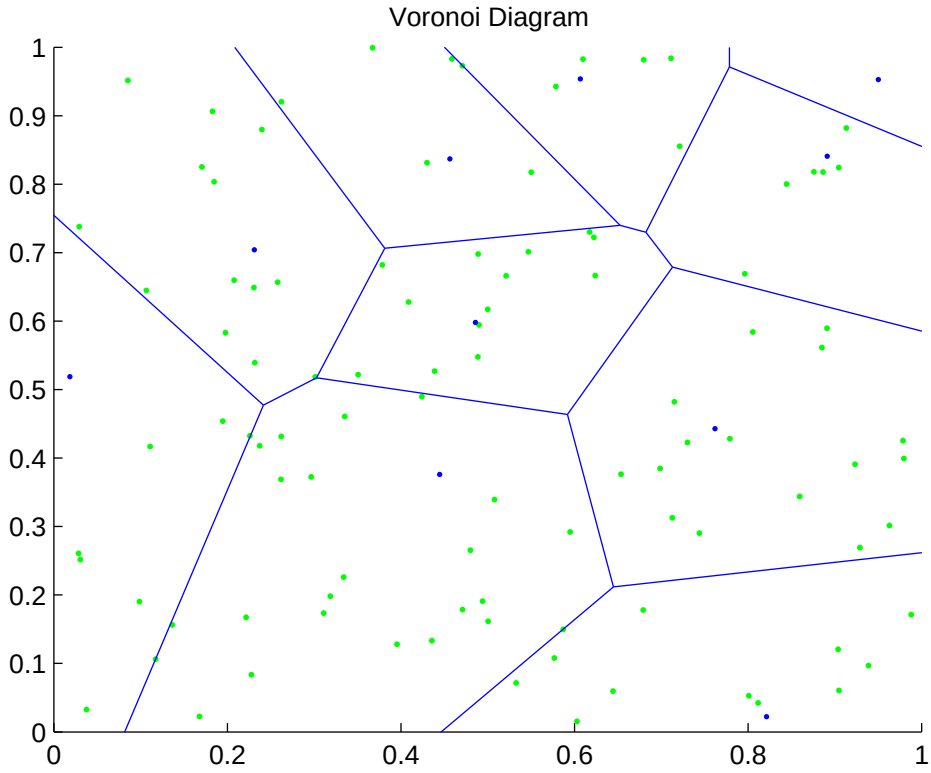


FIGURE 2.1. Voronoi sets of 100 randomly distributed points (green) with 10 randomly identified centers (blue).

The importance of defining a Voronoi set is that we have a discrete subset of points that we can work with. We calculate the mean of each Voronoi set as

$$\mu_i = \frac{1}{|S_i|} \sum_{\mathbf{x} \in S_i} \mathbf{x}$$

where $|S_i|$ is the number of data points within the Voronoi set S_i. This mean will be used to identify new center vectors for the next iteration of the algorithm.

The desired result of the LBG algorithm is that all of the input data will be partitioned into Voronoi sets, or clusters, in such a way that points within a cluster are similar, and thus, can be identified with the prototype of the cluster, the center. Now we are ready to outline the LBG algorithm as follows:

LBG Algorithm

(1) Identify a set of center vectors.

(2) Determine the Voronoi sets S_i for each center $\mathbf{c}_i$.

   (a) Present a vector $\mathbf{x}_j$ to the network.

   (b) Calculate the distance of $\mathbf{x}_j$ to each center $\mathbf{c}_i$,

   (c) Determine the index, k, of the center vector with the minimum distance to $\mathbf{x}_j$. That is, $k = \text{argmin}_{i \in I} \|\mathbf{x}_j - \mathbf{c}_i\|$ where I is the set of center indices.

   (d) Identify the winning center $\mathbf{c}^* = \mathbf{c}_k$.

   (e) The vector $\mathbf{x}_j$ is then an element of the Voronoi set S_k associated to center $\mathbf{c}_k$.

(3) Repeat steps (a) through (e) until each vector in the input data set has been considered.

(4) Update each of the center vectors by calculating the mean of all points within each Voronoi set and equating the new center vector with the mean

$$\mathbf{c}_i = \frac{1}{|S_i|} \sum_{\mathbf{x} \in S_i} \mathbf{x}.$$

(5) Repeat steps 2. through 4. Until some convergence criterion is achieved.

The algorithm may be repeated for a finite set of iterations, until assignment of each data point does not change from one iteration to the next, or until some tolerance has been met. One such stopping criterion could be if the error of the reconstruction is small enough, then the algorithm may terminate. The distortion error is one measurement of reconstruction error.

Definition 2. *The* ***distortion error*** *of a data set X consisting of p points with regard to a set of centers labeled by indices J is*

$$E(X, J) = \frac{1}{p} \sum_{j \in J} \sum_{\boldsymbol{x} \in S_j} \|\boldsymbol{x} - \boldsymbol{c}_j\|^2$$

Note that if we let X^* indicate the matrix of points each identified with their winning centers, then the distortion error could also be calculated as $\frac{1}{p}\|X - X^*\|_F^2$ where

$$\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$$

is the Frobenius norm.

Note that we have not discussed how to initialize the original center vectors around which the data is clustered. Proper initialization is a crucial issue for any iterative algorithm and can greatly affect the outcome. Therefore, initializing the center vectors must be done with care. One could define the centers as points on a lattice or grid, define centers with certain properties for a particular data set, choose random data points within the data set, or use

random vectors that may or may not be a subset of the original data set. Depending on the distribution of the original data set and the distribution of the centers, some Voronoi sets may have a large number of points within them and some may be empty. Note that [7] indicates that in a supervised or semi-supervised problem assignment constraints can be used to avoid empty or unnecessary clusters. Another important parameter is the choice of the number of centers around which to partition the data. If an inappropriate choice is made, poor results may occur. As one might guess, the increase in the number of centers often allows for a decrease in the distortion error of the data.

2.2. Introduction to Locally Linear Embedding

The Locally Linear Embedding (LLE) algorithm [71] is an unsupervised dimensionality reduction algorithm that determines a mapping of data, lying in a higher dimensional vector space, to a lower dimensional vector space while optimizing the maintenance of local spatial relationships within the data. Through this map, the LLE algorithm uncovers a lower dimensional representation of the data with the goal of preserving the topology and neighborhood structure of the original higher dimensional data.

Given a data set, X, consisting of p points of dimension D as input vectors, the LLE algorithm determines p embedding vectors of dimension d, where $d < D$, that reflect the structure of the original data. The first step in implementing the LLE algorithm is to determine the neighbors associated to each of the high dimensional data points. Typically, this is done by determining a fixed number of nearest neighbors K by considering those data points with the smallest distance determined by the metric being used, often Euclidean distance.

The second step of the algorithm is to associate to each neighbor a weight. This weight is calculated by solving a least squares problem that minimizes a certain reconstruction error $\epsilon(W)$. More precisely, if $\mathbf{x}_i$ denotes the i^{th} data point from a set of p points, and N_i denotes the indices of its set of nearest neighbors, one determines the values of w_{ij} that minimize the expression

$$\epsilon(W) = \sum_{i=1}^{p} \|\mathbf{x}_i - \sum_{j \in N_i} w_{ij}\mathbf{x}_j\|^2$$

where W denotes the $p \times p$ matrix containing the weights w_{ij} (padded out with zeros corresponding to points not identified as nearest neighbors). Note that the expression $\epsilon(W)$ is minimized subject to the constraint that for each i, $\sum_{j \in N_i} w_{ij} = 1$. This constraint ensures that the weights are invariant to translations and the form of the error ensures the weights are also invariant to rescalings and rotations.

The final step of the algorithm is to determine a set of lower dimensional vectors, $\mathbf{y}_i$, that minimize the function

$$\phi(\mathbf{Y}) = \sum_{i=1}^{p} \|\mathbf{y}_i - \sum_{j \in N_i} w_{ij}\mathbf{y}_j\|^2$$

using only the local geometry obtained by the weights. This function in minimized subject to the constraint that the data is sphered or whitened, $\frac{1}{p}\sum_{i=1}^{p} \mathbf{y}_i\mathbf{y}_i^T = I$, and it is centered around the origin $\sum_{i=1}^{p} \mathbf{y}_i = \mathbf{0}$. It can be shown that this optimization problem corresponds to the eigenvector problem $MY^T = Y^T\Lambda$ where $M = I - W - W^T + W^TW = (I - W)^T (I - W)$ and Λ is the diagonal matrix of Lagrange multipliers. The minimal solution is the bottom d eigenvectors (i.e. those corresponding to the smallest nonzero eigenvalues of M). The i^{th} row of Y^T corresponds to $\mathbf{y}_i$.

If our data set corresponds to a sampling of a manifold and if this sampling is sufficiently dense, then a fundamental assumption of the algorithm is that each data point and its nearest neighbors can be characterized by a locally linear patch of the manifold, hence the name Locally Linear Embedding. Therefore, in the second step of the algorithm, each $\mathbf{x}_i$ is approximated by a linear combination of its neighbors. Data points that were close together in the original higher dimensional space should still be close together after mapped to lie in the lower dimensional space thus preserving the topology of the original data set. In the following subsections, we derive the algorithm in more detail.

2.2.1. Nearest Neighbors. The first step in implementing the LLE algorithm is to determine the neighbors associated to each of the high dimensional data points. Determining the nearest neighbors of a specific data point involves finding those data points that are the most similar. One way to measure similarity is to use a Euclidean distance metric (however, other metrics may also be used). The most straightforward way to perform this task is to determine a fixed number of nearest neighbors by considering those data points with the smallest distance determined by the metric being used. Alternatively, nearest neighbors can be identified by classifying as neighbors all data points that fall within a ball of fixed radius about each point. Therefore, the number of nearest neighbors could differ for each data point. However, if the radius of each ball is chosen to be too small, some points may be isolated. In this paper, the nearest neighbors of each point was found by determining a fixed number, K, data points with the smallest non-zero Euclidean distance from the original point.

Determining the fixed number of neighbors, K, is the only free parameter in the LLE algorithm. There is some art in choosing an appropriate value for K. If the manifold is well-sampled, then each data point and its nearest neighbors lie approximately on a locally linear piece of the manifold. However, if the number of nearest neighbors, K, is chosen to be too large, the region may no longer be linear and might include points which are geodesically far away. However, choosing K to be too small may be problematic as the eigenvector problem to determine the embedding vectors becomes singular. ALso, the span of a set of K points is a linear space of dimension at most $K - 1$. Therefore, the dimension of the target vector space, d, should be chosen to be strictly less than the number of nearest neighbors. Note that [71] did not give guidance on how to choose an appropriate number of nearest neighbors. However, [57] gives an hierarchical approach to automatically select an optimal parameter value which has been shown to be quite precise, although computationally intensive. This will be discussed further in 2.2.4.2

Another issue arises when the dimension of the original high dimensional data, D, is larger than K. Section 2.2.4.3 will discuss this complication as well as a way to determine an appropriate value for K. Finally, it is important to realize that the embedding reconstruction greatly depends on the number of neighbors being used.

2.2.2. Least Squares Problem to Find Weights. The second step of the LLE algorithm is to determine the weights used to associate each point with its nearest neighbors. This can be done by minimizing the distance between a point and a linear combination of all of its nearest neighbors where the coefficients of this linear combination are defined by the weights. Let N_i be the set of neighbors associated to a single point $\mathbf{x}_i$, let p be the number of data points being considered, and let D denote the dimension of the ambient space of

the data. Our goal then is to determine the weights, w_{ij}, associated to each point, $\mathbf{x}_i$, and each of its nearest neighbors, $\mathbf{x}_j \in N_i$. Note that the weight, w_{ij}, between two points that are not nearest neighbors is defined to be 0. Thus, a data point can only be reconstructed from points determined to be its nearest neighbors. Now, the weights are determined by minimizing differences between each point and a linear combination of the nearest neighbors to the point. Let W denote the matrix of weights with entries w_{ij}. The cost function of the reconstruction error is then given by

$$\epsilon(W) = \sum_{i=1}^{p} \|\mathbf{x}_i - \sum_{j \in N_i} w_{ij}\mathbf{x}_j\|^2. \tag{2.1}$$

For each i, a constraint, $\sum_{j \in N_i} w_{ij} = 1$, is implemented to ensure that these weights are invariant to translations. Note that the form of the errors ensures the weights are also invariant to rescalings and rotations. Proofs of these transformation invariance relations can be found in Section 2.2.4.1. Using the sum-to-one constraint, the constraint that $w_{ij} = 0$ if $\mathbf{x}_j$ is not in the set N_i, and a little linear algebra, we see that

$$\begin{aligned}
\epsilon(W) &= \sum_{i=1}^{p} \|\mathbf{x}_i - \sum_{j \in N_i} w_{ij}\mathbf{x}_j\|^2 \\
&= \sum_{i=1}^{p} \| \sum_{j \in N_i} w_{ij}(\mathbf{x}_i - \mathbf{x}_j)\|^2 \\
&= \sum_{i=1}^{p} \left(\sum_{j \in N_i} w_{ij}(\mathbf{x}_i - \mathbf{x}_j) \right)^T \left(\sum_{j \in N_i} w_{ij}(\mathbf{x}_i - \mathbf{x}_j) \right) \\
&= \sum_{i=1}^{p} \left(\sum_{j,k \in N_i} w_{ij} w_{ik} c^i_{jk} \right)
\end{aligned}$$

where

$$c_{jk}^{i} = (\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_k). \tag{2.2}$$

This notation represents a $K \times K$ covariance matrix C_i defined for each point x_i with j, k entry c_{jk}^{i}.

Now, we want to minimize these errors using the constraint $\sum_{j \in N_i} w_{ij} = 1$. This can be done using Lagrange Multipliers. Thus, our optimization problem

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \sum_{i=1}^{p} \left(\sum_{j,k \in N_i} w_{ij} w_{ik} c_{jk}^{i} \right) \\ \text{subject to} \quad & \sum_{j \in N_i} w_j = \ 1 \end{aligned}$$

becomes

$$\underset{\mathbf{w}}{\text{minimize}} \sum_{i=1}^{p} \left(\sum_{j,k \in N_i} w_{ij} w_{ik} c_{jk}^{i} \right) - \sum_{i=1}^{p} \lambda_i \left(\sum_{j \in N_i} w_{ij} - 1 \right)$$

where λ_i are the Lagrange multipliers for each contraint. Fix i in order to determine the weights associated to the neighbors of a single point, x_i, with associated Lagrange multiplier λ.

$$\min \sum_{j,k \in N_i} w_j w_k c_{jk} - \lambda \left(\sum_{j \in N_i} w_j - 1 \right)$$

This optimization problem can be solved by finding the critical values of this cost function. Thus if we differentiate with respect to each weight, we have:

$$\begin{aligned}
&\frac{\partial}{\partial w_m}\left(\sum_{j,k\in N_i} w_j w_k c_{jk} - \lambda\left(\sum_{j\in N_i} w_j - 1\right)\right) \\
= &\sum_{j,k\in N_i}\left(\frac{\partial w_j}{\partial w_m} w_k c_{jk} + w_j \frac{\partial w_k}{\partial w_m} c_{jk}\right) - \lambda \sum_{j\in N_i} \frac{\partial w_j}{\partial w_m} \\
= &\sum_{j,k\in N_i} \delta_{jm} w_k c_{jk} + \sum_{j,k\in N_i} w_j \delta_{km} c_{jk} - \lambda \sum_{j\in N_i} \delta_{jm} \\
= &\sum_{k\in N_i} w_k c_{mk} + \sum_{j\in N_i} w_j c_{jm} - \lambda
\end{aligned}$$

where δ_{ij} corresponds to the Kroeneker delta in which $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. Setting this equation to 0 and solving we have

$$\sum_{k\in N_i} w_k c_{mk} + \sum_{j\in N_i} w_j c_{jm} - \lambda = 0$$

which implies

$$2\sum_{k\in N_i} w_k c_{mk} = \lambda.$$

Finally, we have

$$\sum_{k\in N_i} w_k c_{mk} = \frac{\lambda}{2}.$$

Now, differentiating our original equation with respect to λ we have

$$\frac{\partial}{\partial \lambda}\left(\sum_{j,k\in N_i} w_j w_k c_{jk} - \lambda\left(\sum_{j\in N_i} w_j - 1\right)\right) = \sum_{j\in N_i} w_j - 1.$$

Setting this equal to 0, yields

$$\sum_{j\in N_i} w_j = 1.$$

Since we know that weights are invariant under rescalings we can scale them such that

$$\tilde{w_k} = \frac{2}{\lambda} w_k.$$

Putting both of these derivatives together we can solve the system of equations

$$\begin{cases} \sum_{k \in N_i} \tilde{w_k} c_{mk} = 1 \\ \sum_{k \in N_i} \tilde{w_k} = 1 \end{cases}$$

which yields

$$C_i \tilde{\mathbf{w}} = \mathbf{e} \tag{2.3}$$

where C_i corresponds to the covariance matrix determined by $c^i_{jk} = (\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_k)$, $\tilde{\mathbf{w}}$ is the column vector of weights associated to a single point, and $\mathbf{e}$ is the vector of all ones. Thus in order to find the reconstruction weights, it is only necessary to solve

$$\tilde{\mathbf{w}} = C_i^{-1} \mathbf{e}$$

for each i where the weights are rescaled so that they sum to one. Thus, we have derived the least squares problem to determine the weights that reconstruct the high dimensional data points of dimension D to the lower dimension embedding data points of dimension d. We can form a weight matrix, W, where each row, i, corresponds to the weights between the point $\mathbf{x}_i$ and every other point. Note that W is extremely sparse as the weight between any two points that are not nearest neighbors is defined to be zero.

2.2.3. Eigenvector Problem. The third and final step of the LLE algorithm is to determine the low dimensional embedding vectors, $\mathbf{y}_i$, of dimension d by using the reconstruction weights, w_{ij}, of the high dimensional data vectors, $\mathbf{x}_i$. The only information used in this portion of the algorithm is the geometry obtained by the weights. Here, it is necessary to minimize a cost function for the errors between the reconstruction weights and the outputs, $\mathbf{y}_i$ as follows:

$$\phi(Y) = \sum_{i=1}^{p} \|\mathbf{y}_i - \sum_{j=1}^{p} w_{ij}\mathbf{y}_j\|^2.$$

Y denotes the $d \times p$ matrix where the columns are the embedding vectors. In order to find these reconstruction vectors, $\mathbf{y}_i$, the following optimization problem must be solved for fixed weights, w_{ij}.

Using linear algebra, we can manipulate our cost function as follows:

$$\begin{aligned}
\phi(Y) &= \sum_{i=1}^{p} \|\mathbf{y}_i - \sum_{j=1}^{p} w_{ij}\mathbf{y}_j\|^2 \\
&= \sum_{i=1}^{p} \left\langle \mathbf{y}_i - \sum_{j=1}^{p} w_{ij}\mathbf{y}_j, \mathbf{y}_i - \sum_{j=1}^{p} w_{ij}\mathbf{y}_j \right\rangle \\
&= \sum_{i=1}^{p} \langle \mathbf{y}_i, \mathbf{y}_i \rangle - \sum_{i=1}^{p} \left\langle \mathbf{y}_i, \sum_{j=1}^{p} w_{ij}\mathbf{y}_j \right\rangle - \sum_{i=1}^{p} \left\langle \sum_{j=1}^{p} w_{ij}\mathbf{y}_j, \mathbf{y}_i \right\rangle + \sum_{i=1}^{p} \left\langle \sum_{j=1}^{p} w_{ij}\mathbf{y}_j, \sum_{j=1}^{p} w_{ij}\mathbf{y}_j \right\rangle \\
&= \sum_{i=1}^{p} \left(\langle \mathbf{y}_i, \mathbf{y}_i \rangle - 2\sum_{j=1}^{p} w_{ij} \langle \mathbf{y}_i, \mathbf{y}_j \rangle + \sum_{j,k=1}^{p} w_{ij}w_{ik} \langle \mathbf{y}_j, \mathbf{y}_k \rangle \right) \\
&= \sum_{i,j=1}^{p} \langle \mathbf{y}_i, \mathbf{y}_j \rangle \left(\delta_{ij} - 2w_{ij} + \sum_{k=1}^{p} w_{ij}w_{ik} \right)
\end{aligned}$$

where $< \cdot, \cdot >$ is the standard inner product. Note that $\sum_{i,j=1}^{p} \langle \mathbf{y}_i, \mathbf{y}_j \rangle w_{ij} = \sum_{i,j=1}^{p} \langle \mathbf{y}_j, \mathbf{y}_i \rangle w_{ji}$ by basic properties of inner products. Thus,

$$\begin{aligned}\phi(Y) &= \sum_{i,j=1}^{p} \langle \mathbf{y}_i, \mathbf{y}_j \rangle \left(\delta_{ij} - 2w_{ij} + \sum_{k=1}^{p} w_{ij} w_{ik} \right) \\ &= \sum_{i,j=1}^{p} M_{ij} \langle \mathbf{y}_i, \mathbf{y}_j \rangle\end{aligned}$$

where $M_{ij} = \delta_{ij} - w_{ij} - w_{ji} + \sum_{k=1}^{p} w_{ij} w_{ik}$ where all of the w_{ij} come from the weight matrix W. Thus,

$$M = I - W - W^T + W^T W = (I - W)^T (I - W).$$

We see that M is symmetric even though w_{ij} is not necessarily equal to w_{ji}. This fact will become important shortly. Also, M is extremely sparse as W is, and it is also semi-positive definite.

Now we want to show that $\phi(Y) = \sum_{i,j=1}^{p} M_{ij} \langle \mathbf{y}_i, \mathbf{y}_j \rangle = tr(YMY^T)$. Using the fact that the trace function is commutative, we see that

$$\begin{aligned}tr(YMY^T) &= tr(MYY^T) \\ &= tr\left(\begin{bmatrix} m_{11} & \cdots & m_{1p} \\ \vdots & \ddots & \vdots \\ m_{p1} & \cdots & m_{pp} \end{bmatrix} \begin{bmatrix} \|\mathbf{y}_1\|^2 & \mathbf{y}_1^T \mathbf{y}_2 \cdots & \mathbf{y}_1^T \mathbf{y}_p \\ \vdots & \ddots & \vdots \\ \mathbf{y}_p^T \mathbf{y}_1 & \mathbf{y}_p^T \mathbf{y}_2 \cdots & \|\mathbf{y}_p\|^2 \end{bmatrix} \right) \\ &= tr\left(\begin{bmatrix} m_{11}\|\mathbf{y}_1\|^2 + \cdots + m_{1p}\mathbf{y}_p^T\mathbf{y}_1 & \cdots & m_{11}\mathbf{y}_1^T\mathbf{y}_p + \cdots + \|\mathbf{y}_p\|^2 \\ \vdots & \ddots & \vdots \\ m_{p1}\|\mathbf{y}_1\|^2 + \cdots + m_{p1}\mathbf{y}_p^T\mathbf{y}_1 & \cdots & m_{p1}\mathbf{y}_1^T\mathbf{y}_p + \cdots + m_{pp}\|\mathbf{y}_p\|^2 \end{bmatrix} \right) \\ &= m_{11}\|\mathbf{y}_1\|^2 + \cdots + m_{1p}\mathbf{y}_p^T\mathbf{y}_1 + m_{12}\mathbf{y}_1^T\mathbf{y}_2 + \cdots + m_{p2}\mathbf{y}_p^T\mathbf{y}_2 + \cdots\end{aligned}$$

$$\cdots + m_{p1}\mathbf{y}_1^T\mathbf{y}_p + \cdots + m_{pp}\|\mathbf{y}_p\|^2$$

Recall now that M is symmetric, so $m_{ij} = m_{ji}$. This then yields

$$tr(YMY^T) = \sum_{i,j=1}^{p} M_{ij} \left\langle \mathbf{y}_i, \mathbf{y}_j \right\rangle$$

as desired.

Our problem then becomes

$$\min_Y tr(YMY^T)$$

$$\text{subject to } YY^T = I$$

where the constraint is equivalent to saying that the embedding vectors are sphered or whitened. Thus, they are uncorrelated, and their variances equal unity. Note that Y is orthogonal in the row space but the embedding vectors are not required to be orthogonal. This constraint does not change the problem due to the invariance under rotations and rescalings as discussed previously. Otherwise, letting $\mathbf{y}_i = 0$ for each i would be the optimal solution. We also use the fact that translations do not affect the cost function, so we require the outputs to be centered at the origin, adding the constraint:

$$\sum_{i=1}^{p} \mathbf{y}_i = \mathbf{0}$$

We will again use Lagrange multipliers to solve this problem. Our Lagrangian becomes

$$L(Y, \mu) = tr(YMY^T) - \sum_{i,j=1}^{d} \mu_{ij}(YY^T - I)_{ij}$$

where each μ_{ij} is the Lagrange multiplier for each constraint. Taking the derivative of this Lagrangian with respect to the matrix Y and equating to zero will yield our desired solution. The derivative of $tr(YMY^T)$ with respect to Y is a well-known identity:

$$\frac{\partial}{\partial Y}\left(tr(YMY^T)\right) = Y\left(M + M^T\right)$$

The derivative of the scalars $\alpha = \sum_{i,j=1}^{d} \mu_{ij}(YY^T - I)_{ij}$ with respect to the matrix Y can be computed as follows:

$$\begin{aligned}
\frac{\partial \alpha}{\partial Y} &= \sum_{i,j=1}^{d} \mu_{ij} \begin{bmatrix} \frac{\partial (YY^T - I)_{ij}}{\partial y_{11}} & \cdots & \frac{\partial (YY^T - I)_{ij}}{\partial y_{1p}} \\ \vdots & \ddots & \vdots \\ \frac{\partial (YY^T - I)_{ij}}{\partial y_{d1}} & \cdots & \frac{\partial (YY^T - I)_{ij}}{\partial y_{dp}} \end{bmatrix} \\
&= \begin{bmatrix} \sum_{h=1}^{d} (\mu_{ih} + \mu_{hi}) y_{h1} & \cdots & \sum_{h=1}^{d} (\mu_{ih} + \mu_{hi}) y_{hp} \\ \vdots & \ddots & \vdots \\ \sum_{h=1}^{d} (\mu_{dh} + \mu_{hd}) y_{h1} & \cdots & \sum_{h=1}^{d} (\mu_{dh} + \mu_{hd}) y_{hp} \end{bmatrix}
\end{aligned}$$

Setting these two derivatives equal to each other, we see that

$$(2YM)_{ij} = 2\sum_{k=1}^{p} y_{ik} m_{kj} = \sum_{h=1}^{d} (\mu_{ih} + \mu_{hi}) y_{hj} = (\frac{\partial \alpha}{Y})_{ij}.$$

Then equating coefficients on both sides, we see that the $d \times d$ diagonal matrix N of the form

$$N_{ij} = \begin{cases} 0 & \text{if } i \neq j \\ \mu_{ij} + \mu_{ij} & \text{if } i = j \end{cases}$$

satisfies this equation. Thus, if we let $2\Lambda = N$, we see

$$\begin{aligned} 2YM &= 2\Lambda Y \\ \implies (YM)^T &= (\Lambda Y)^T \\ \implies MY^T &= Y^T\Lambda. \end{aligned}$$

Thus, Y^T is the matrix of eigenvectors of M, and Λ is the corresponding diagonal matrix of eigenvalues. The optimal embedding up to rotations, translations, and rescalings of the embedding space can then be found by solving this eigenvector problem. The Rayleigh-Ritz theorem as described in [47] gives an indication of which eigenvectors actually solve the problem. Using this, we need to obtain the bottom $d+1$ eigenvectors of the matrix, M, (those eigenvectors corresponding to the smallest eigenvalues in increasing order). We will see shortly that the eigenvector corresponding to the smallest eigenvalue is the unit vector with all equal components corresponding to the mean of the data. We discard this eigenvector, leaving the second through the $d+1$ eigenvectors. Thus, the embedding vectors that solve the LLE algorithm are these d remaining eigenvectors. When discarding the bottom eigenvector, it forces each of the other eigenvectors to sum to zero by orthogonality enforcing the constraint

$$\sum_{i=1}^{p} \mathbf{y}_i = \mathbf{0}$$

which requires that the embedding vectors to be centered around the origin.

To see that there exists a unit eigenvector with all equal components as described above, we need to verify that this in fact is a solution. Assume that $\mathbf{u} = \alpha\mathbf{e}$ where $\mathbf{e}$ is the vector

of all ones and α is a scalar. If $\mathbf{u}$ is an eigenvector, then

$$\begin{aligned} M\mathbf{u} &= \lambda\mathbf{u} \\ \implies M\alpha\mathbf{e} &= \lambda\alpha\mathbf{e} \\ \implies M\mathbf{e} &= \lambda\mathbf{e} \end{aligned}$$

which implies that

$$\sum_{j=1}^{p} M_{ij} = \lambda$$

for each row i. This indicates that in order for there to exist a unit eigenvector with all equal components, each row of the M matrix must sum to a scalar, λ. In fact, we can show that each row has zero sum. Given

$$M_{ij} = \delta_{ij} - w_{ij} - w_{ji} + \sum_{k=1}^{p} w_{ki}w_{kj}$$

we can see

$$\begin{aligned} \sum_{j=1}^{p} M_{ij} &= \sum_{j=1}^{p} \delta_{ij} - \sum_{j=1}^{p} w_{ij} - \sum_{j=1}^{p} w_{ji} + \sum_{j=1}^{p}\sum_{k=1}^{p} w_{ki}w_{kj} \\ &= 1 - 1 - \sum_{j=1}^{p} w_{ji} + \sum_{j=1}^{p}\sum_{k=1}^{p} w_{ki}w_{kj} \\ &= 0 - \sum_{j=1}^{p} w_{ji} + \sum_{k=1}^{p} w_{ki} \qquad \text{since } \sum_{j=1}^{p} w_{kj} = 1 \\ &= 0 \end{aligned}$$

Thus, there does exist a unit eigenvector of all equal components corresponding to the eigenvalue zero which we may discard as described above.

The third step of the LLE algorithm involves solving this eigenvector problem to determine the unique solution Y. The solutions to the problem are the d-dimensional columns of the Y matrix where each column, j, of Y corresponds to column, j, of X, and each row of Y is an eigenvector of the matrix M. Note that although each weight was determined locally by reconstructing a data point by its nearest neighbors, the optimal embedding Y was determined by a $p \times p$ eigensolver which is a global undertaking that uses the information from all points. Therefore, through the LLE algorithm we were able to obtain low-dimensional embedding vectors that preserve the local topology of a high-dimensional data set by determining a global coordinate system.

2.2.4. Other Considerations. In this section, we will describe and prove, in more detail, particular considerations drawn from the derivation of LLE. We will prove the invariance relations of the weights, give an approach to choosing an appropriate number of nearest neighbors, discuss the regularization necessary when the number of neighbors K is greater than D the dimension of the original data set, and discuss discovering and enforcing the inherent dimensionality d of the underlying manifold.

2.2.4.1. *Invariance.* In this section, we will discuss some invariance relations of the weights. The weights that minimize the reconstruction errors are invariant to the isometries of rotations, rescalings, and translations of the data points. The proofs of these results follow.

Rescalings: The general form of the errors ensures that the weights are invariant to rescalings. Let $\alpha \in \mathbb{R}$ represent a scaling factor. Then

$$\begin{aligned}
\epsilon(W) &= \sum_{i=1}^{p} \|\alpha \mathbf{x}_i - \sum_{j \in N_i} w_{ij}(\alpha \mathbf{x}_j)\|^2 \\
&= \sum_{i=1}^{p} \|\alpha \mathbf{x}_i - \alpha \sum_{j \in N_i} w_{ij} \mathbf{x}_j\|^2 \\
&= \sum_{i=1}^{p} \alpha \|\mathbf{x}_i - \sum_{j \in N_i} w_{ij} \mathbf{x}_j\|^2 \\
&= \alpha \sum_{i=1}^{p} \|\mathbf{x}_i - \sum_{j \in N_i} w_{ij} \mathbf{x}_j\|^2
\end{aligned}$$

Thus, the weights that minimize the above cost function will clearly minimize the weights in the original cost function for the reconstruction errors

$$\epsilon(W) = \sum_{i=1}^{p} \|\mathbf{x}_i - \sum_{j \in N_i} w_{ij} \mathbf{x}_j\|^2$$

Rotations: Recall that if Q is an orthonormal matrix, then its transpose is equal to its inverse. If Q is orthonomal then an immediate consequence is that for any $\mathbf{x}$

$$\|Q\mathbf{x}\| = \|\mathbf{x}\|.$$

This can be seen by noting

$$\|Q\mathbf{x}\|^2 = (Q\mathbf{x})^T(Q\mathbf{x}) = \mathbf{x}^T Q^T Q \mathbf{x} = \mathbf{x}^T I \mathbf{x} = \mathbf{x}^T \mathbf{x} = \|\mathbf{x}\|^2$$

Now, using the fact that rotation matrices are orthogonal, we see that

$$\begin{aligned}
\epsilon(W) &= \sum_{i=1}^{p} \|Q\mathbf{x}_i - \sum_{j \in N_i} w_{ij} Q\mathbf{x}_j\|^2 \\
&= \sum_{i=1}^{p} \|Q(\mathbf{x}_i - \sum_{j \in N_i} w_{ij}\mathbf{x}_j)\|^2 \\
&= \sum_{i=1}^{p} \|\mathbf{x}_i - \sum_{j \in N_i} w_{ij}\mathbf{x}_j\|^2
\end{aligned}$$

as desired.

Translations: Using the constraint, $\sum_{j \in N} w_{ij} = 1$, we ensure that these weights are invariant to translations. Let $\alpha\mathbf{e}_k$ represent a translation vector where α is a scalar and $\mathbf{e}_k$ is the elementary column vector with one in the k^{th} position and zeros everywhere else. Then, we can see that the weights are invariant to translations as follows

$$\begin{aligned}
\epsilon(W) &= \sum_{i=1}^{p} \|(\mathbf{x}_i + \alpha\mathbf{e}_k) - \sum_{j \in N_i} w_{ij}(\mathbf{x}_j + \alpha\mathbf{e}_k)\|^2 \\
&= \sum_{i=1}^{p} \|\mathbf{x}_i + \alpha\mathbf{e}_k - \sum_{j \in N_i} w_{ij}\mathbf{x}_j + \sum_{j \in N_i} \mathbf{w}_{ij}\alpha\mathbf{e}_k\|^2 \\
&= \sum_{i=1}^{p} \|\mathbf{x}_i + \alpha\mathbf{e}_k - \sum_{j \in N_i} w_{ij}\mathbf{x}_j - \alpha\mathbf{e}_k\|^2 \\
&= \sum_{i=1}^{p} \|\mathbf{x}_i - \sum_{j \in N_i} w_{ij}\mathbf{x}_j\|^2
\end{aligned}$$

2.2.4.2. *Choosing an Appropriate Value for K.* Note that [71] did not give guidance on how to choose an appropriate number of nearest neighbors. It is suggested in [57] that it is possible to check every possible value of K nearest neighbors up to some maximum value in every step of the algorithm and determine the optimal value by finding the K whose

associated embedding space best reconstructs the high-dimensional structure. This "quality" measure is determined by the residual variance, defined as $1 - \rho^2_{D_x D_y}$.

Here, ρ is the linear correlation coefficient taken over the $p \times p$ matrices of Euclidean distances, D_x and D_y, between pairs of points in X, a $D \times p$ dimensional matrix of the high dimensional data, and Y, a $d \times p$ dimensional matrix of the low dimensional embedding vectors where p indicates the number of data points. A linear correlation coefficient measures the strength and direction of a linear relationship between two variables. It lies between -1 and 1 with values of ρ close to 0 indicating a weak linear correlation between the data points and values of ρ close to 1 in magnitude indicating a strong linear correlation. Here a positive ρ value indicates a direct relationship between X and Y and a negative ρ value indicates an indirect relationship [46].

Methodically choosing each value of K and running the full LLE algorithm in order to determine the optimal K is straightforward but computationally intensive. Thus, a hierarchical approach is proposed in [57]. This method suggests instead of running the full LLE algorithm, the first step to determine K nearest neighbors and then the second step to determine the weights associated to these nearest neighbors should be computed for $K \in [1, K_{max}]$ where K_{max} has been designated. The reconstruction error between the original high dimensional data and the linear combination of its nearest neighbors, $\epsilon(W)$ (Equation 2.1), should be calculated for each value of K. Then for every value of K that minimizes this error, the third step to determine the embedding vectors is performed in order to calculate the residual variance. The optimal K is selected as the one that minimizes the residual variance $1 - \rho^2_{D_x D_y}$.

This hierarchical method is computationally less expensive than the straightforward method discussed as the eigenvector problem–which is the most expensive computation of the algorithm–is only performed for a small number of K values. Experiments done by [57] indicate that the hierarchical method for finding this optimal value of K are quite precise. However, [3] indicates that the residual variance is not a good measure of the local geometric structure of the data.

A similar global approach as [57], is proposed in [3]. However, instead of using the minimum residual variance to determine the optimal K value, a new measure called Preservation Neighborhood Error–which incorporates both the global manifold behavior and the local geometry of the data–is used. Also in this paper, [3] indicates a method to select a K_i for each point $\mathbf{x}_i$ locally based on graph theory.

Other work by [52] indicates that an optimal K need not be selected as often a range of K values produce stable reconstructions. This claim is dependent on sampling density and manifold geometry. We have not seen this to be the case for many of the data sets considered in our work, however.

In Chapter 5, we will explore a new method to automatically determine an appropriate choice of nearest neighbors for each data point by using sparsity of numerical results in a modified optimization problem to determine the weights. We call this method Sparse Locally Linear Embedding.

2.2.4.3. *Regularization.* In the situation where the original dimension of the data, D, is fairly low, it is often necessary to choose the number of neighbors, K, to be greater than this dimension to avoid the eigenvector problem becoming singular. If $K > D$, then the set of nearest neighbors, N_i, of data point x_i is no longer linearly independent, and thus,

there is not a unique solution for determining the reconstruction weights. In this case, the covariance matrix C_i defined above in Equation 2.2 becomes singular or nearly singular. A regularization must be implemented in order to suspend this breaking down of the algorithm.

One such regularization would be to add a small multiple of the identity to the covariance matrix which, in turn, corrects the sum of the squares of the weights so that the weights favor a uniform distribution. The optimization problem then finds the set of weights that come closest to the point representing uniform distribution of magnitude for each of the weights [71]. In this paper, the regularization that is used is

$$C_i \leftarrow C_i + I * tol * tr(C_i)$$

where tol is a tolerance that is sufficiently small, usually 0.001, and $tr(C_i)$ denotes the trace of C_i. This regularizer is sufficient to make the covariance matrix well-conditioned allowing one to determine a unique solution to the optimization problem to determine the weights. Other regularization methods are proposed in [53] and [28], but we will not focus on these here.

2.2.4.4. *Discussion of the Lower Dimensionality d.* It is common to choose the dimension of the embedding vectors, d, to be 2 or 3 due to the fact that such an embedding space is the easiest to visualize. However, it is desirable to obtain information about the instrinsic dimensionality, d, of the high dimensional manifold. There have been approaches proposed to do this by [66]. However, it is indicated by [71] that traditional methods to estimate the inherent dimensionality d of the data set such as Principal Component Analysis (PCA) or box counting can be used to estimate this number as well. If information about the manifold's intrinsic dimensionality is known a priori or if it is desired to embed the manifold in a

particular dimension, then some modifications in the second step of LLE can be implemented to force the dimensionality d. In [71], it is indicated that this modification may be done by projecting each data point of the higher dimensional data and its nearest neighbors into its inherent d-dimensional subspace before implementing the least squares reconstruction. The global lower dimensional coordinate system is determined by computing the d bottom eigenvectors of the covariance matrix C. The projection decreases the rank of the matrix C, and the weights are computed as described above.

2.2.5. An Example. This section consists of an example illustrating the LLE algorithm's topology preserving capabilities. The data set consists of images of a black square translated over a background of random noise. The black square is allowed to split and "wrap" around each boundary edge of the background. To construct the data set, we start with a 20×20 matrix consisting of random entries between 0 and 1. Within this matrix, a 10×10 zero matrix is superimposed. The data set is generated by considering all positions of the 10×10 matrix inside the 20×20 matrix of random noise (allowing both horizontal and vertical wrapping). This generates 400 points of dimension 400. Sample images are displayed in Figure 2.2. Thus from a topological point of view, the data set corresponds to a noisy sampling of a torus in $\mathbb{R}^{400}$. We defined the nearest neighbors, of each element in the data set, to be the 4 nearest data points in $\mathbb{R}^{400}$. The LLE algorithm was then used to map the data to $\mathbb{R}^3$. The resulting embedded data in $\mathbb{R}^3$ is displayed in Figure 2.3 and reflects the original topological structure quite clearly. Note that colors of the 400 dimensional images correspond to points in the reduced space.

In general, results from experiments suggest that LLE can indeed be successful in its goal of nonlinear dimensionality reduction that captures inherent topological properties through

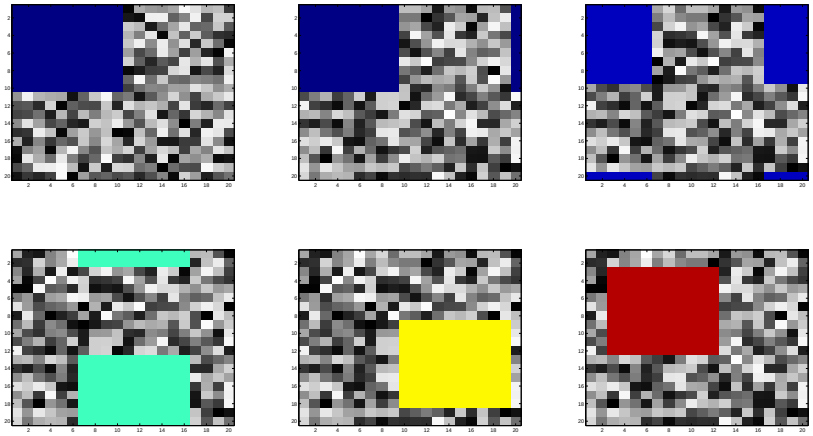


FIGURE 2.2. Sample points from the data set of 400 points in $\mathbb{R}^{400}$ created by traversing and wrapping a 10×10 black square in a 20×20 random square.

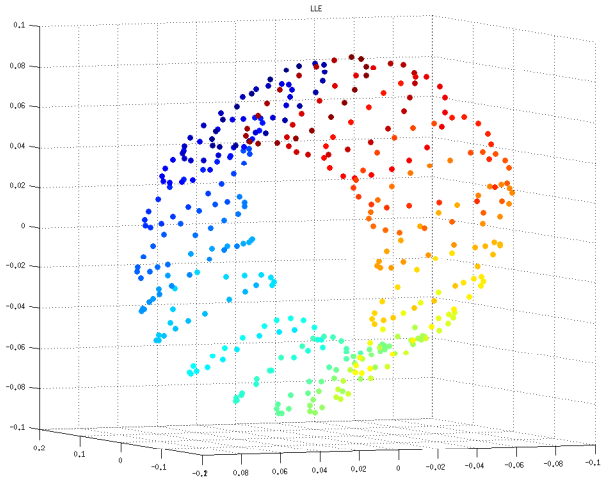


FIGURE 2.3. A plot of the embedding vectors obtained by LLE of 400 points in $\mathbb{R}^{400}$ reduced down to $\mathbb{R}^3$.

a single, linear algebra derived, map. Note that linear methods such as PCA would not uncover this nonlinear structure of the data.

2.3. METRICS

Distance can be thought of as a scalar measure of similarity for two signals. The way in which the distance between two objects is computed can produce completely different

results. Recall in the LBG algorithm discussed in Section 2.1, many distance computations are performed, i.e. the distance between each point and the centers, with assignment given to the center with the smallest distance. As we discussed there, typically Euclidean distance is used. However, varying the choice of metric will vary the Voronoi sets after each iteration, and thus, the final clustering will also vary. In this section, we briefly discuss 5 different choices of distance measure.

The first metric to be considered is the standard Euclidean distance (or 2-norm) defined as

$$||\mathbf{x}-\mathbf{y}||_2 = \sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\ldots+(x_n-y_n)^2}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$. This is the most intuitive distance measure, i.e. the distance between two points in n-dimensional space as if measured by a ruler.

Next, we will consider the taxicab metric (or 1-norm) metric defined as

$$||\mathbf{x}-\mathbf{y}||_1 = \sum_{i=1}^{n} |x_i - y_i|.$$

This distance metric is called the taxicab metric as it measures the distance between two points as a car would drive through a city of square blocks from one point to the other.

A third metric of consideration is the Chebyshev distance (or ∞-norm) defined as

$$||\mathbf{x}-\mathbf{y}||_\infty = \max_{\mathbf{i}} |x_i - y_i|,$$

the largest coordinate difference in any dimension between two points. These first three metrics are referred to as ℓ_p norms.

Now, we consider the Mahalanobis distance, related to the Euclidean distance but distinct in that it incorporates correlations between the data. The definition is as follows

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T C^{-1} (\mathbf{x} - \mathbf{y})}$$

where C is the covariance matrix. We define our data matrix X as the $n \times p$ matrix with p objects of size n, $\overline{X}$ as the $n \times p$ matrix with columns all equal to the mean of X, and X_c as the mean centered data matrix $X_c = (X - \overline{X})$. Then, the covariance matrix is defined as

$$C = \frac{1}{p-1} X_c \cdot X_c^T.$$

Note when the covariance matrix is the identity (i.e. the data is uncorrelated and has unit variance), the Mahalanobis distance is analogous to the Euclidean distance.

In Figure 2.4, we see an illustration of the unit discs of the four metric choices described above with the circle representing Euclidean, the diamond representing Taxicab, the square representing Chebyshev, and the ellipse representing Mahalanobis.

Finally, we consider the spectral angle distance defined as the smallest angle between two vectors,

$$\theta(\mathbf{x}, \mathbf{y}) = \arccos \frac{< \mathbf{x}, \mathbf{y} >}{||\mathbf{x}||||\mathbf{y}||}$$

where $< \cdot, \cdot >$ is the standard inner product and $|| \cdot ||$ is the magnitude measured using the 2-norm and $0 \leq \theta \leq \frac{\pi}{2}$. Note that this distance measure is invariant to scalar multiplication,

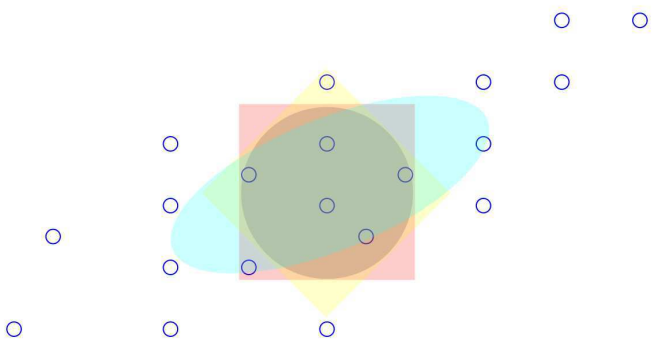


FIGURE 2.4. Illustration of metric choices with the circle representing Euclidean, the diamond representing Taxicab, the square representing Chebyshev, and the ellipse representing Mahalanobis.

e.g. for scalars a, b,

$$\begin{aligned}\theta(a\mathbf{x}, b\mathbf{y}) &= \arccos \frac{(< a\mathbf{x}, b\mathbf{y} >)}{||a\mathbf{x}||||b\mathbf{y}||} \\ &= \arccos \frac{ab(< \mathbf{x}, \mathbf{y} >)}{||\mathbf{x}||||\mathbf{y}||} \\ &= \theta(\mathbf{x}, \mathbf{y})\end{aligned}$$

Note, however, that this is not true of the other metrics defined above. Therefore, spectral angle is less sensitive to illumination conditions. The spectral angle distance is a pseudometric as the distance between two distinct points can equal zero.

In Chapter 3, we will use each of these choices of metrics to analyze a variety of images and compare the results. We will also consider adding ℓ_1 and ℓ_2 regularization terms to objective functions of optimization problems to induce sparsity, positivity, and uniformity in decision variables in Chapters 5 and 7.

CHAPTER 3

Biological Landscape Organizer and Semi-supervised Segmenting Machine (BLOSSM)

3.1. Introduction

In this chapter, we discuss an application of the well-known Linde-Buzo-Gray clustering algorithm discussed in Section 2.1 to landscape ecology. We introduce a friendly graphical user interface dubbed BLOSSM (Biological Landscape Organizer and Semi-Supervised Segmenting Machine) that we have developed. BLOSSM quantizes the pixels of an image and thus, can reveal the structure of the color space of an image. We consider this algorithm's performance using four choices of color space (RGB, Quantized RGB, Named Color, and CIELab) and five similarity measures (ℓ_1, ℓ_2, ℓ_∞, Mahalanobis distance, and spectral angle). We analyze a sample image and reveal important features pertaining to the clustering.

3.2. Motivation: An Ecology Application

A longer growing season is one of the most well-documented biological consequences of Earth's warming climate. Earlier dates for leaf-opening and flowering for many plants, including garden plants, native trees and wildflowers, have been observed around the world during the past 30 years. In contrast, a new evaluation of recent research indicates that many plant species are shortening their annual growth cycle in response to climate warming. Plants that are greening and flowering earlier are also ending growth earlier in the year. A

shortened period of growth may be a result of current environmental conditions, such as a mid-season drought in a warm summer, or could be the result of years of conditioning to past environmental conditions. Many species cannot extend the period over which they grow and flower beyond a limited number of days [21, 38, 58, 67, 68].

Surprisingly, these observations of the annual growth cycle of individual species are not inconsistent with a longer growing season observed, for example, in satellite images of Earth's green season. Earlier springs may occur because early-season species are greening and flowering earlier due to warmer spring air temperatures. Later falls may result from a delay in the growth of other, late-season species in order to avoid extreme mid-season temperatures and summer droughts. The longer growing season can be the result of opposing responses by individual species within a common locale or community.

Consider a meadow with two plants, which in the past grew and flowered over the same period of time. As these plants shift the timing of when they grow and flower in different ways, the period of time in which they are both flowering will decrease, and eventually there may be a gap between when they flower. The rate of decrease will be much more rapid and lead to gaps sooner if the plants are also shortening their annual growth cycles as has been observed in recent climate change experiments. Indeed, the co-flowering patterns of some subalpine plant species pairs has been decreasing, potentially as a result of earlier snowmelt and earlier flowering by some species.

As time gaps open up when different species are growing and flowering, there will be visible changes to natural landscapes. For example, a Rocky Mountain meadow that at peak season is filled with diverse colors of flowers may instead flower in waves of reduced color variation. The period of peak flowering may decrease, potentially affecting the economy of

regions dependent on ecotourism. Other consequences may be changes in plant-pollinator interactions and the establishment of invasive species that can take advantage of the new niche created by the temporal gaps in native plant flowering patterns.

New approaches to quantify changes in flowering patterns will be essential to determine plant community responses to global environmental changes, especially climate change. Here, we present a mathematical approach for quantifying color space in images of Rocky Mountain subalpine meadows to determine the timing and duration of flowering events, including patterns of co-flowering. This approach considers each pixel in an image represented by its RGB color and divides the set of all pixels into groups according to similarity in the color space. All pixels in each group are then represented by a prototype of that group, allowing for a much coarser representation and a quantization of the information in the image.

Investigating natural imagery can provide a quantitative measure of changes in flowering patterns. In a high resolution digital image, pixels can be used to identify and characterize various types of flowers. For instance, in an image of a landscape such as Figure 3.1, we can quantify each type of flower by identifying the number of pixels that are yellow, blue, or white. This can be used to ascertain the flower cover of a particular type of flower within a region by identifying the number of pixels colored yellow, for instance, and determining its percentage of the entire area–the total number of pixels–within an image. Once a set of images is quantified to show the spatial relationships and patterns over a period of time, response to global environmental changes can be understood.

3.3. Overview

If the color information of a landscape image can be classified in some way, then the ground cover can be quantified. One approach to do this is using color quantization. Color

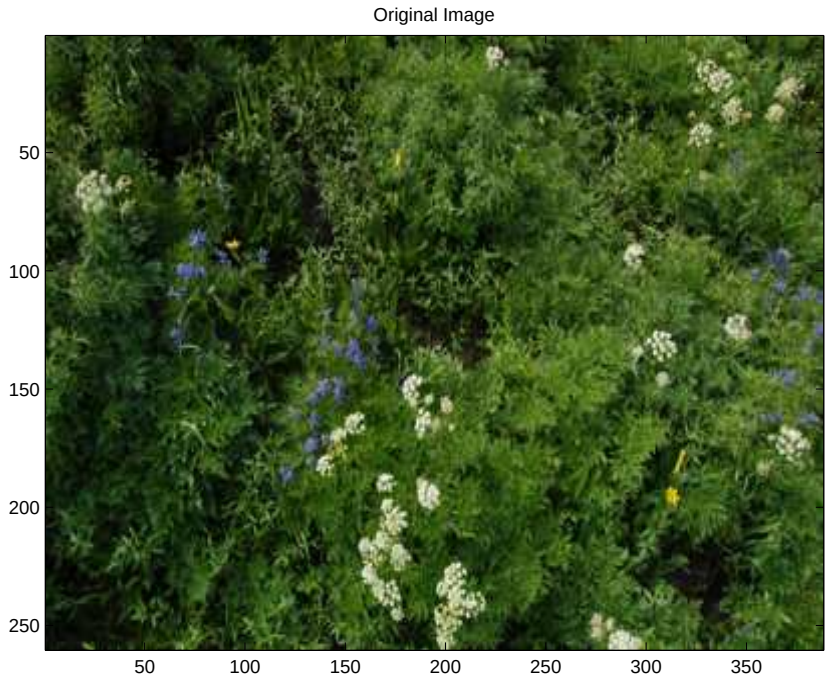


FIGURE 3.1. Sample image of landscape data set.

quantization reduces the number of colors used in an image. Much work has been done to quantize the color space of natural imagery. The wide variety of approaches include statistical-based, graph theoretical, clustering, gradient descent, among many other techniques [23, 24, 30, 44, 48, 59, 60, 61, 62, 65, 80].

We will focus on clustering. Clustering is an effective tool used in data mining, the process of deducing patterns from data, in which groups of objects, or clusters, are formed such that objects within a cluster are similar, and objects in different clusters are quite distinct. Thus, the within class variance of a cluster is small, and the between class variance is large [43]. Using clustering, the color content of an image can be quantized at a much coarser level (while maintaining most of the visual information) by allowing each pixel's color to be identified with a prototype as determined by a quantization algorithm.

Many algorithms have been developed with the end goal of clustering data. See [49] for a nice review of clustering algorithms. Here, we will focus on the well-studied Linde-Buzo-Gray (LBG) algorithm discussed in Section 2.1 [54].

Each pixel in an image is assigned a numeric value representing a distinct color. Do note however, if the resolution of the image is not fine enough, these pixel values may actually combine subpixel colors representing multiple colors from the natural landscape. The color space of an image has a dimension of at most $A \cdot B$ if the resolution of the image is $A \times B$. The LBG algorithm used to quantize the color space of an image greatly reduces this dimension. The number of centers that the data is clustered around determines the maximum dimension of the reduced color space. It is important to note that for a particular image and a particular set of centers some centers could potentially be unused, and thus, the number of centers is an upper bound for the dimension of the reconstruction color space.

As we will see in Section 3.6, the way in which data is represented and how similarity is measured between two objects can drastically change the outcome of the LBG clustering algorithm. We will consider a variety of color spaces in which to analyze our data as well as a variety of metrics with which to measure similarity. In Section 2.3, we discussed five different choices of distance measures that will be used throughout this chapter. The next section briefly describes each of the choices of color space.

3.4. Color Spaces

The choice of how to represent data can often affect algorithm performance on the data. We will be exploring the effect of representing image (i.e. color) data in 4 different color spaces: RGB, Quantized RGB, Named Color, and CIELab. We present a brief overview of each color space in the following subsections as well as the typical set-up of the data.

3.4.1. Red, Green, Blue. An object under a fixed illumination condition, as perceived by the human eye, is often represented by considering a particular map to $\mathbb{R}^3$ obtained by integrating, at each small region of an object, the product of the spectral reflectance curve against three particular frequency response curves. We refer to these three functionals as maps to red, green, and blue (RGB) space [83]. As a result of this map, a digital photograph typically represents a given object/illumination pair as an $A \times B \times 3$ data array where the first two coordinates record the location in the image and the last coordinate records the values of the red, green, and blue functionals on the associated spectral reflectance curve. By combining the three $A \times B$ color sheets, one can well approximate the human perception of the object/illumination pair, see Figure 3.2.

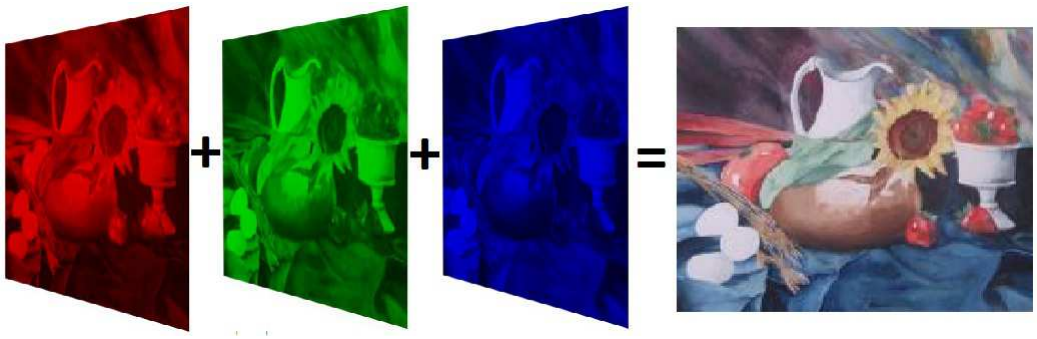


Figure 3.2. Illustration of the red, green, and blue sheets associated to pixels of an image.

Reflecting this structure, the input data we will be considering consists of $A \times B \times 3$ arrays corresponding to digital pictures of natural imagery. The entries in the three $A \times B$ sheets correspond to the energy near the red, green, or blue frequencies at each pixel. We have chosen to process our images using MATLAB. Each color component of a pixel is an integral value between 0 and 255 (corresponding to an eight bit representation). Each pixel is associated to a point in $\mathbb{R}^3$ representing the (RGB) color of the pixel. As there are three

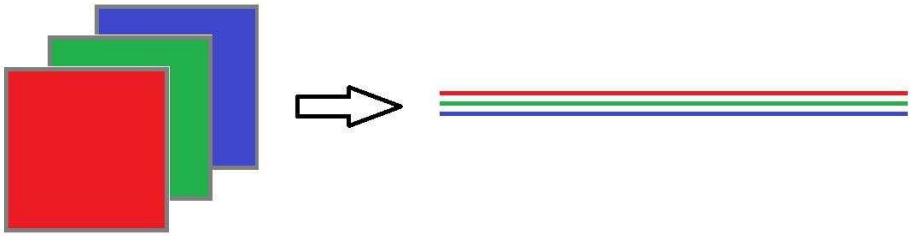


FIGURE 3.3. Illustration of creating a data matrix from RGB data.

components of each color with 256 possible choices each, there are $256^3 = 16,777,216$ distinct colors that can be represented.

In our analysis, each sheet pertaining to the RGB components of an image was converted from a matrix of dimension equal to the resolution of each image to a long column vector of dimension $1 \times p$ where p is the number of pixels in the image. A new matrix, X, of dimension $3 \times p$ was created to contain all of the data entries of these long row vectors. By organizing the data in this way, we see that each column of the matrix corresponds to the RGB components of an individual pixel within an image, see Figure 3.3 for an illustration.

3.4.2. QUANTIZED RGB. The choice of representing color information in 8 bits quantizes the continuous functionals of the red, green, and blue maps to 256 possibilities for each color sheet. We further quantize this representation to an even coarser level, representing color information in only 5 bits with $2^5 = 32$ possibilities and thus, only $32^3 = 32768$ distinct colors. The 32 values are in increments of 8 starting with 3.5 and ending with 251.5 for each sheet of color information. This defines what we call the quantized RGB space. The reason for this further quantization will become apparent in the next subsection.

3.4.3. NAMED COLOR. At a very young age, humans are taught to distinguish between colors. Through the psychophysical experiments of Berlin and Kay [12], 11 universal color categories have been designated: white, black, red, green, yellow, blue, brown, purple, pink,

orange, and gray. The Named Color model introduced in [10] and [78] is a fairly new, fuzzy, parametric model based on these 11 universal categories. In this model, any color stimulus is assigned a value between 0 and 1 for each universal color category or 'named color', indicating the percentage of 'named color' present in the stimulus, with a total sum equal to 1. This model directly reflects how color is named by humans. For instance, a sample with values of 0.5 orange and 0.5 red might be called red-orange while a sample with values 0.7 orange and 0.3 red might be called reddish orange.

The Named Color model can be accessed by the look-up table which can be used to easily convert from RGB data to Named Color data [78]. The look-up table provides 3 columns corresponding to the Quantized RGB color space described in the previous subsection, and the 11 remaining columns correspond to the associated Named Color data in $\mathbb{R}^{11}$. Thus, given an RGB stimulus, the closest Quantized RGB point will reveal the associated Named Color data point.

3.4.4. CIELab. The final color model we will discuss is the CIELab model developed in 1976 in order to be more perceptually uniform [84]. Perceptual uniformity means that a change in color (numeric) value should produce a change of similar visual importance, i.e. there should be high correlation between the Euclidean distance between two points and the perceived visual color difference between the color stimuli. The three coordinates (L, a, b) form a Cartesian coordinate space where L represents the achromatic signal of lightness (with 0 indicating black and 100 indicating white), a represents the chromatic channel of red-greenness (negative values indicate green while positive indicate red), and b represents chromatic channel of yellow-blueness (negative values indicate blue while positive indicate yellow). This model, that approximates human vision, exceeds the RGB color space, and

conversion from one model to the next is not straightforward. However, there are built-in MATLAB functions in the Image toolbox to perform this conversion.

3.5. Biological Landscape Organizer and Semi-supervised Segmenting Machine

To aid in analyzing images in a variety of color spaces with a choice of several metrics, we have created an interactive, friendly graphical user interface which we call Biological Landscape Organizer and Semi-Supervised Segmenting Machine (BLOSSM). As the name indicates, BLOSSM is a semi-supervised method primarily designed to analyze landscape images but is applicable to any other type of image as well.

As mentioned in Section 2.1, proper initialization of any iterative algorithm is critical. One key feature of BLOSSM is that it can easily aid in this initialization. BLOSSM allows supervision from the user in identifying starting centers by manually selecting pixels that reflect each of the easily visualized distinct colors within an image as the starting centers. The option to select centers randomly or select the standard 8 RGB colors with entries either 0 or 255 is implemented as well. Identification of centers by the user alleviates many difficulties in initializing the center vectors. A palette of centers can be selected from multiple images and then saved for future use. For instance, see Figure 3.4, for an example of the GUI and starting centers chosen to initialize the algorithm for a sample landscape image.

A choice of color space and choice of metric can be designated in which clustering will be performed. BLOSSM then executes the LBG algorithm using a number of iterations specified by the user (e.g. 15 iterations were performed in the analyses presented in this paper, as the distortion error appeared to level off at this point for an appropriate number and choice of center vectors).

The GUI displays a reconstruction of the original image using the quantization determined by the algorithm. Instead of observing an image with potentially p, distinct colors, the reconstruction displays an image using only the distinct number of (used) centers as colors. This affords an extremely coarse quantization as typically the number of pixels is very large in a high resolution image and the number of centers can be chosen to be small. This algorithm allows for an automatic determination of the predominate colors present within an image and identifies each pixel with its appropriate color.

The user may then learn certain characteristics of the analysis. Visual characteristics are presented: a pie chart displaying the distribution of pixels identified with each color in the reduced color space, swatches of the final centers, images associated to each of the individual clusters created by displaying those pixels identified with a final center. The user of BLOSSM is also able to determine various characteristics pertaining to a particular color such as number of pixels associated with that color, the percent of the total area, the number of contiguous regions greater than a certain number of pixels (to avoid counting noise), and the average number of pixels in each contiguous region. In the case of the landscape images presented in this paper, the contiguous regions can analogously be thought of as flowers or foliage of a certain color. Note that misclassified pixels will be isolated while pixels classified correctly will be adjacent and fall within a contiguous region.

Another feature of the GUI is the ability to combine colors. For instance, there may be a few ending centers that all represent shades of a certain color. These centers (and the pixels identified with each of these centers) can be combined in order to analyze all shades at once. This numeric information is displayed in a table for all colors, including those that have been combined.

The number of contiguous regions of a particular color may be determined by the following recursive algorithm implemented in the GUI. The premise of the algorithm is to first classify all pixels identified together as a certain color, e.g. yellow, by looking at the spectra of the pixels, i.e. the numeric components of the pixel, using the LBG algorithm described above. Then, within each cluster of points identified together, we want to determine how many patches or contiguous regions exist. This can be accomplished by considering the spatial location of pixels within the image via their indices, and determining if pixels are adjacent to one another by index values. The algorithm is as follows:

Determining Contiguous Regions within a Voronoi Set S_i:

(1) Randomly pick a pixel x, at location (i, j) within an image, that is an element of the given Voronoi set, S_i.

(2) Remove x from the Voronoi set.

(3) Consider another pixel, y, remaining in the Voronoi set.

(4) Determine if y is adjacent to pixel x, i.e. if y is in one of the following 8 locations $(i-1, j-1)$, $(i-1, j)$, $(i-1, j+1)$, $(i, j-1)$, $(i, j+1)$, $(i+1, j-1)$, $(i+1, j)$, or $(i+1, j+1)$.

   (a) If so, remove y from the Voronoi set, repeat step 4. recursively until there are no more pixels that are adjacent to y.

   (b) If not, consider another pixel remaining in the Vornoi set.

(5) Repeat step 4. until there is no point remaining in the Voronoi set that is adjacent to point x.

(6) Repeat steps 1-5. until the Voronoi set is empty.

Finally, the entropy–sometimes referred to as the Shannon index–may be computed [45]. This measurement is defined as

$$H' = -\sum_{i=1}^{N} p_i \ln p_i$$

where p_i is the proportion of individuals belonging to the ith class in the dataset of interest and N is the number of classes. Therefore,

$$p_i = \frac{|S_i|}{p}$$

where p is the number of pixels within an image and $|S_i|$ is the number of elements in Voronoi set S_i. The natural log that we have choose to implement will yield entropy in the unit of natural digits or nats. When the proportions of all clusters are relatively equal, the Shannon entropy will be roughly equal to $\ln(N)$. If the proportions of the clusters are quite different, for example one cluster contains most of the data, then the entropy approaches zero. Note that in BLOSSM, entropy may be computed for all of the original Voronoi sets, or colors may first be combined and then entropy may be computed using combined colors as long as all pixels are represented in some cluster. The entropy decreases as clusters are combined since the diversity decreases.

3.6. Analysis

In our analysis, we have chosen a sample landscape image to consider. It is a 2592×3872 (10036224 pixels) natural image of a subalpine meadow near the Rocky Mountain Biological Laboratory in Gothic, Colorado provided by Dr. David Inouye of the University of Maryland.

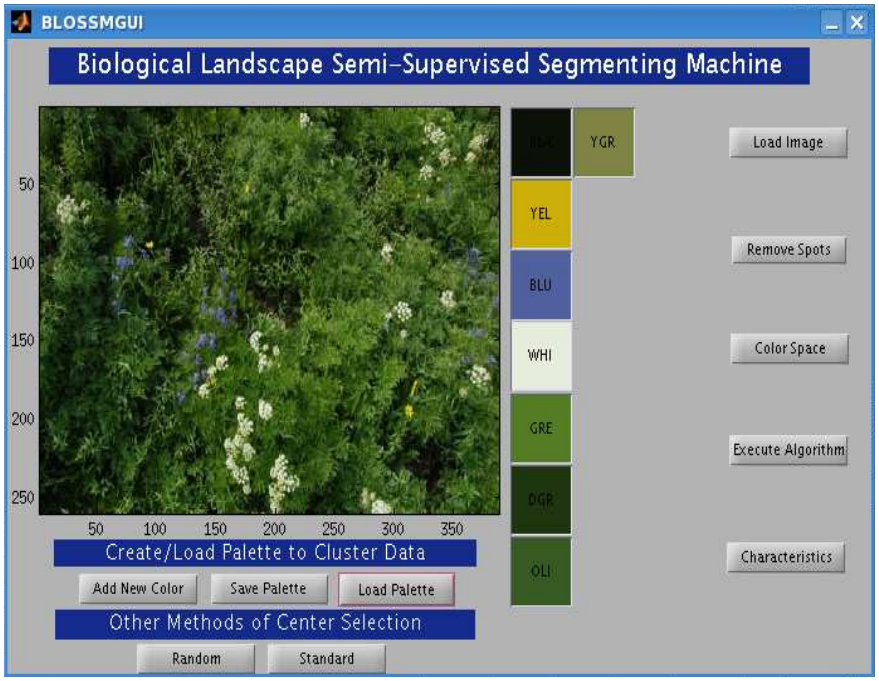


FIGURE 3.4. BLOSSM implemented on an image with 8 centers manually selected by the user to reflect distinct colors within image.

We have rescaled this image to be of size 260×388 (100880 pixels) purely to speed up computations. Do note that all of the techniques implemented in this chapter could be used on the full-size image, though. This is a sample landscape image that an ecologist might be interested in analyzing to determine the ground cover of a particular region, and thus, precisely the type of image that BLOSSM was designed to analyze. Further analysis of this image will be explored in Section 3.7.

The image itself as well as the GUI is displayed in Figure 3.4. We have initialized the LBG algorithm by selecting 8 starting centers as pixels from the image that we feel best identify the distinct colors within the image. These starting centers are displayed on the right of the GUI.

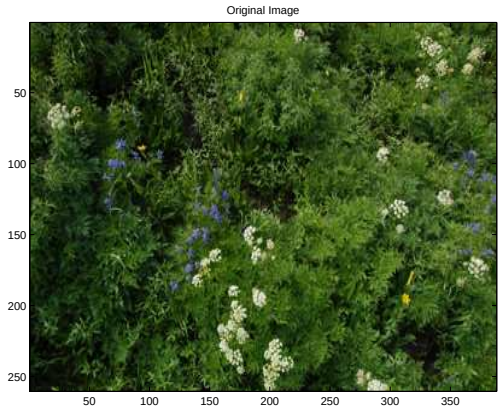
We will now analyze the performance across all color spaces and all metrics for this image, using the LBG clustering algorithm. The Figures 3.5-3.12 are grouped by color space and

the metrics are varied. We have chosen to compute the entropy, denoted as E, for each reconstruction as well as the distortion error, displayed in individual figures.

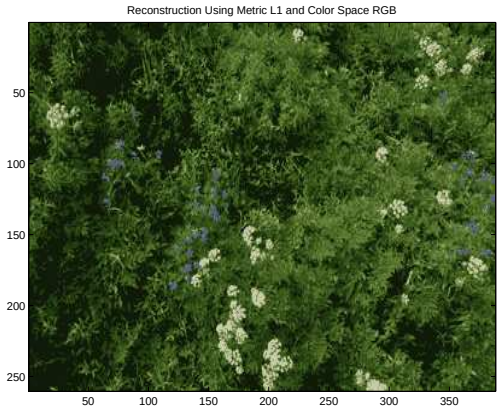
In Figure 3.5, we consider the color space RGB and vary the metrics. Notice that subtleties such as the yellow flowers do not appear as yellow using the ℓ_1 norm or spectral angle. This is a major concern if the focus of this clustering is to determine the ground cover of this landscape, with a particular emphasis on characterizing the flower cover. The blue is present in each metric but is a bit subtle. We observe that the Mahalanobis distance appears to denoise edges and has less variance in spatial regions, particularly in the foliage. Notice, however that if the goal is to distinguish flowers, the Mahalanobis distance gives the most contrast between the foliage and flowers. The spectral angle reconstruction appears quite fuzzy and blurred and visually is the worst reconstruction of the original. The entropy is the largest for the spectral angle, however. In comparing the distortion errors displayed in Figure 3.6, we see that spectral angle is much higher than the other measures with the ℓ_1 and ℓ_2 norms producing the smallest error.

The Quantized RGB color space, as might be expected, reflects similar reconstructions as RGB, Figure 3.7. The entropy is smaller for Quantized RGB than RGB in the ℓ_p spaces but larger in the other measures. Also, the distortion errors are much larger, Figure 3.8. Note the strange behavior in the Mahalanobis and spectral angle distortion error.

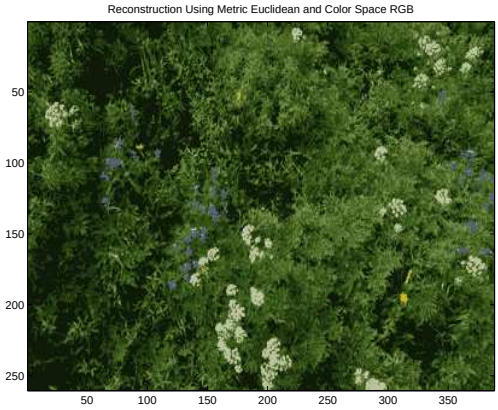
Reconstructing the landscape image in the Named Color space produces much different results than in the other color spaces, Figure 3.9. Notice that there does not seem to be much visual difference in the 4 shades of green (at least in every metric except spectral angle). This indicates that even though visually there does not seem to be as much difference, there is greater contrast in the 11-dimensional named color space with respect to the green shades.



(A) Original

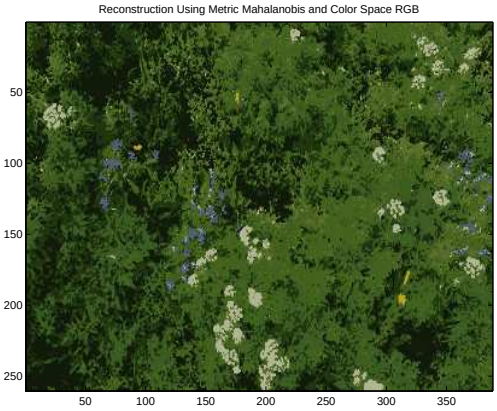


(B) ℓ_1, $E = 1.7091$

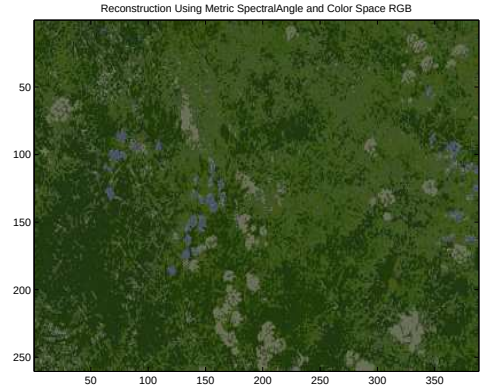


(C) ℓ_2, $E = 1.6343$

(D) ℓ_∞, $E = 1.6312$

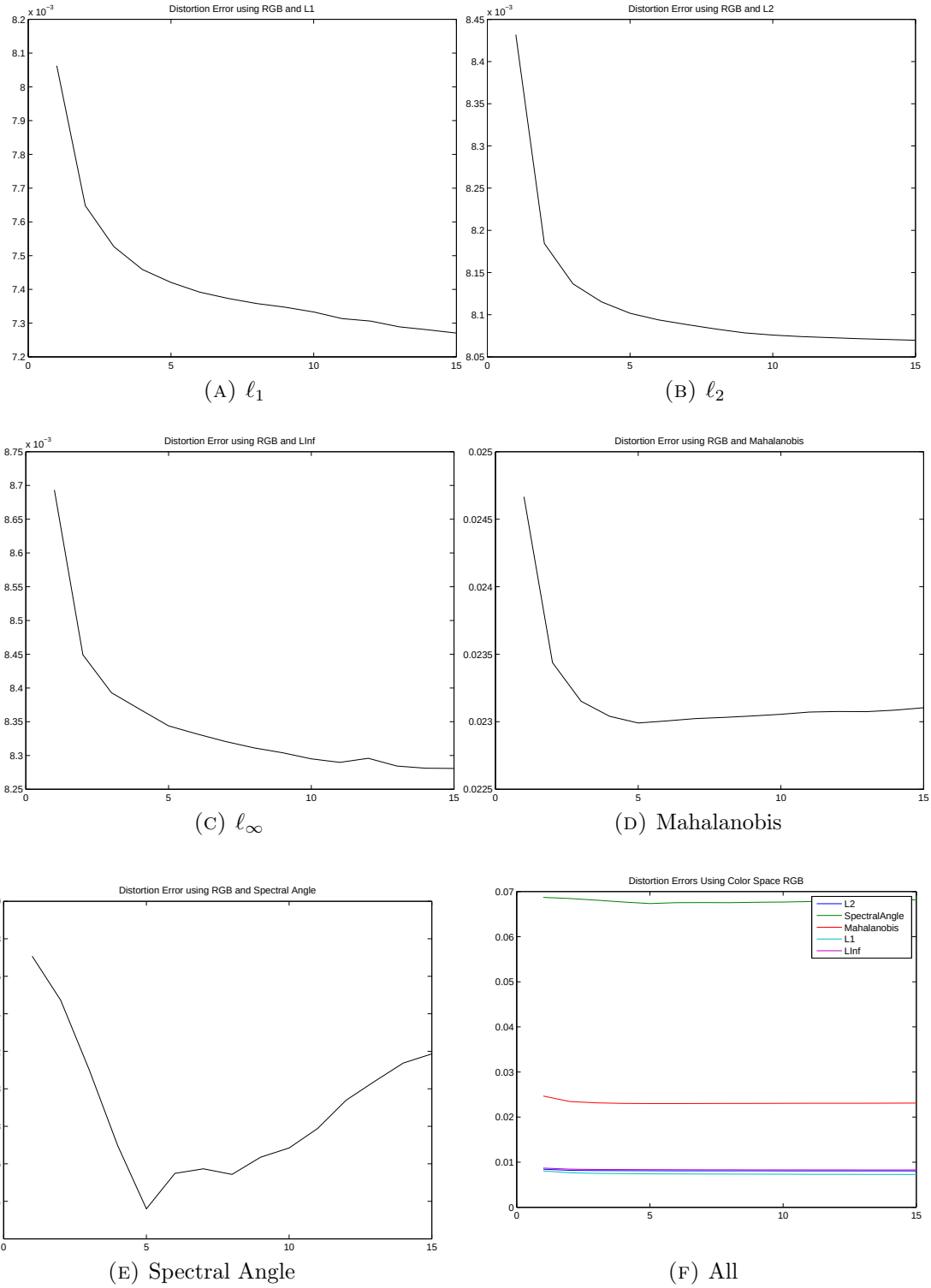


(E) Mahalanobis, $E = 1.668$



(F) Spectral Angle, $E = 1.8395$

FIGURE 3.5. Reconstructions using the LBG algorithm with fixed color space RGB and varying the metric used for still life image.

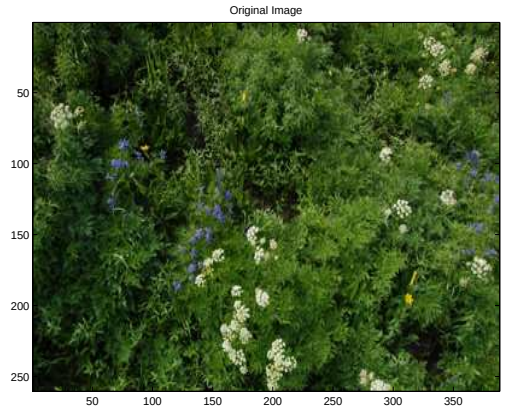


(A) ℓ_1 (B) ℓ_2

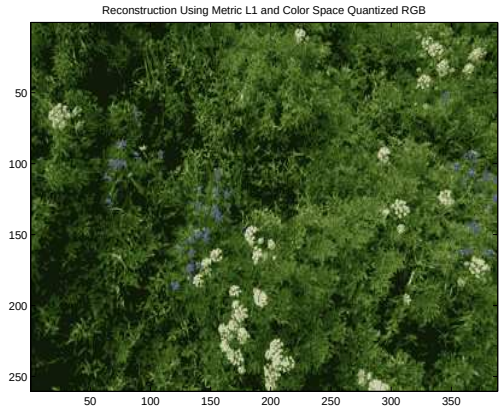
(C) ℓ_∞ (D) Mahalanobis

(E) Spectral Angle (F) All

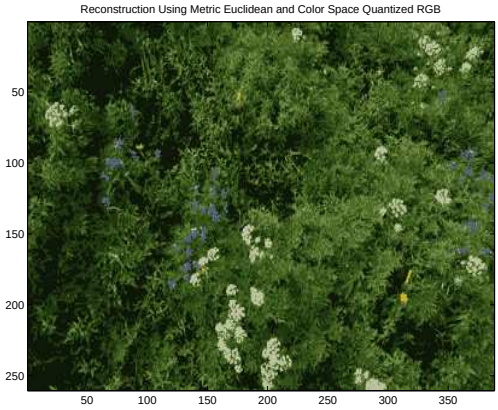
FIGURE 3.6. Distortion errors using the LBG algorithm with fixed color space RGB and varying the metric used for still life image.



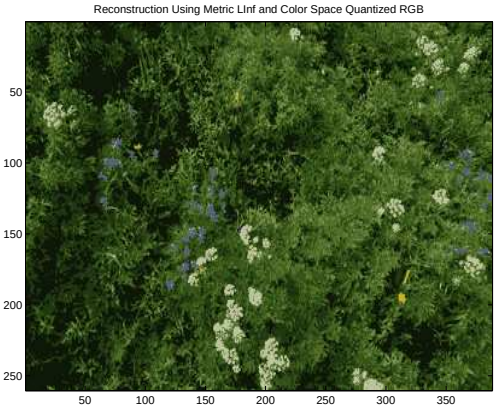
(A) Original



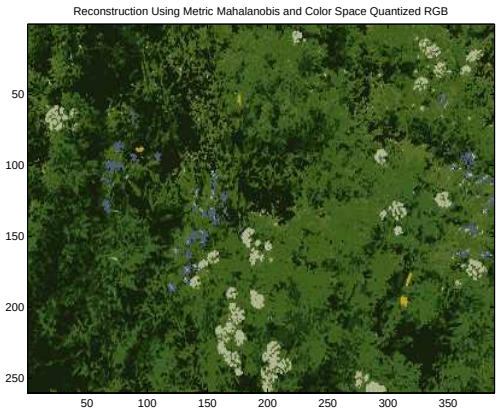
(B) ℓ_1, $E = 1.6596$



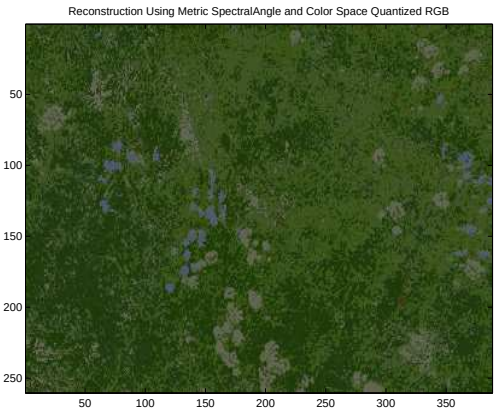
(C) ℓ_2, $E = 1.6312$



(D) ℓ_∞, $E = 1.5989$

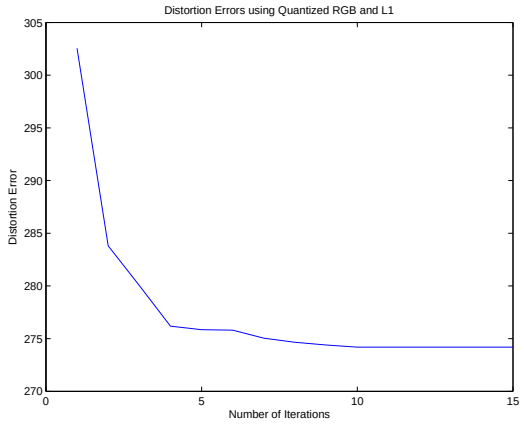


(E) Mahalanobis, $E = 1.6805$

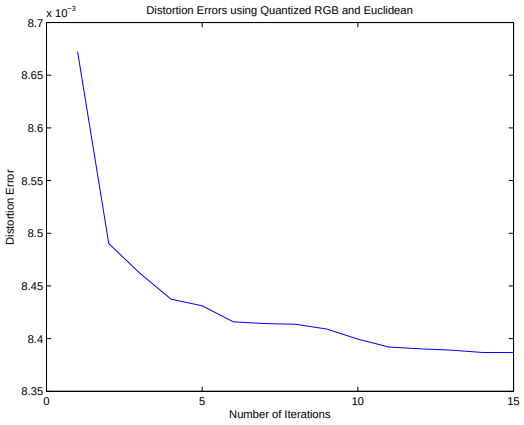


(F) Spectral Angle, $E = 1.8619$

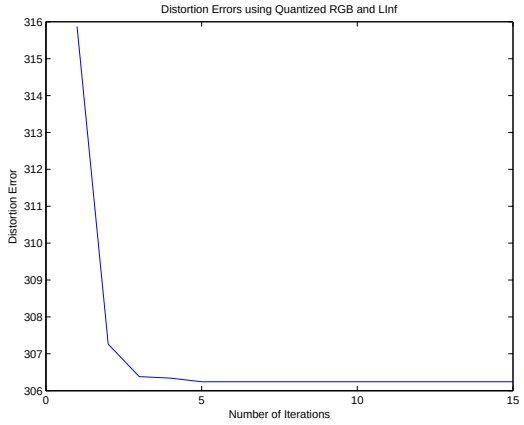
FIGURE 3.7. Reconstructions using the LBG algorithm with fixed color space Quantized RGB and varying the metric used for still life image.



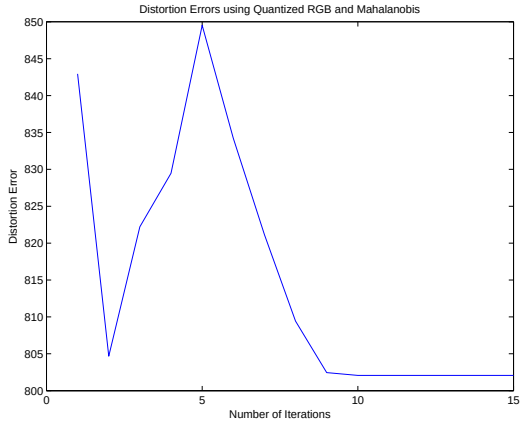
(A) ℓ_1



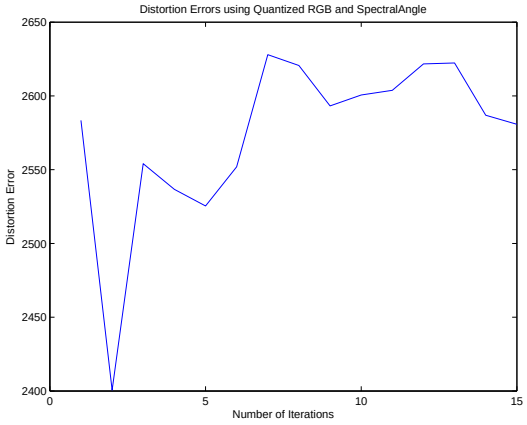
(B) ℓ_2



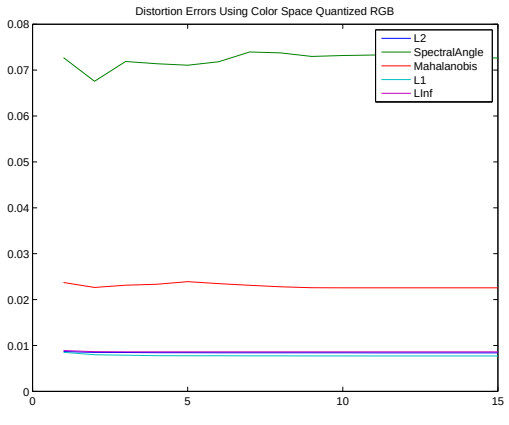
(C) ℓ_∞



(D) Mahalanobis



(E) Spectral Angle



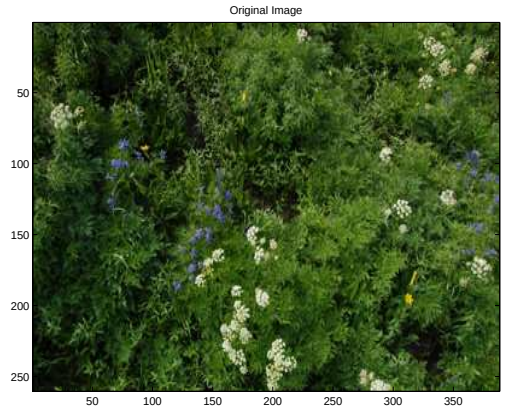
(F) All

FIGURE 3.8. Distortion errors using the LBG algorithm with fixed color space Quantized RGB and varying the metric used for still life image.

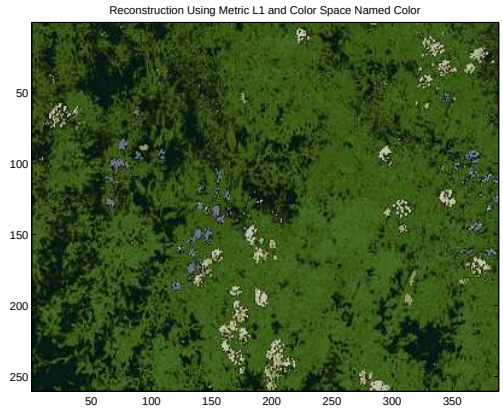
Note, though, that the colors of the flowers do not appear to be very vivid. However, there is sharp contrast between the flowers and the foliage using the Mahalanobis distance, making it easier to 'count' the number of flowers. In this case, spectral angle seems to give the most accurate visual reconstruction to the original in the shading of the foliage, yet the yellow flowers are not visible. Note that the spectral angle yields the smallest distortion error while Mahalanobis distance yields the largest, Figure 3.10. It is important to realize that these distortion error computations were done in the 11-dimensional Named Color space not the 3-dimensional space that all other color spaces reside in.

We will finally consider this image in the CIELab color space varying all metrics, Figure 3.11. The CIELab color space is the only space in which the yellow flowers are visible in the reconstructions for every metric (except for spectral angle). As we have seen in the other analyses, all three of the ℓ_p norms appear to give a very nice reconstruction, the Mahalanobis distance appears to smooth the objects, and the spectral angle gives a poor, blurred reconstruction. The distortion errors seem comparable to the RGB color spaces in that spectral angle has the most error while the ℓ_p norms have the smallest error, Figure 3.12. However, the error is much lower for CIELab. This is a fair comparison with the RGB color spaces as they are both measured in $\mathbb{R}^3$.

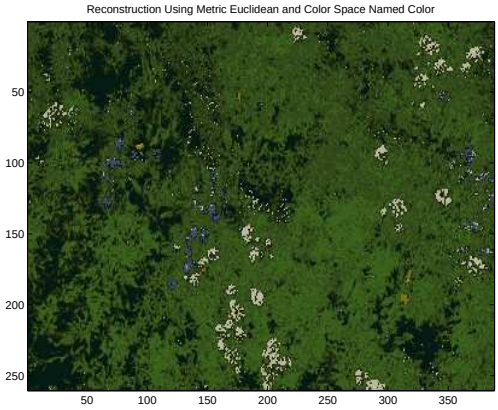
This analysis reveals that changing the color space and the metric will affect the reconstruction obtained after clustering with a vector quantization algorithm such as LBG. It seems that spectral angle performs poorly in each color space considered, except for Named Color. The ℓ_p norms generally perform well, but may miss some subtleties. The Mahalanobis distance appears to smooth out regions, providing less shading, but this may enable easier



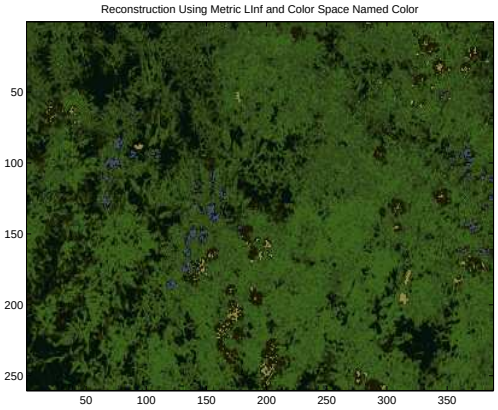
(A) Original



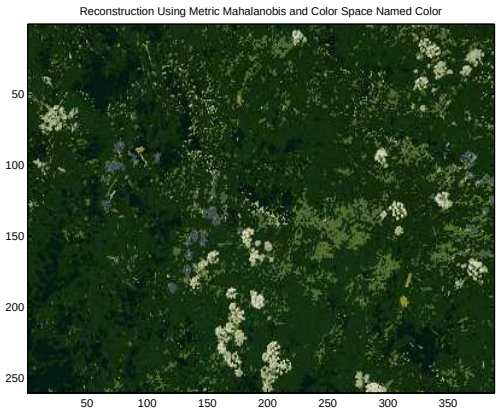
(B) ℓ_1, $E = 1.5647$



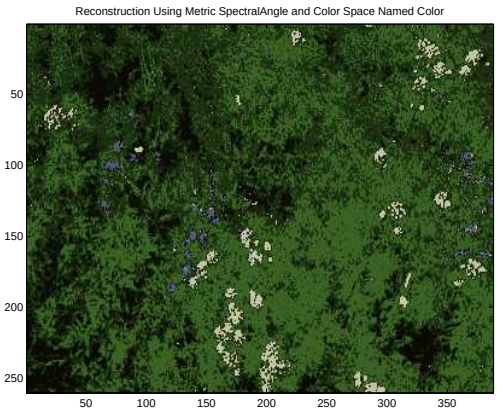
(C) ℓ_2, $E = 1.5761$



(D) ℓ_∞, $E = 1.6292$

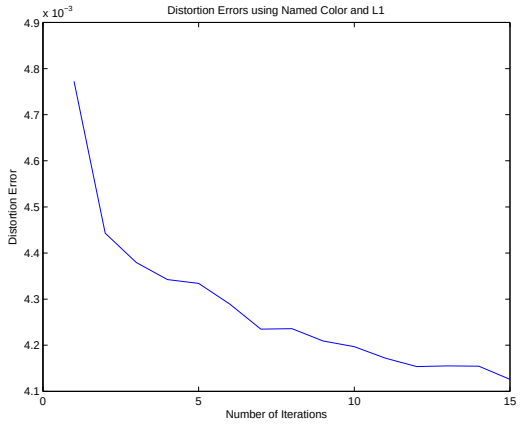


(E) Mahalanobis, $E = 1.3951$

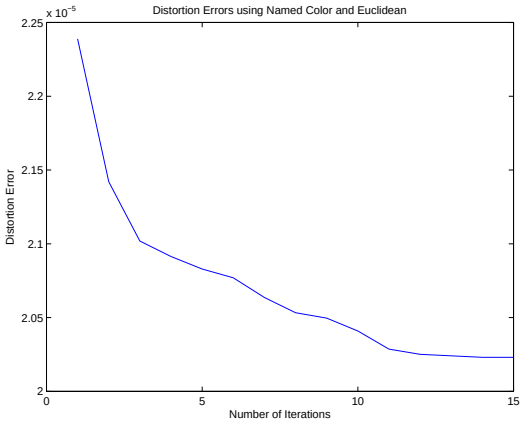


(F) Spectral Angle, $E = 1.4834$

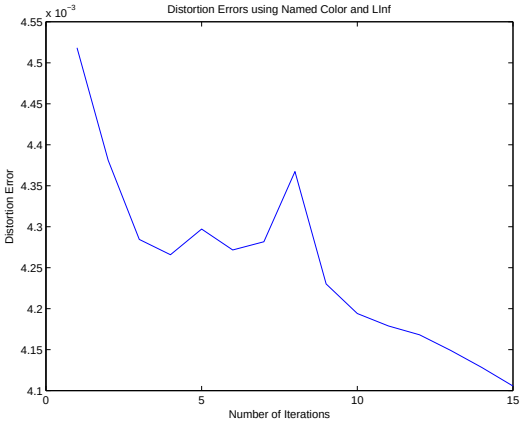
FIGURE 3.9. Reconstructions using the LBG algorithm with fixed color space Named Color and varying the metric used for still life image.



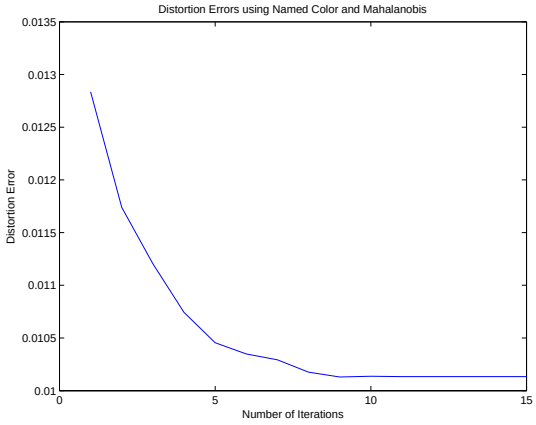
(A) ℓ_1



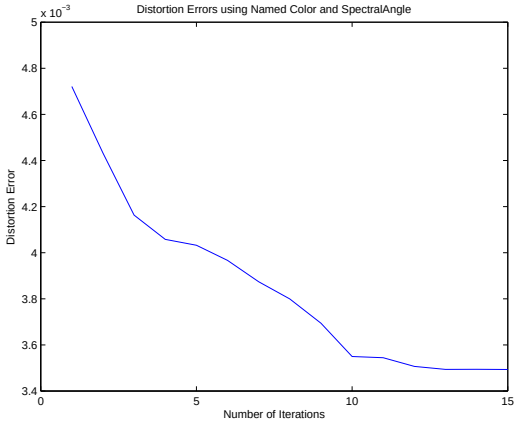
(B) ℓ_2



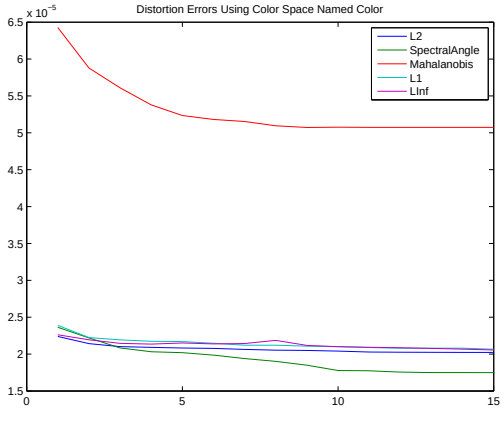
(C) ℓ_∞



(D) Mahalanobis

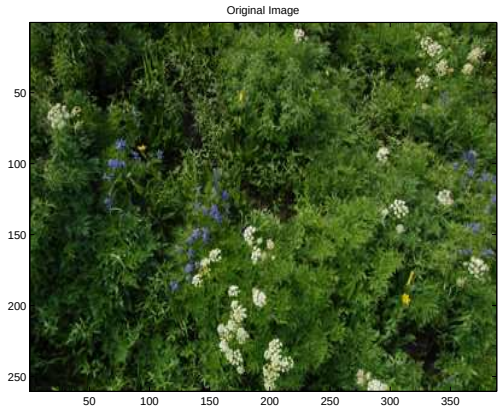


(E) Spectral Angle

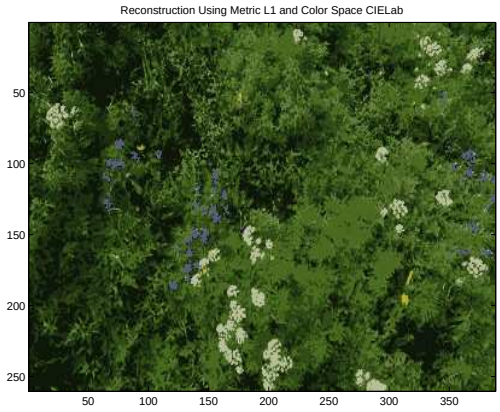


(F) All

FIGURE 3.10. Distortion errors using the LBG algorithm with fixed color space Named Color and varying the metric used for still life image.

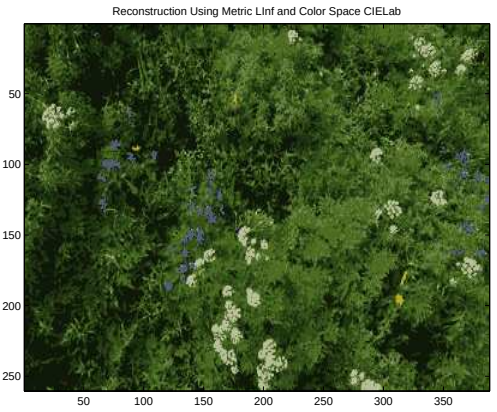


(A) Original

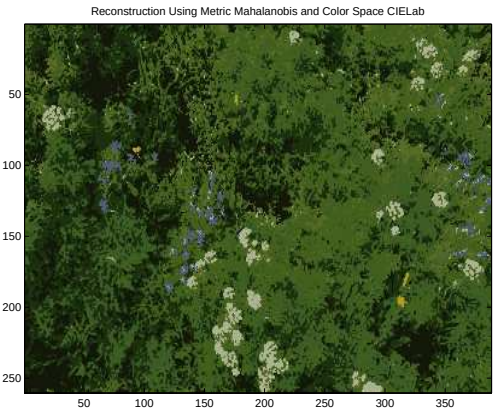


(B) ℓ_1, $E = 1.6469$

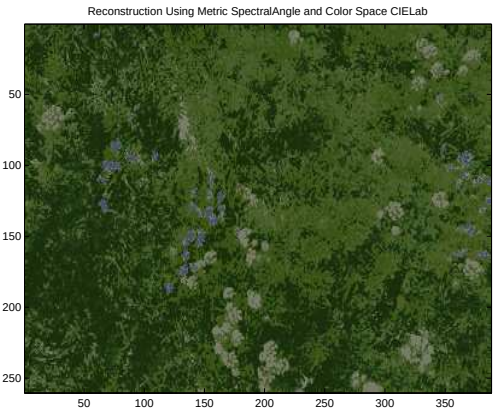
(C) ℓ_2, $E = 1.6606$



(D) ℓ_∞, $E = 1.652$

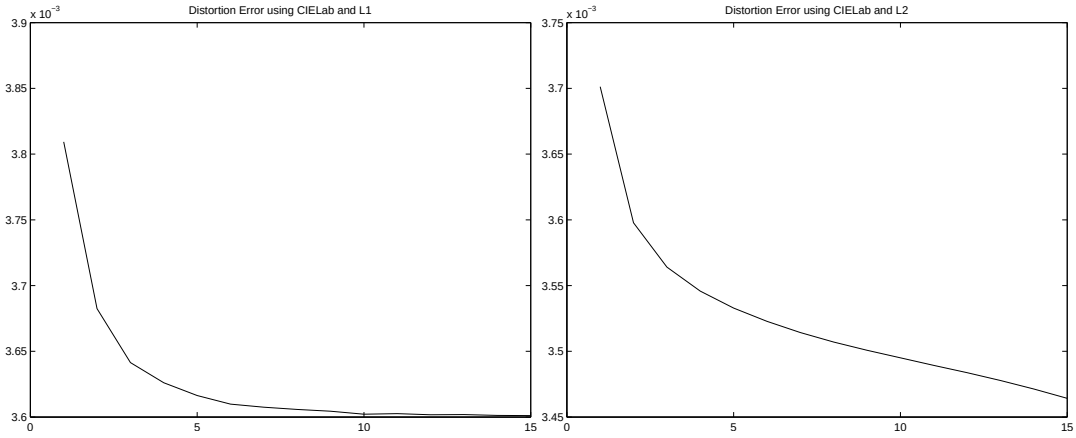


(E) Mahalanobis, $E = 1.6521$



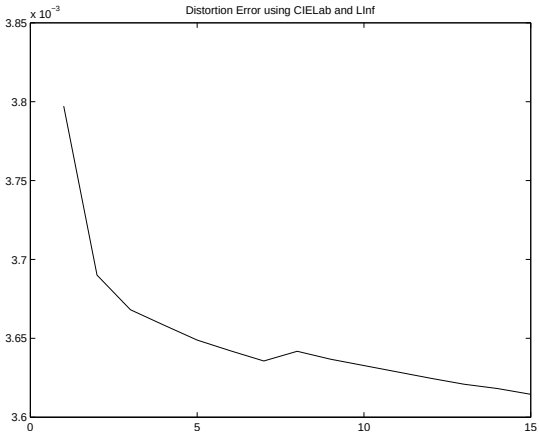
(F) Spectral Angle, $E = 1.7653$

FIGURE 3.11. Reconstructions using the LBG algorithm with fixed color space CIELab and varying the metric used for still life image.

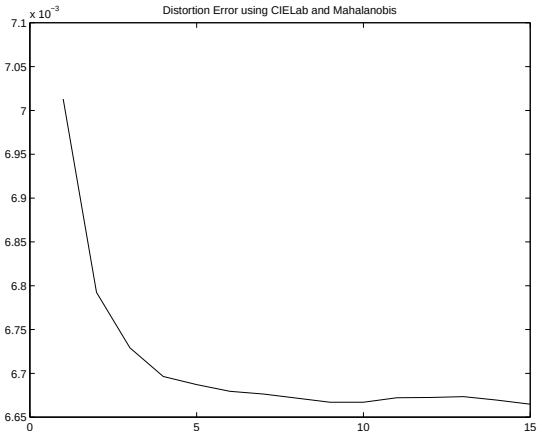


(A) ℓ_1

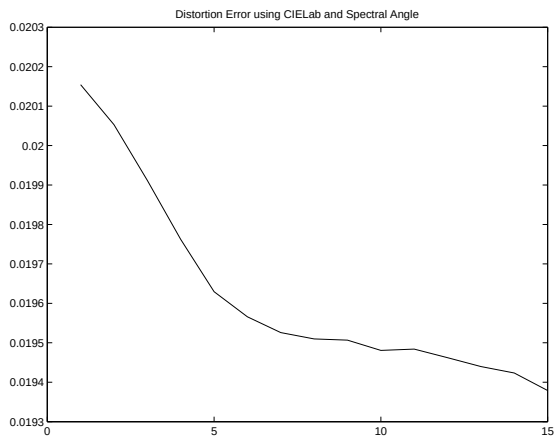
(B) ℓ_2



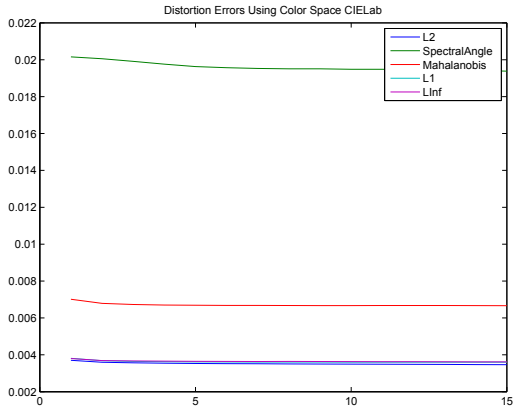
(C) ℓ_∞



(D) Mahalanobis



(E) Spectral Angle



(F) All

FIGURE 3.12. Distortion errors using the LBG algorithm with fixed color space CIELab and varying the metric used for still life image.

separation between objects and background. Depending on the application, an appropriate choice may be selected.

To further understand the effect of changing the color space and metric on clustering an image, we have chosen to analyze two additional images with different colorings and objects. This analysis is presented in Appendix A.

3.7. Ecology Application

In this section, we will make precise how an ecologist may use BLOSSM to analyze landscape images. Ecologists are often interested in determining the ground cover of a landscape. This analysis is often done in the field by manually counting species falling along the vertices of a grid within a certain plot. BLOSSM was designed to automatically perform this analysis by considering images.

As mentioned previously in Section 3.5 visual depictions of the color information, as determined by a set of starting centers and the LBG algorithm using a choice of color space and choice of metric, are available. In Figure 3.13, we see the reconstruction image of our landscape image clustered using the RGB color space with Euclidean distance (these choices will be used throughout this section). The resulting final centers are displayed on the right. A pie chart displaying the distribution of pixels identified with each color in the reduced color space (i.e. the final centers) can be shown, see Figure 3.14. Also, images associated to each of the individual clusters–created by displaying those pixels identified with each final center–can be revealed, see Figure 3.15.

A number of ecological variables can be ascertained using the information produced by BLOSSM:

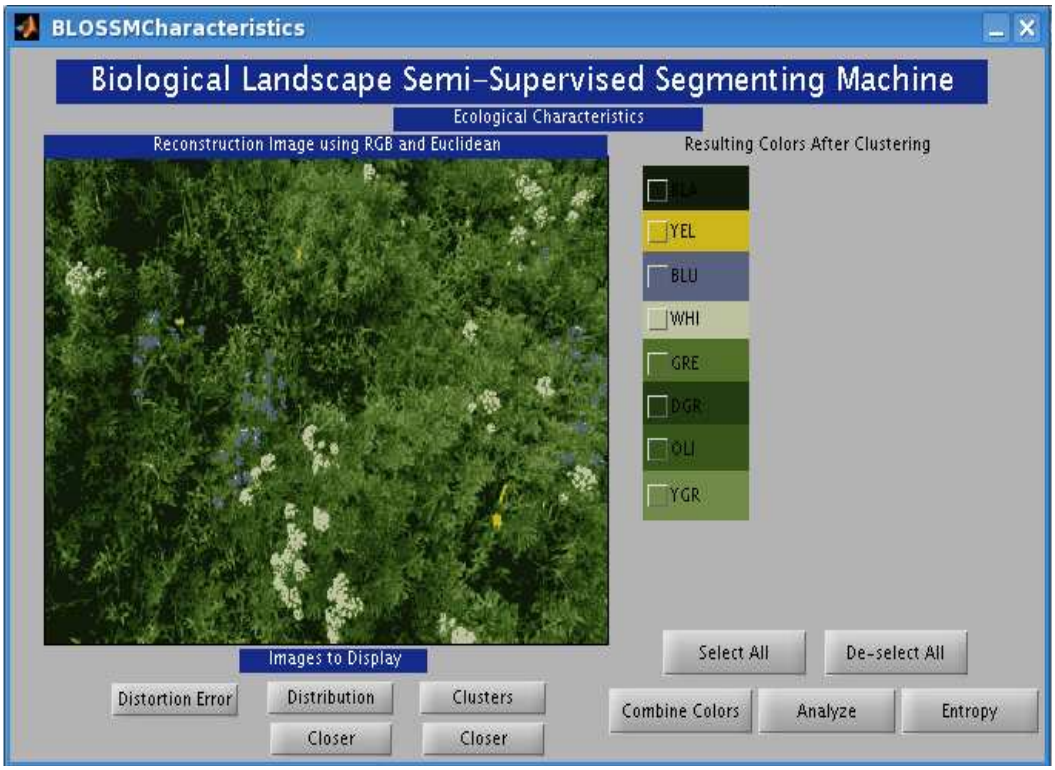


FIGURE 3.13. Characteristic GUI, displaying the reconstruction image and final centers found after clustering.

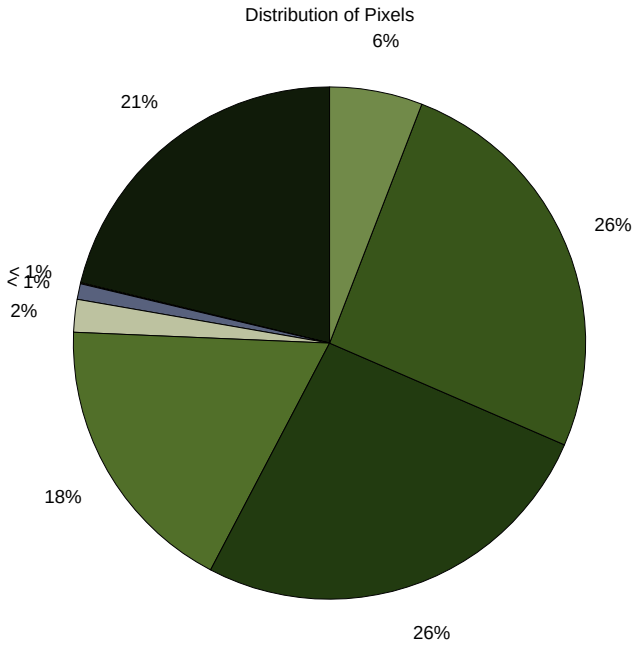


FIGURE 3.14. Distribution of clusters.

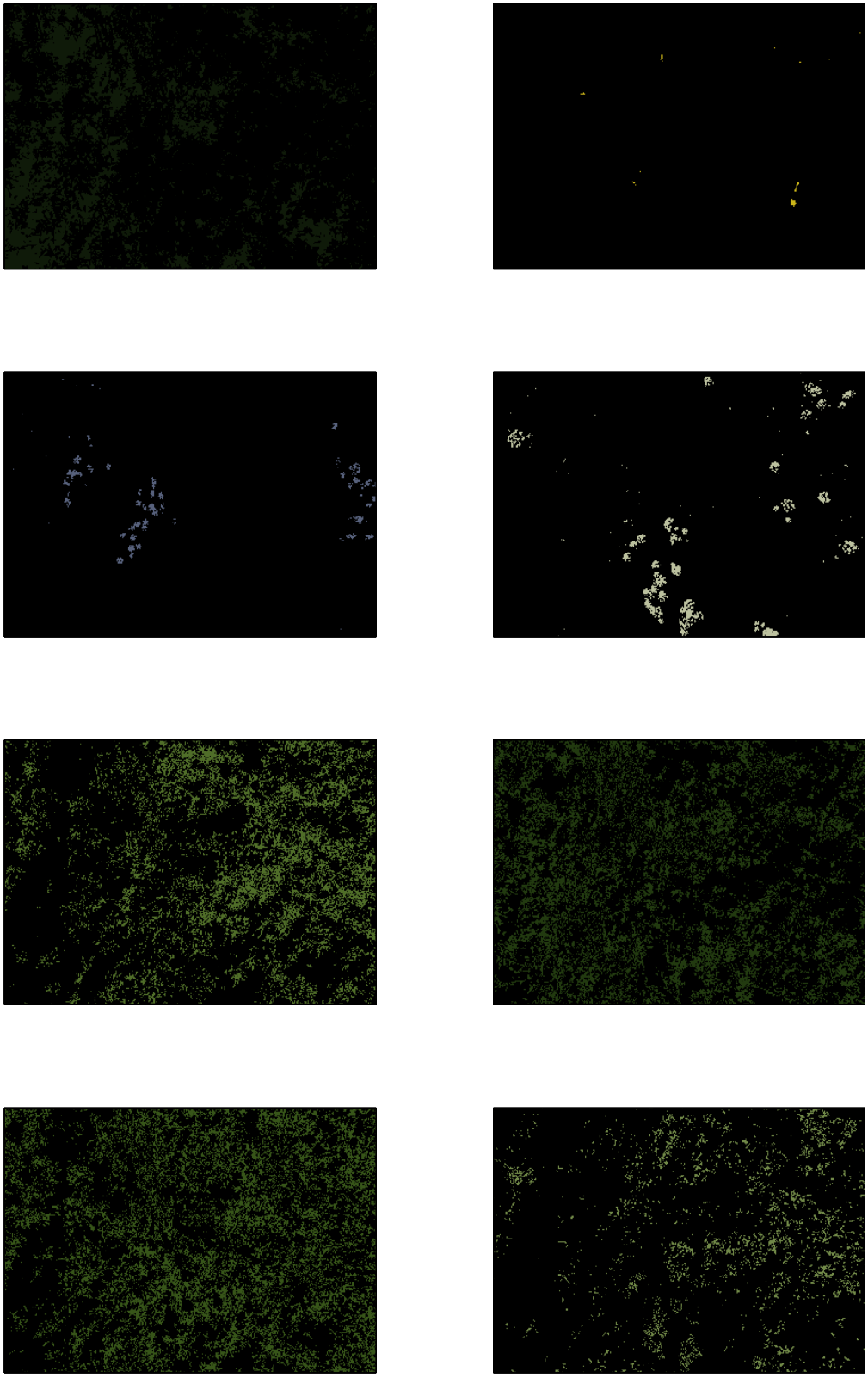


FIGURE 3.15. Individual clusters.

- Color abundance, typically the number of (used) centers or possibly fewer if colors are combined
- Number of flower clusters for each color
- Size of flower clusters for each color
- Abundance of 'morphospecies' as characterized by color and cluster size
- Richness, a count of the total number of morphospecies
- Species diversity, as measured by Shannon entropy defined as

$$H' = -\sum_i p_i \log p_i$$

  where p_i is the proportion of individuals belonging to the ith species in the dataset of interest
- Evenness, how close each species in an environment are, defined as

$$J' = \frac{H'}{H'_{max}}$$

  where H' is defined as above and $H'_{max} = \ln N$ with N the total number of species

For each cluster (color) to be analyzed, a visual depiction of some of this information is displayed in an image such as Figure 3.16. We see the number of pixels classified as that color, the proportion of the total number of pixels, the number of contiguous regions (flowers in this case), and the average size of each of these contiguous regions are all displayed. Also, we can see pixels that were considered to be noise are not included in the bottom image of 3.16 but can be seen in the middle image. This numerical data is displayed in a table for all distinct clusters chosen to be analyzed, see Table 3.1.

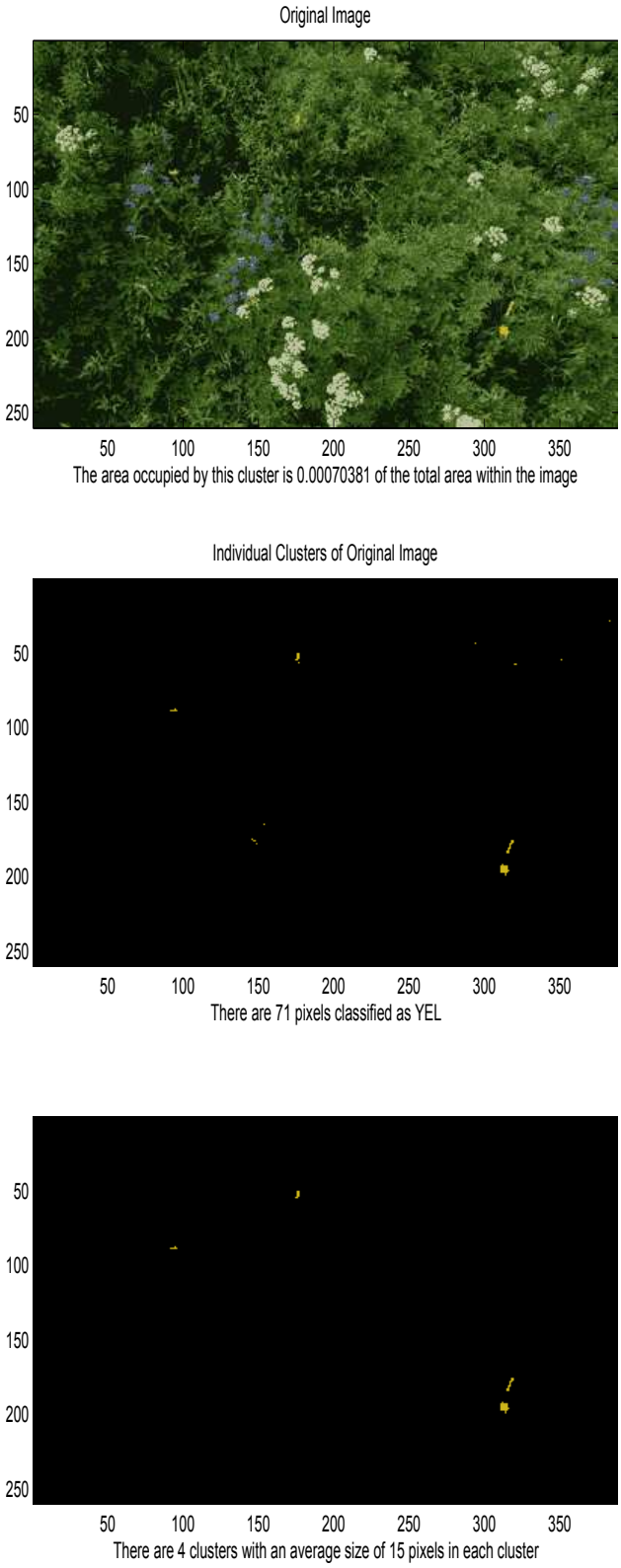


FIGURE 3.16. Characteristics regarding the yellow cluster of pixels.

TABLE 3.1. Table of characteristics regarding all clusters of pixels.

Color	Number of Pixels	Percent of Area	Number Regions	Avg Size Regions
BLA	21398	0.2121	354	55.6751
YEL	71	7.0381e-04	4	15
BLU	997	0.0099	37	24.6486
WHI	2074	0.0206	51	37.8039
GRE	18145	0.1799	524	29.2366
DGR	26410	0.2618	641	36.9610
OLI	25863	0.2564	675	33.7941
YGR	5922	0.0587	306	12.8105

TABLE 3.2. Table of characteristics regarding clusters of pixels with all green clusters combined into a single group.

Color	Number of Pixels	Percent of Area	Number Regions	Avg Size Regions
BLA	21398	0.2121	354	55.6751
YEL	71	7.0381e-04	4	15
BLU	997	0.0099	37	24.6486
WHI	2074	0.0206	51	37.8039
AGR	76340	0.7567	53	1.4355e+03

Note that in this analysis the entropy (or diversity measurement) was 1.6343, and thus, the evenness is 0.8219 as there are 8 distinct colors. Now, let us consider how some of these variables change as we combine colors, in this case the four shades of green. Table 3.2 displays the resulting output. The entropy for this computation was 0.67044, and the evenness is 0.4166, if it is determined that there are 5 species present. This low number indicates that this image does not have an evenness of species as evidenced by the relatively small number of flowers and the abundance of foliage.

3.8. Conclusion

In this chapter, we have considered a well-known algorithm, the LBG clustering algorithm, applied to analyzing landscape ecology images. We observed the impact of changing the metric as well as the color space under which this analysis was performed. A graphical user interface, BLOSSM, was developed to allow users to easily implement this algorithm

with a variety of input choices and output options. Finally, we described how an ecologist, or any other scientist, with images to analyze may use BLOSSM.

CHAPTER 4

Locally Linear Embedding Clustering Algorithm

4.1. Introduction

Manifold learning in data analysis assumes that a set of observations, taken as a whole, is locally well approximated by a topological (or even geometric) manifold. This assumption implies that the data is locally well approximated by a linear space, i.e., it is locally flat. A fundamental goal of manifold learning is to uncover the underlying structure of this approximating manifold and to find low dimensional representations that preserve the structure and topology of the original data set optimally [82], [75], [71]. A frequent simplifying assumption is that the local dimension is constant over the entire data set. Alternatively, one may model a set of data as a collection of manifolds, allowing for intersections and for variations in dimension. For instance, the union of the xy-plane and the z-axis is not a manifold but decomposes naturally as a union of two manifolds of differing dimension. We have found this multiple manifold assumption to be appropriate for natural imagery consisting of distinct objects, e.g., a landscape image consisting of flowers, cacti, and ground vegetation.

Points in a data set can typically be thought of as lying close to a low dimensional manifold if the points are parameterized by a relatively small number of continuous variables [56], [82], [31]. For instance, a manifold structure could underlie a collection of images of a single object undergoing a change of state (such as illumination, pose, scale, translation, etc.). One way to uncover this structure is to map the collection to a high dimensional vector

space by considering each image as a point with dimensionality corresponding to the number of pixels in the image and with coordinate values corresponding to the brightness of each pixel [71], [48], [75]. Many algorithms have been implemented on such data sets in order to uncover a low dimensional manifold that reflects the inherent structure of the high dimensional data. Linear methods such as Principal Component Analysis [50] and Multidimensional Scaling [27] have been around for many years while nonlinear methods such as ISOMAP [75], Locally Linear Embedding [71], Hessian Eigenmaps [31], and Laplacian Eigenmaps [9] are more recent and have proven capable of extracting highly nonlinear embeddings.

In this chapter, we focus on the Locally Linear Embedding (LLE) algorithm applied at the pixel level. More precisely, our data sets do not consist of a set of images but rather the pixels comprising a single image. Analysis of LLE applied to pixels has been implemented previously in [70] and has been considered in the context of hyperspectral images by [4], [42], [22]. These works confirm the existence of an underlying structure. The goal of this chapter is to utilize LLE to represent the underlying structure of color data in an image as a union of linear spaces and to quantize color space accordingly. While examples will be drawn from the color space of digital images within the visible spectrum, it is important to note that such images are a special case of hyperspectral imagery. Implementations in one setting can typically be implemented in the more general setting with minor modifications.

In the LLE algorithm, a weighted graph is first constructed from a set of data as a stand-in for the local manifold structure [71]. The algorithm next determines a set of d *embedding vectors* by discarding the eigenvector corresponding to the smallest eigenvalue of an associated graph Laplacian and keeping the 2^{nd} through $d+1^{st}$ eigenvectors. Arranging the d vectors as columns of a matrix, the rows of this matrix provide a map of the original

data to $\mathbb{R}^d$. The graph Laplacian encodes the number of connected components of the graph as the dimension of its null space. Interpreting results in the LLE algorithm becomes problematic if the null space has dimension greater than one as eigenvectors in a null space are unique only up to rotation. Thus, a canonical ordering of eigenvectors is ill defined. As it is quite reasonable to expect the color space of natural images to be lying on multiple manifolds, it is also natural to expect multiple connected components amongst the union of the manifolds. In order to alleviate this problem, we perturb the graph Laplacian in the direction of a circulant graph Laplacian to reduce the co-rank to one. From the d-dimensional embedding, we apply a technique for proximity/color segmentation. This is done through an unsupervised clustering algorithm that exploits the topology preserving properties of LLE.

Put another way, natural images are not random; they have structure in their color space in that adjacent pixels tend to have similar color values. These piecewise continuous variations lead to a piecewise manifold approximating the data. Through LLE, this piecewise manifold is revealed as a piecewise linear manifold. The segmentation is accomplished by uncovering the principal direction of an epsilon ball of points and segmenting the data such that points determined to be close enough to this principal vector and similarly colored to the center of the epsilon ball are classified together and removed from the data. This iterative approach has proven robust in the presence of noise, with the input parameters reflecting the accuracy of segmentation desired. Thus, the algorithm exploits the transformation of local one-manifold structure to local linear structure in the mapped data.

In Section 4.2, we present an overview of the Locally Linear Embedding algorithm and present the graph Laplacian perturbation to reduce to the case of co-rank one. Section 4.3 discusses the algorithm in conjunction with color quantization. We present an example to

observe that the geometric structure of a color image is revealed in a reconstruction image by exploiting locally-linear variations in pixel space through subspace segmentation. Section 4.4 demonstrates the algorithm's ability to reduce the color space of natural imagery and uses this algorithm in conjunction with the classical Linde-Buzo-Gray vector quantization algorithm [54], [60] in the context of a landscape ecology application. The contributions of this chapter include a method for resolving LLE rank issue problems (without carrying out a decomposition into connected components) in such a manner that the local topological structure of the data is preserved, a technique for subspace segmentation, and the development of an associated clustering algorithm.

4.2. Connecting Components in Locally Linear Embedding

Recall in Section 2.2 that we discussed the LLE algorithm in detail. We now present a couple of considerations when using this algorithm.

4.2.1. Special Considerations in Implementation of LLE on Natural Imagery.

As natural images have the feature that many pixel colors are quite similar, it would not be surprising to find pixels with identical colors. Thus, when we consider an image as a collection of points in $\mathbb{R}^3$, we may observe distinct image pixels whose distance apart is zero. In determining nearest neighbors for a point x_i, we have opted to only include points whose distance from x_i is greater than zero.

We have observed that for many natural images, the K nearest neighbor's graph has corank larger than 1 indicating more than one connected component. For disconnected data, LLE can be implemented on each of the graph's connected components separately [71]. In this paper we have chosen a different (and slightly unusual) path in that we artificially connect components by perturbing in the direction of a cycle.

4.2.2. Connecting Disconnected Components. The Laplacian of a graph has 0 as an eigenvalue with multiplicity equal to the number of connected components of the graph [25]. In a similar manner, the matrix M, in the final step of the LLE algorithm, has co-rank corresponding to the number of connected components within the data set where connections are made by linking each data point with its nearest neighbors. Choosing the number of nearest neighbors to be small can lead to many disconnected components. As previously stated, while [71] indicates that LLE be implemented on each of the graph's connected components separately, we have chosen to proceed down a different path by artificially connecting previously disconnected components through a perturbation (much like the second step in LLE that adds a regularization term to the covariance matrix C that would be singular in the case when $k > D$). Here, we perturb M in the direction of a matrix T that has a similar structure to M in that it is positive semidefinite with row sums equal to 0. We chose T to be the Laplacian matrix of a cycle, i.e.

$$T = \begin{bmatrix} 1 & -1 & 0 & & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ & & \ddots & & & \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & \cdots & & 0 & -1 & 1 \end{bmatrix}$$

Thus, in practice, we consider $M' = M + \lambda T$ where λ is a scalar. Matlab experiments suggest that using $\lambda = 10^{-9}$ produces a matrix M' that is artificially connected (i.e. has a corank of 1). As T comes from a cycle, the eigenvectors of M' inherit this circular structure while retaining much of the original topology of the pixel data.

4.3. Locally Linear Embedding Clustering

In this section, we consider a novel way to extend the LLE algorithm to a clustering algorithm. In natural imagery, the variation in color hues of neighboring pixels is often slight. Furthermore, pixels which are not spatially close may also exhibit very slight color variations. We can associate these slight color changes with continuous variations of pixel colors in an approximating manifold. Topological structure associated with these continuous variations is revealed by uncovering multiple underlying manifolds. These manifolds are detected using the Locally Linear Embedding algorithm. Segmenting based on these manifolds allows for quantization of the color space, leading to a clustering algorithm.

4.3.1. Clustering. Data clustering is the name given to creating groups of objects, or clusters, such that objects in one cluster have a shared set of features whereas objects in different clusters have less similarity with respect to these features. Clustering is a fundamental approach to segmenting data. There are many ways to attach pairwise similarity scores to a set of data points and it is important to realize that the clusters could have very different properties and/or shapes depending on these scores. Clustering can be done in a hierarchical manner where clusters are determined by using previously established clusters or in a partitioning manner where the data is clustered simultaneously into disjoint sets. In semi-supervised or constrained clustering algorithms, additional information such as data labels, information about the clusters themselves, etc. is available and utilized [7], [54]. Unsupervised clustering algorithms, in which data is organized without any information like data labels, however, can identify major characteristics or patterns without any supervision from the user. The algorithm of this paper, described in the following subsections, is non-hierarchical and unsupervised.

4.3.2. Why Locally Linear Embedding? The Locally Linear Embedding algorithm, as discussed in Section 4.2, is a dimensionality reduction algorithm that can help uncover the inherent structure and topology of higher dimensional data by determining a map to a lower dimensional space that optimizes for neighborhood relationships. While LLE was intended for the purpose of revealing topological structure, the creators of this algorithm indicate in [71] that some of the ideas presented in LLE, namely the first and third steps, are similar to those of the Normalized Cut algorithm discussed in [72] and other clustering methods such as the one discussed by [61].

Through experimentation, it was observed that if a natural image, considered as a set of RGB color points, is embedded in $\mathbb{R}^2$ using nearest neighbor sets of size 4 and a perturbation matrix T with $\lambda = 10^{-9}$ (as discussed in Subsection 4.2.2), then the data lies on a relatively small collection of lines. When the inherent color of each of the higher-dimensional input vectors was superimposed on the corresponding reconstruction vectors, it was noticed that similar colors fell along the same line. We observed simpler (but similar) behavior when considering a fixed pixel in a set of images of a fixed object under changing ambient illumination conditions. In each of these two cases, the lines in the reduced space suggest a method for quantization.

The following example uses a set of images from the Pattern Analysis Laboratory (PAL) database at Colorado State University. In the data, an individual remained motionless as the ambient illumination conditions were altered (with lights of fixed spectral characteristics). Figure 4.1 shows three such images with different illumination conditions. A data set was formed by considering the RGB values of a single, fixed pixel extracted from 200 such images.

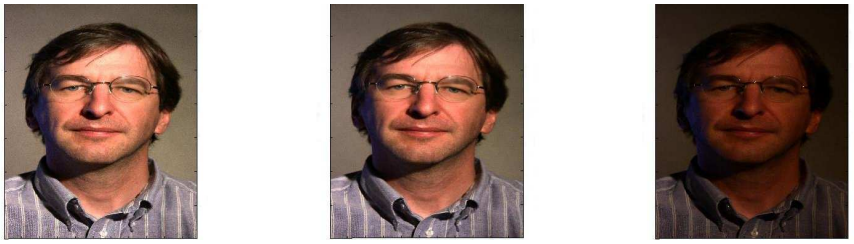
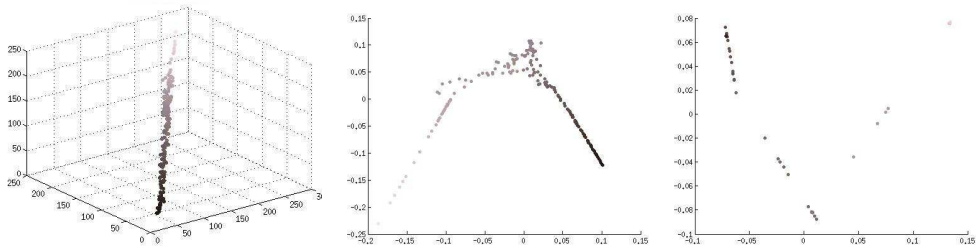


FIGURE 4.1. Images generated by Pattern Analysis Laboratory at Colorado State University where an individual remains motionless and the illumination of the surrounding area varies.

The LLE algorithm was implemented on this data set and mapped into a 2-dimensional space using $k = 4$ nearest neighbors. The null space of M turned out to be 4-dimensional indicating 4 connected components. Thus, our choice of nearest neighbor has artificially disconnected this data set that intuitively should be connected given that it is the smooth variation of illumination at a fixed point. Therefore, either we need to increase the number of nearest neighbors or perturb the data in such a way that the data is reconnected. Note that the smallest value of K that yields a 1-dimensional null space for M is $k = 8$.

In Figure 4.2, we see the original plot of the RGB data points, a plot of the embedding vectors using $k = 8$ nearest neighbors, and a plot of the embedding vectors using $k = 4$ nearest neighbors with the perturbation discussed in Section 4.2.2 that artificially reconnects the data set. We observe that the original three dimensional data appears relatively linear. Using $k = 8$ nearest neighbors, there is a degradation of this linear structure. However, using $k = 4$ nearest neighbors, with M perturbed by T, preserves the linear structure at a local level.

Therefore, we have observed two important properties of LLE. First, if a data set is disconnected using a choice of K nearest neighbors, it can be artificially reconnected using an appropriate perturbation. Provided the perturbation is not too extreme, this does not



(A) Data in $\mathbb{R}^3$ (B) Data in $\mathbb{R}^2$, k=8 (C) Data in $\mathbb{R}^2$, k=4

FIGURE 4.2. Plots of 3D data points generated by extracting the RGB values of a single pixel for each of 200 images and their corresponding embedding vectors as reconstructed by the LLE algorithm using k=8 nearest neighbors, and k=4 nearest neighbors with 'connected' data points.

affect the local topology of the data as expressed by LLE, as the linear structure of similar colors falling along one dimensional subspaces holds for each component. Second, the LLE algorithm is able to uncover the gradation of hue or variance of illumination within an image which corresponds to a linear structure in the plot of the reconstruction image of an RGB color space. Using this linear structure, in which each data point of the reconstruction image is colored according to its corresponding high-dimensional data point, a color space clustering algorithm is obtained.

Essentially, the Locally Linear Embedding Clustering (LLEC) algorithm segments the distinct linear manifolds that appear in the reconstruction plot of the LLE algorithm and then identifies which data points lie close to which subspace. The RGB color information is then overlaid onto the reconstruction data and further used to segment the data by clustering similarly colored points together.

4.3.3. SUBSPACE SEGMENTATION. We propose a subspace segmentation technique in the LLEC algorithm that involves selecting a point, $\mathbf{y}^*$, in the reconstruction data and then constructing an epsilon ball of appropriate size centered around this point. The point $\mathbf{y}^*$ may be chosen randomly or with a more deterministic criterion such as those discussed below.

If $\mathbf{y}^*$ is selected randomly, then there is an element of randomness in the algorithm, allowing for a variety of subspaces to be determined, depending on each $\mathbf{y}^*$ selected. Here we will discuss three non-random approaches that select points reflecting the data density.

One method is to select $\mathbf{y}^*$ to be the point within the data set that has the most points falling within an epsilon ball of the point. Another method is to select $\mathbf{y}^*$ to be the point whose b^{th} nearest neighbor is closer to it than any other point within the data set. Here $b \in \mathbb{Z}^+$, an arbitrary number. While the first idea is computationally intensive, the second has the complication that points are being removed from the data set, so there may occur a moment in the algorithm where the number of data points $p < b$. In this case, we adjust b to be a number less than the number of data points. For instance, $b = \lceil \frac{p}{2} \rceil$, where $\lceil * \rceil$ is the ceiling function, works well in practice.

An alternate approach for selecting the point $\mathbf{y}^*$ searches for the point that falls on a subspace which most reflects a linear structure. This can be implemented as follows. Determine an epsilon ball around each point. Compute the singular value decomposition of a matrix formed by the points in each of these epsilon balls in order to find the singular vectors and singular values. Choose $\mathbf{y}^*$ to be the random point with the smallest ratio of singular values $\frac{\sigma_2}{\sigma_1}$ as this reflects the subspace with the most linear structure. While this approach chooses points falling along structures most easily identified as 'linear', one downside is that it is computationally intensive. It typically produces reconstructions with a smaller distortion error than all of the other methods described above, but it does so by identifying more subspaces.

The various methods for identifying $\mathbf{y}^*$ have features that make each of them attractive, depending on the user's desired result. In this paper, we have chosen to follow the

method that chooses $\mathbf{y}^*$ as a point in a dense region of the data by finding the b^{th} nearest neighbor with the smallest distance. This is the least computationally intensive and produces reconstructions with a relatively small distortion error using relatively few subspaces to reconstruct. Here we have selected $b = 50$.

Once $\mathbf{y}^*$ is selected, a data matrix, A, is formed by inserting each embedding vector falling inside the epsilon ball centered at $\mathbf{y}^*$ into the rows of the matrix. The singular value decomposition, $A = U\Sigma V^T$, is computed to determine the right singular vector corresponding to the largest singular value. This singular vector corresponds to the line that passes through the mean and minimizes the sum of the square Euclidean distances between the points in the epsilon ball and the line, indicating the principal direction of these data points [50]. This unveils the 1-dimensional subspace that we are after. Note a method for K-dimensional subspace segmentation is discussed below. In order to segment the data via its proximity to each subspace, we then determine which points $\mathbf{y}_i$ in the data set satisfy $||\mathbf{y}_i - \mathbb{P}\mathbf{y}_i|| < \epsilon_1$ where ϵ_1 is some tolerance to identify those points that are 'close enough to' the subspace in consideration, and $\mathbb{P}$ is the projection onto the first right singular vector.

Now, as many lines overlap or intersect, simply using proximity to the subspace will not yield an appropriate segmentation. Thus, the data is further segmented by identifying those points, $\mathbf{y}_i$, whose RGB color values, $\mathbf{x}_i$, are most similar to the color of $\mathbf{y}^*$, $\mathbf{x}^*$, by computing $||\mathbf{x}_i - \mathbf{x}^*|| < \epsilon_2$ where ϵ_2 is again some tolerance to identify which points are 'similarly' colored to $\mathbf{y}^*$. Points that are identified as being 'close enough to' the subspace generated by the random point and 'similarly' colored to this point are identified together and removed from the data set. The process is repeated until all points have been identified with a distinct

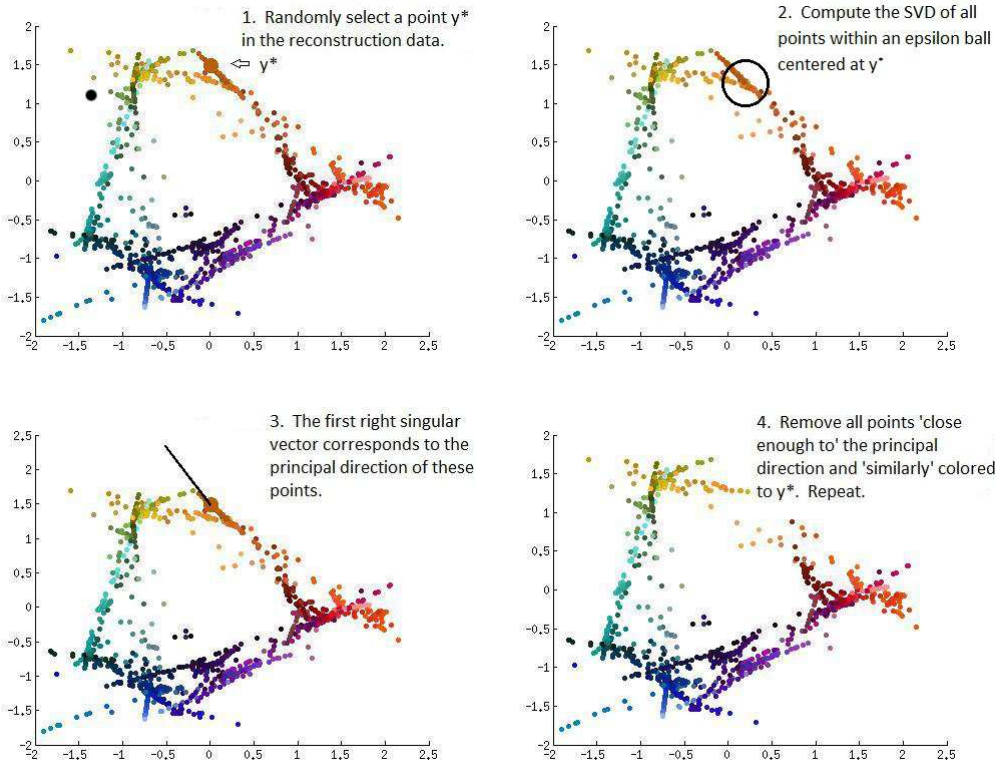
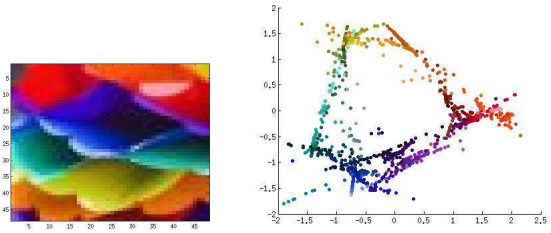


FIGURE 4.3. Illustration of subspace segmentation algorithm.

subspace using both proximity and color information. Thus, we obtain a clustering based on both color and spatial proximity metrics (in the 2-D representation).

Figure 4.3 provides a step-by-step illustration of this subspace segmentation algorithm. Also refer to Figure 4.4 to see an actual implementation of this algorithm. Observe that the subimages represent the subspaces iteratively removed from the reconstruction data.

We have described the approach for 1-D subspace segmentation, but this method can be used for multiple dimensions as well by considering pixels sufficiently correlated with the first several principal components. Thus, if an m-dimensional subspace segmentation is desired, the m right singular vectors corresponding to the m largest singular values form the principal directions of the data and span the subspace we want to uncover. Once this subspace is uncovered, the rest of the approach described above is analogous for a multidimensional



(A) Original

(B) 2D Reconstruction

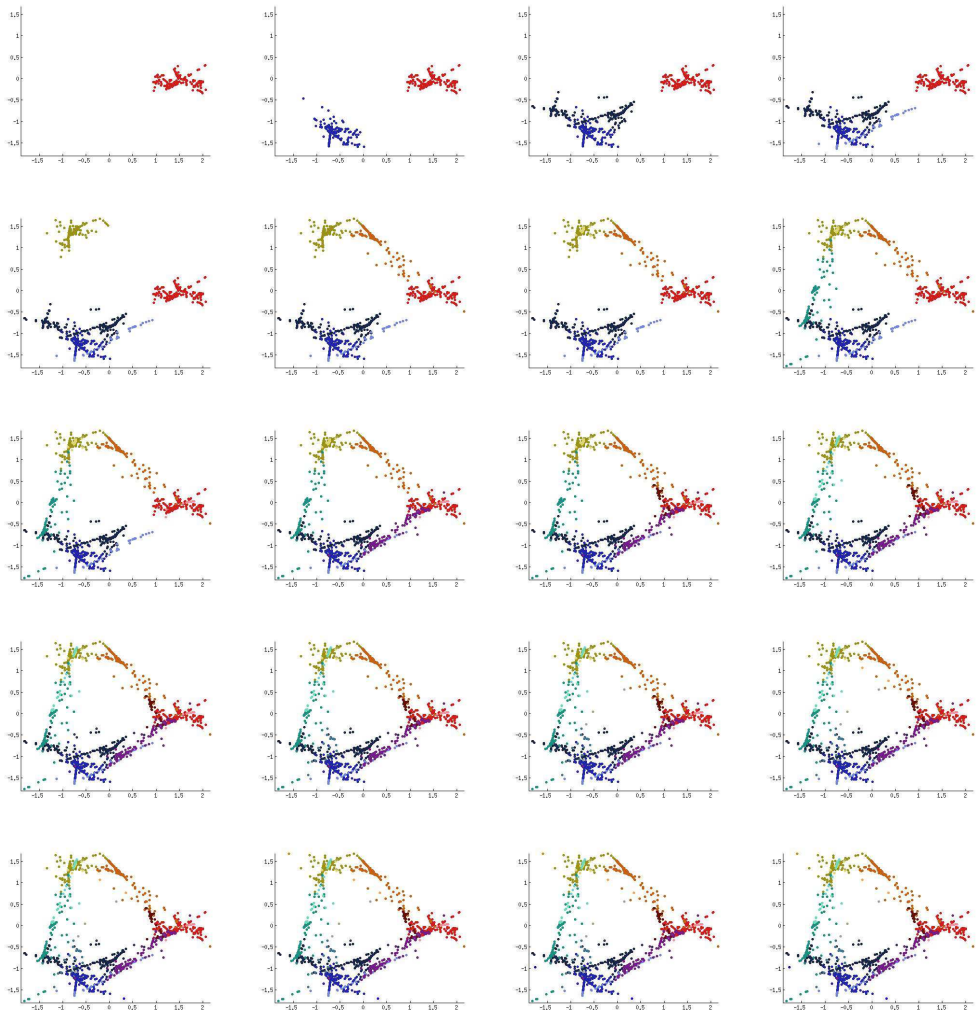


FIGURE 4.4. Subspace segmentation of 2D reconstruction data using LLEC with accuracy tolerances of approximately 0.4 by initializing $\mathbf{y}^*$ as the point whose b^{th} nearest neighbor has the smallest distance.

subspace segmentation. Note that this subspace segmentation technique has proven robust in the case of noise whereas methods such as Generalized Principal Component Analysis (GPCA) [81] have proven less effective in our experiments.

4.3.4. The Locally Linear Embedding Clustering Algorithm. The LLEC algorithm for quantizing the color space of natural imagery is summarized in the steps below. First, embed a data set X with points of dimension $D = 3$ into a lower dimension $d = 2$ using $k = 4$ nearest neighbors by using LLE. Next, identify the distinct subspaces appearing in the reconstruction data, Y, of the LLE algorithm by using the process described in the previous section. Through this subspace segmentation, determine the Voronoi sets, S_i, formed by identifying those points that are 'close enough to' each subspace and 'similarly' colored to the point $\mathbf{y}^*$. Note that the number of distinct Voronoi sets is the number of distinct subspaces. Let's call this number S. Then, calculate the mean of the colors, $\mu_i = \frac{1}{|S_i|} \sum_{\mathbf{y} \in S_i} \mathbf{y}$ of each set. Finally, identify all points $\mathbf{y} \in S_i$ by the prototype μ_i. Note that each $\mathbf{y}_i$ in the reconstruction data corresponds to a unique $\mathbf{x}_i$ in the original data space, so this determines a clustering of the data set X.

4.4. Implementation

As indicated in [30] a major complication in color space quantization often relates to varying shades of a given color due to illumination. We have observed that the LLEC algorithm handles this illumination component by identifying various shades of a given hue as a unique subspace and all pixels that are elements of this subspace can be identified together.

At this time, the LLEC algorithm is not a fast algorithm as the procedure for performing LLE and the search to determine those points that are 'close enough to' each subspace and

'similarly' colored to the random point being considered are computationally intensive. We will see however that LLEC does an excellent job of quantizing the color space of natural imagery. A benefit of LLEC is that the only free parameters in the algorithm are ϵ_1 and ϵ_2, the tolerances which can be specified by the user to reflect the desired accuracy of the quantization. The value of LLEC then is that it can be implemented on an image to determine the natural subspaces of the color space. The knowledge obtained by segmenting these subspaces can then be used in conjunction with other clustering algorithms such as the Linde-Buzo-Gray (LBG) [60] vector quantization algorithm to identify the starting centers as the subspaces unveiled in the LLEC algorithm. We will see this applied shortly.

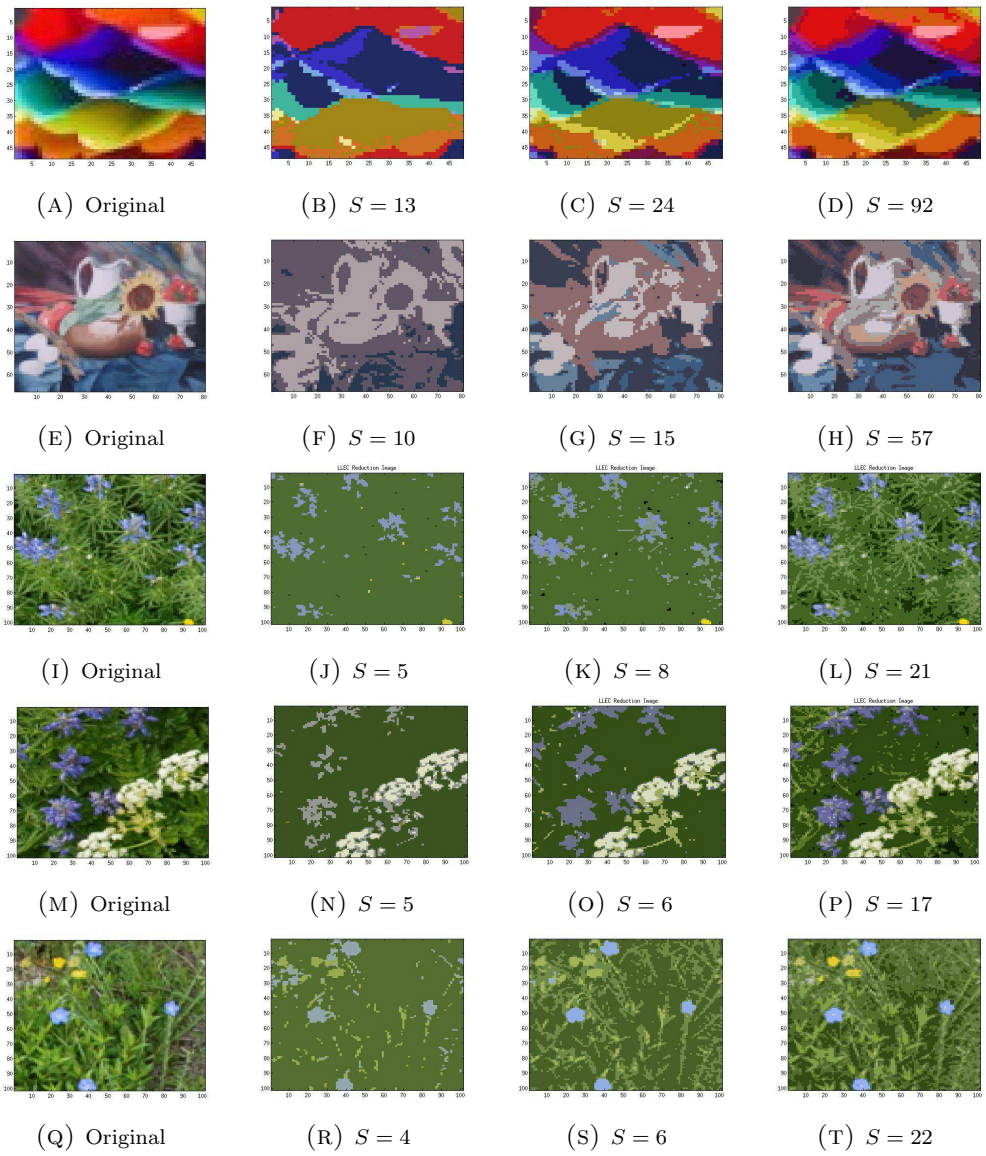
4.4.1. LLEC used to Quantize Color Space. First, let's consider the ability of the LLEC algorithm to quantize the color space of a variety of images. In each of these examples, the images were processed using MATLAB. Each sheet pertaining to the red, green, or blue component of pixels within an image was converted from a matrix of dimension equal to the resolution of each image to a long row vector of dimension $1 \times p$ where p is the number of pixels in the image. A new matrix, X, of dimension $3 \times p$ was created to contain all of the data entries of these long row vectors. By organizing the data this way, we see that each column of the matrix corresponds to the RGB components of an individual pixel which is a data point to be analyzed. We have chosen to use the Euclidean metric to calculate distance–a measure of proximity–between points.

Let's first consider LLEC's ability to segment the color space of a natural image and then use this segmentation to quantize the color space. We highlight in Figure 4.4 LLEC's ability to segment the subspaces in the 2-dimensional plot using accuracy tolerances of approximately 0.4 for a sample image. In Figure 4.5, we observe the color quantizations

obtained for this image as well as others using various accuracy tolerances. Note that in Figure 4.5, each of the original images were of resolution 100×100 or less. We see that as ϵ_1 and ϵ_2 decrease, the reconstructions become better representations of the original images.

4.4.2. LLEC Implemented with LBG. Let's now see how LLEC can be implemented in conjunction with another clustering algorithm on a class of large images. These images are of a subalpine meadow near the Rocky Mountain Biological Laboratory in Gothic, Colorado provided by Dr. David Inouye of the University of Maryland. Each image is of resolution 2592×3872 which generates 10,036,244 pixels. We cannot implement the LLEC algorithm directly on images from this landscape data set as its implementation requires constructing a pixel by pixel matrix. We have chosen to implement LLEC in conjuction with the Linde-Buzo-Gray algorithm [60], [54] on a set of these images. The simplicity of the LBG algorithm makes it desirable, but other clustering algorithms such as those discussed in [43], [44], [54], [7], etc. could be used alternatively. The LBG algorithm is an iterative competitive learning algorithm that, in essence, determines all points that fall within a Voronoi region around specified center vectors, calculates the mean of all points within this region, updates the center of this set to be equal to the mean, and then iterates the process until a fixed number of iterations has been met or some stopping criteria is achieved. Refer to Section 2.1 for more details. Proper initialization is a crucial issue for any iterative algorithm and can greatly affect the outcome. Therefore, we have chosen four different methods to initialize the center vectors for comparison purposes, requiring little supervision (if any) from the user.

The first method chosen to determine the center vectors used in the iterative LBG algorithm is LLEC. Here we use the LLEC algorithm to create a palette of colors for the data to be clustered around by identifying the natural subspaces of subimages of images within



(A) Original (B) $S = 13$ (C) $S = 24$ (D) $S = 92$

(E) Original (F) $S = 10$ (G) $S = 15$ (H) $S = 57$

(I) Original (J) $S = 5$ (K) $S = 8$ (L) $S = 21$

(M) Original (N) $S = 5$ (O) $S = 6$ (P) $S = 17$

(Q) Original (R) $S = 4$ (S) $S = 6$ (T) $S = 22$

FIGURE 4.5. Reconstruction images of LLEC with variances tolerances. Column two has tolerance $\epsilon_1 = \epsilon_2 = 0.6$. Column three has tolerance $\epsilon_1 = \epsilon_2 = 0.4$. Column four has tolerance $\epsilon_1 = \epsilon_2 = 0.2$. S denotes the number of distinct subspaces. The distortion errors from left to right, top to bottom are as follows: (Image 1) $DE = 4.2162 \times 10^3$, $DE = 2.2738 \times 10^3$, $DE = 764.723$, (Image 2) $DE = 2.0091 \times 10^3$, $DE = 1.4704 \times 10^3$, $DE = 580.750$, (Image 3) $DE = 2.74827 \times 10^3$, $DE = 2.0479 \times 10^3$, $DE = 650.323$, (Image 4)$DE = 2.6723 \times 10^3$, $DE = 1.6792 \times 10^3$, $DE = 623.756$, (Image 5)$DE = 1.9229 \times 10^3$, $DE = 1.0782 \times 10^3$, $DE = 537.293$.

the landscape data set, namely Figures 4.5i, 4.5m, and 4.5q. The benefit of this method is that all subspaces are identified in an unsupervised manner.

The next method involves choosing the eight three dimensional data points with components either 0 or 255 and 17 other data points sampled near the green, yellow, blue, white, and black colors as centers. This requires supervision from the user to identify which colors seem to predominantly appear in the data set of images.

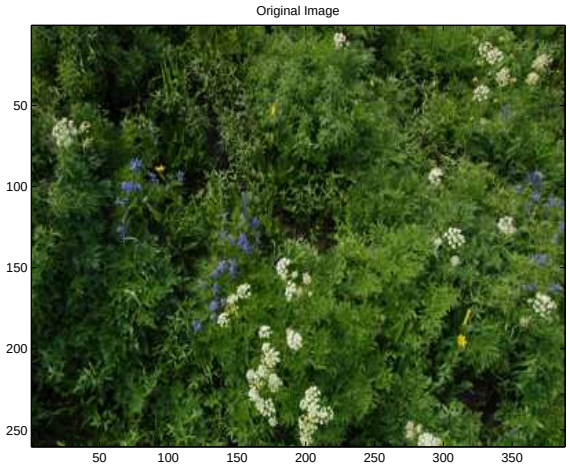
The third method involves choosing 25 random centers. That is 25 data points, $[r, g, b]$ are chosen such that $r, g, b \in [0, 255]$. Choosing random centers in this way does not guarantee that any of the colors identified to be centers will be similar to colors that appear in the actual image, and thus, many of the centers could be potentially unused in the clustering.

The final method involves choosing 25 random centers from the data set. That is, choose 25 columns of the data matrix X randomly to be the centers that the data points are clustered around. The benefit of this method is that this is the only approach that identifies actual points within the data set as centers. However, not all natural subspaces may be represented as we will observe shortly.

Figure 4.6 reveals the performance of each method on one sample image from the landscape data set. In several implementations on various images within the landscape data set, we have observed similar results. It appears that all methods for determining the centers result in fairly accurate reconstruction images. However, we have observed in practice that the two methods of using the LLEC algorithm to determine centers and identifying random centers within the data set tend to result in the lowest distortion errors as calculated by

$$D(X, J) = \frac{1}{p} \sum_{j \in J} \sum_{\mathbf{x} \in S_j} \|\mathbf{x} - \mathbf{c}_j\|^2$$

where X is a data set consisting of p points with regard to a set of centers labeled by indices J. Note that if we let X^* indicate the matrix of points each identified with the centroid



(A) Original

(B) LLEC

(C) Identifying Centers

(D) Random Centers

(E) Random Centers from Data

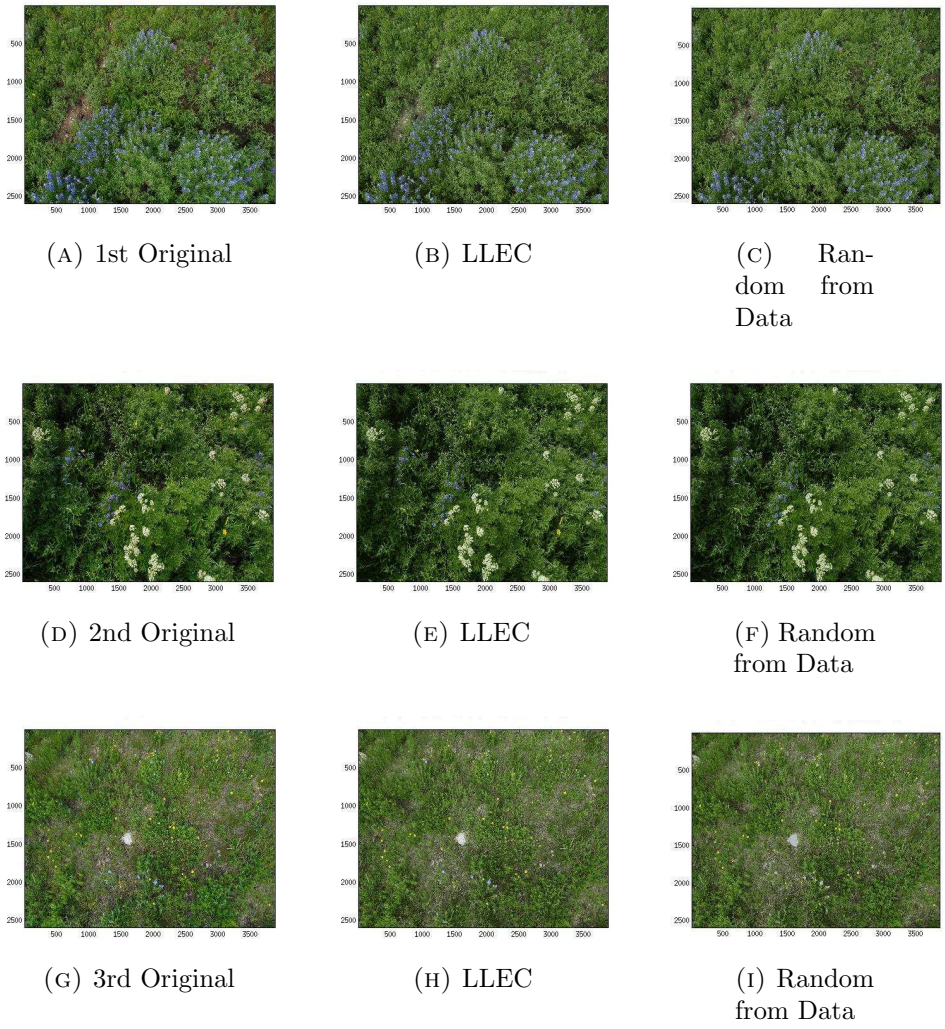
FIGURE 4.6. Reconstruction images after quantizing the color space of original image with LBG using indicated method to determine the centers. Note that the respective distortion errors of each implementation with 15 iterations are: 140.0250, 342.6351, 219.0756, and 146.7013.

of the Voronoi region that each point is assigned to, then the distortion error could also be calculated as $\|X - X^*\|_F^2$ where $\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$ is the Frobenius norm.

We notice in Figure 4.7 however that LLEC gives a better reconstruction visually. Observe that the reconstruction obtained by identifying centers as random points within the data often does not capture all subspaces within the image. In particular, the yellow flowers in the second example and the blue flowers in the third example do not appear in the reconstruction. Thus, it appears that LLEC used in conjunction with LBG is able to reconstruct the image with minimal error and the most accurate representation visually.

4.5. Conclusion

In this chapter, we have presented a novel algorithm, LLEC, to cluster and segment the color space of natural imagery. Within this algorithm, is a method to reconnect artificially disconnected components (resulting from a choice of K nearest neighbors) as well as a technique for one dimensional subspace segmentation that can be extended to multi-dimensional segmentation which is robust in the presence of noise. We have seen that LLEC does an excellent job of quantizing the color space of imagery with the only input parameters directly related to the accuracy of the quantization. However, LLEC does have some limitations. As already mentioned, LLEC is computationally intensive. Also, in the formulation of the LLE algorithm, it is required to create matrices of size $p \times p$, where p is the number of pixels in the image. For large images, this may require a prohibitively large amount of memory. Thus, LLE and LLEC, in turn, perform well on small sized images when being implemented in this manner. However, if techniques such as the sampling methods discussed in [59], [29] or the stitching method as discussed in [4] are implemented, this limitation may be alleviated.



(A) 1st Original (B) LLEC (C) Random from Data

(D) 2nd Original (E) LLEC (F) Random from Data

(G) 3rd Original (H) LLEC (I) Random from Data

FIGURE 4.7. Quantizing the color space of the original image with LBG using indicated method to determine the centers. Note that the respective distortion errors of these two implementations with 15 iterations are: (1st Original) 210.3490 and 210.6900, (2nd Original) 140.0250 and 146.7013, (3rd Original) 172.5580 and 170.7743.

Even with these limitations, we see that LLEC is useful in identifying the natural subspaces within an image.

CHAPTER 5

Sparse Locally Linear Embedding

5.1. Introduction

Several algorithms have been introduced with the purpose of reducing the dimension of a data set, as high dimensional data can often be represented appropriately in a lower dimensional space due to correlations and redundancies in the data. Various dimensionality reduction techniques include linear methods such as Principal Component Analysis [50] and Multidimensional Scaling [27] while nonlinear methods such as ISOMAP [75], Locally Linear Embedding (LLE) [71], Hessian Eigenmaps [31], and Laplacian Eigenmaps [9] have proven capable of extracting the underlying structure of real data which is typically nonlinear.

In this chapter, we will focus on nonlinear techniques, particularly the LLE algorithm discussed in detail in Chapter 2. The Locally Linear Embedding (LLE) algorithm [71] is an unsupervised dimensionality reduction algorithm that determines a mapping of data, lying in a higher dimensional vector space, to a lower dimensional vector space while optimizing the maintenance of local spatial relationships within the data. Through this map, the LLE algorithm uncovers a lower dimensional representation of the data with the goal of preserving the topology and neighborhood structure of the original higher dimensional data.

The primary free parameter of the LLE algorithm is K, the choice of nearest neighbors. This choice greatly affects the embedding results as it determines the local and global representation of the high-dimensional data in a lower-dimensional space. As discussed in Sections

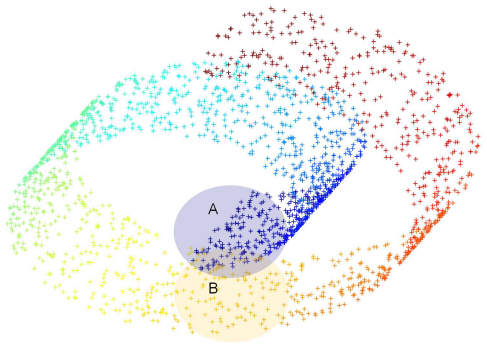


FIGURE 5.1. If K is too large, points in oval A could potentially be represented by nearest neighbors in oval B.

2.2.1 and 2.2.4.2, it is often difficult to appropriately choose K. If K is chosen too large, then the neighborhoods could no longer be locally linear, or points that are actually far away following the curvature of the manifold could be identified as nearest neighbors, as an example, see Figure 5.1. If K is chosen too small, then local patches are under-sampled and unable to appropriately preserve the topological structure of the data set as a lower-dimensional embedding. In the standard derivation of the algorithm, if K nearest neighbors are allowed, then each data point $\mathbf{x}_i$ will be represented by K nearest neighbors, i.e. typically all of the weights w_{ij} associated from data point $\mathbf{x}_i$ to each of its nearest neighbors $\mathbf{x}_j$ is nonzero.

Consider the following example of 2304 points residing in $\mathbb{R}^3$, see Figure 5.2. We observe that changing K has great influence on the embedding results, see Figure 5.3. Thus, finding an appropriate choice of K is necessary to determine the optimal embedding. As discussed in Section 2.2.4.2, a variety of methods have been proposed to optimally select K that are computationally intensive.

However, representing each data point uniformly with a choice of nearest neighbors K may not be appropriate for all data points (i.e. noise, isolated points, holes in the data, etc). The density and intrinsic dimensionality likely differ for the neighborhoods of each point $\mathbf{x}_i$

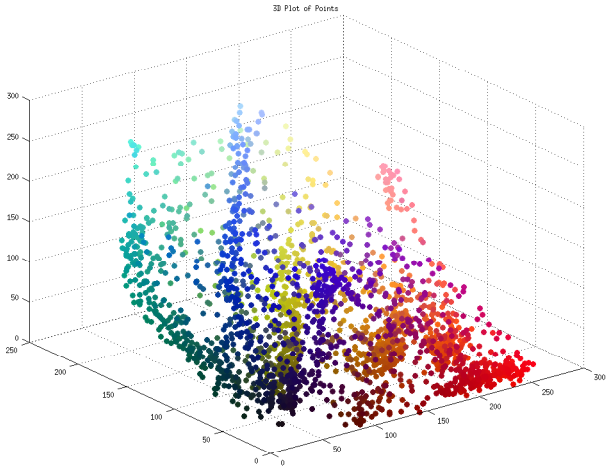
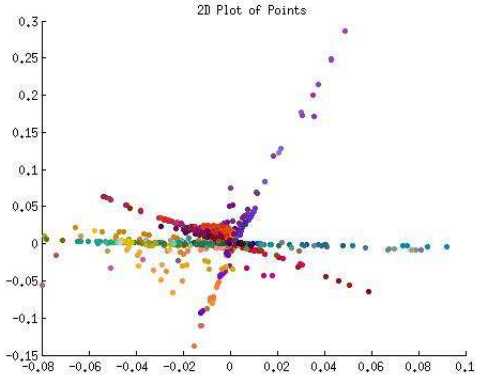
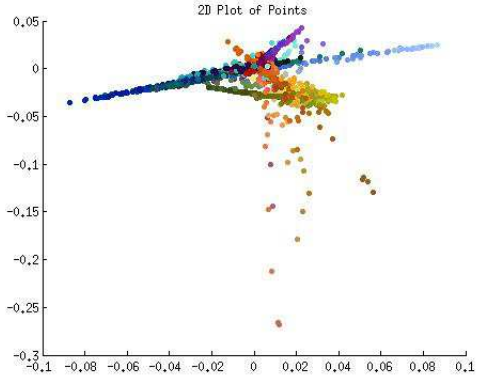


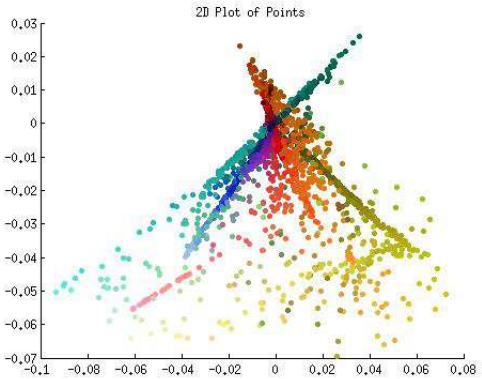
FIGURE 5.2. Original data in $\mathbb{R}^3$



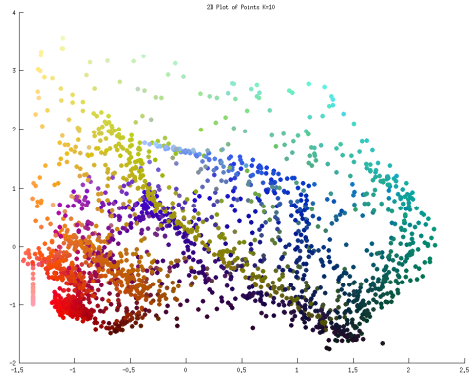
(A) $K = 4$



(B) $K = 6$



(C) $K = 8$



(D) $K = 10$

FIGURE 5.3. LLE reconstruction plots in $\mathbb{R}^2$ using various choices for the number of nearest neighbors K

[3], and thus, it seems that an appropriate number of nearest neighbors should be selected for each point instead of a single value of K chosen for all points.

In this chapter, we will consider allowing many more nearest neighbors than necessary and driving weights of unnecessary nearest neighbors–that actually are not very close to the data point being considered–to zero. Such a sparse representation of weights allows for identification of the 'true' nearest neighbors of each data point and a more appropriate local reconstruction. This method determines a neighborhood size K_i for each data point automatically. We call this method Sparse Locally Linear Embedding.

5.2. Sparse Locally Linear Embedding

Regularization using the ℓ_1 norm is becoming more common and understood in the data analysis community. It has been used in a variety of applications such as support vector machines, compressed sensing, image analysis, error correction, and matrix completion, and it is known to produce sparsity in decision variables of optimization problems when used as a regularization term in the objective function [11], [16], [19], [17], [18], [32], [36], [40], [64], [76].

Recall in standard LLE, the weights w_{ij}–associated from the point in consideration $\mathbf{x}_i$ to each of its nearest neighbors $\mathbf{x}_j$–are determined by solving the following optimization problem

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \sum_{i=1}^{p} \|\mathbf{x}_i - \sum_{j \in N_i} w_{ij}\mathbf{x}_j\|_2^2 \\ \text{subject to} \quad & \sum_{j \in N_i} w_j = \ 1 \end{aligned}$$

As indicated before, we would like to induce sparsity in the weights associated to nearest neighbors. We do so by including an ℓ_1 regularization term in the objective function to determine the weights.

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \lambda \sum_{i=1}^{p} \sum_{j \in N_i} |w_{ij}| f(d(\mathbf{x}_i, \mathbf{x}_j)) + \sum_{i=1}^{p} \|\mathbf{x}_i - \sum_{j \in N_i} w_{ij} \mathbf{x}_j\|_2^2 \\ \text{subject to} \quad & \sum_{j \in N_i} w_j = \ 1 \end{aligned} \tag{5.1}$$

We have included λ, a parameter to enforce the importance of driving weights of nearest neighbors to zero, as well as a data-weighted scaling where f is some nonnegative function of the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ as determined by some metric such as the p-norm,

$$||\mathbf{x}||_p = \left(\sum_{k=1}^{n} |x_k|^p \right)^{\frac{1}{p}} \text{ where } \mathbf{x}_i \in \mathbb{R}^n.$$

We have observed that if f is omitted sparsity does not seem to be induced. At the end of this section are suggestions for choosing this nonnegative function.

Recall that standard LLE actually solves a least squares problem for each point

$$C_i \tilde{\mathbf{w}} = \mathbf{e}$$

as described in Section 2.2.2 where C_i corresponds to the covariance matrix determined by $c^i_{jk} = (\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_k)$ (with j, k indices of nearest neighbors to $\mathbf{x}_i$, $\tilde{\mathbf{w}}$ is the column vector of weights associated to a single point, and $\mathbf{e}$ is the vector of all ones. Therefore, instead of solving the potentially very large problem 5.1 once, we may think of this as a much smaller,

decoupled problem

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \lambda \sum_{j \in N_i} |w_j| f(d(\mathbf{x}_i, \mathbf{x}_j)) + \|\mathbf{x}_i - \sum_{j \in N_i} w_j \mathbf{x}_j\|_2^2 \\ \text{subject to} \quad & \sum_{j \in N_i} w_j = 1 \end{aligned} \tag{5.2}$$

to be solved for each data point $\mathbf{x}_i$. Solving all of these problems can be trivially parallelized.

The absolute value in optimization problem 5.2 provides a bit of difficulty. However, as indicated in [13], we may reformulate our optimization problem in two manners. Let us consider an optimization problem of the form

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \sum_{i=1}^{n} c_i |x_i| \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{b} \end{aligned}$$

with nonnegative c_i and $\mathbf{x} \in \mathbb{R}^n$. First, notice that $|x_i|$ satisfies $x_i \leq z_i$ and $-x_i \leq z_i$ for some z_i. Therefore, our problem can be reformulated as

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \sum_{i=1}^{n} c_i z_i \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{b} \\ & x_i \leq z_i \text{ for } i = 1, \ldots, n \\ & -x_i \leq z_i \text{ for } i = 1, \ldots, n \end{aligned}$$

As a second approach, we may introduce new variables x_i^+ and x_i^-, required to be nonnegative, and rewrite our problem using $x_i = x_i^+ - x_i^-$ and $|x_i| = x_i^+ + x_i^-$

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \sum_{i=1}^{n} c_i(x_i^+ - x_i^-) \\ \text{subject to} \quad & A\mathbf{x}^+ - A\mathbf{x}^- = \mathbf{b} \\ & \mathbf{x}^+ \geq \mathbf{0} \\ & \mathbf{x}^- \geq \mathbf{0} \end{aligned}$$

where $\mathbf{x}^+$ is the vector of x_i^+ entries and $\mathbf{x}^-$ is the vector of x_i^- entries. The goal is to have $x_i = x_i^+$ or $x_i = -x_i^-$, and thus, either x_i^+ or x_i^- equals 0. We see this by considering the nonnegativity of c_i. Suppose that $c_i > 0$ for all i for simplicity, otherwise the problem is degenerate. Assume that optimal solutions $x_i^+ > 0$ and $x_i^- > 0$. We reduce x_i^+ and x_i^- by the same amount r,

$$\begin{aligned} A_i(x_i^+ - r) - A_i(x_i^- - r) &= b_i \\ A_i x_i^+ - A_i r - A_i x_i^- + A_i r &= b_i \\ A_i x_i^+ - A_i x_i^- &= b_i \end{aligned}$$

Thus, the problem still remains feasible. Note that A_i denotes the ith row of matrix A and b_i denotes the ith entry of vector b. Since $c_i > 0$, reducing x_i^+ and x_i^- by r will in turn reduce the cost of the objective function, contradicting optimality of x_i^+ and x_i^-. Thus, either $x_i^+ = 0$ or $x_i^- = 0$. This is true for any $c_i > 0$, $i = 1, \ldots, n$. Notice that either reformulation requires adding $2n$ constraints and n variables.

Now, let us return to our optimization problem 5.2. We will use the latter approach described above to remove the absolute value from our objective function by introducing nonnegative variables w_j^+ and w_j^- such that $w_j = w_j^+ - w_j^-$ and $|w_j| = w_j^+ + w_j^-$

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \lambda \sum_{j \in N_i} (w_j^+ + w_j^-) f(d(\mathbf{x}_i, \mathbf{x}_j)) + \|\mathbf{x}_i - \sum_{j \in N_i} (w_j^+ - w_j^-)\mathbf{x}_j\|_2^2 \\ \text{subject to} \quad & \sum_{j \in N_i} (w_j^+ - w_j^-) = 1 \\ & w_j^+, w_j^- \geq 0 \end{aligned} \tag{5.3}$$

We rewrite our objective function as

$$\begin{aligned} & \lambda \sum_{j \in N_i} (w_j^+ + w_j^-) f(d(\mathbf{x}_i, \mathbf{x}_j)) + \|\mathbf{x}_i - \sum_{j \in N_i} (w_j^+ - w_j^-)\mathbf{x}_j\|_2^2 \\ & = \lambda \sum_{j \in N_i} (w_j^+ + w_j^-) f(d(\mathbf{x}_i, \mathbf{x}_j)) + < \mathbf{x}_i - \sum_{j \in N_i} (w_j^+ - w_j^-)\mathbf{x}_j, \mathbf{x}_i - \sum_{j \in N_i} (w_j^+ - w_j^-)\mathbf{x}_j > \\ & = \lambda \sum_{j \in N_i} (w_j^+ + w_j^-) f(d(\mathbf{x}_i, \mathbf{x}_j)) + \mathbf{x}_i^T \mathbf{x}_i - 2 \sum_{j \in N_i} (w_j^+ - w_j^-)\mathbf{x}_i^T \mathbf{x}_j + \\ & \qquad \sum_{j,k \in N_i} (w_j^+ - w_j^-)(w_k^+ - w_k^-)\mathbf{x}_j^T \mathbf{x}_k \\ & = \sum_{j,k \in N_i} (w_j^+ w_k^+ - w_j^+ w_k^- - w_j^- w_k^+ + w_j^- w_k^-)\mathbf{x}_j^T \mathbf{x}_k - 2 \sum_{j \in N_i} (w_j^+ - w_j^-)\mathbf{x}_i^T \mathbf{x}_j + \\ & \qquad \lambda \sum_{j \in N_i} (w_j^+ + w_j^-) f(d(\mathbf{x}_i, \mathbf{x}_j)) + \mathbf{x}_i^T \mathbf{x}_i \\ & = \sum_{j,k \in N_i} \mathbf{x}_j^T \mathbf{x}_k (w_j^+ w_k^+ - w_j^+ w_k^- - w_j^- w_k^+ + w_j^- w_k^-) \qquad \text{(Quadratic)} \\ & \qquad + \sum_{j \in N_i} \left(-2\mathbf{x}_i^T \mathbf{x}_j + \lambda f(d(\mathbf{x}_i, \mathbf{x}_j)) \right) (w_j^+ - w_j^-) \qquad \text{(Linear)} \\ & \qquad + \mathbf{x}_i^T \mathbf{x}_i \qquad \text{(Constant)} \end{aligned}$$

This optimization problem can be rewritten as a quadratic program of the form

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{w}^T Q\mathbf{w} + \mathbf{c}^T\mathbf{w} \\ \text{subject to} \quad & A\mathbf{w} = \mathbf{b} \\ & \mathbf{w} \geq 0 \end{aligned} \tag{5.4}$$

Note, removing the constant term will not affect the optimal solution to this problem. The optimization problem 5.4 is in $n = 2K$ variables with equality constraint, $A\mathbf{w} = \mathbf{b}$, an $m \times n$ system where $m = 1$. More explicity,

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}^+ \\ \mathbf{w}^- \end{bmatrix} = \begin{bmatrix} w_i^{(1)+} \\ w_i^{(2)+} \\ \cdots \\ w_i^{(K)+} \\ w_i^{(1)-} \\ w_i^{(2)-} \\ \cdots \\ w_i^{(K)-} \end{bmatrix}$$

where we use the notation $w_i^{(\ell)}$ to denote the weight associated from point x_i to its ℓth nearest neighbor for $\ell = 1, \ldots, K$. Then, $Q = Q_i$ for each i with

$$Q_i = \begin{bmatrix} \tilde{Q}_i & -\tilde{Q}_i \\ -\tilde{Q}_i & \tilde{Q}_i \end{bmatrix}$$

where

$$\tilde{Q}_i = \begin{bmatrix} \mathbf{x}_i^{(1)T}\mathbf{x}_i^{(1)} & \mathbf{x}_i^{(1)T}\mathbf{x}_i^{(2)} & \dots & \mathbf{x}_i^{(1)T}\mathbf{x}_i^{(K)} \\ \vdots & \ddots & & \vdots \\ \mathbf{x}_i^{(K)T}\mathbf{x}_i^{(1)} & \mathbf{x}_i^{(K)T}\mathbf{x}_i^{(2)} & \dots & \mathbf{x}_i^{(K)T}\mathbf{x}_i^{(K)} \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & \dots & 1 & -1 & \dots & -1 \end{bmatrix}$$

$$\mathbf{b} = 1$$

$$\mathbf{c} = \begin{bmatrix} -2\mathbf{x}_i^T\mathbf{x}_i^{(1)} + \lambda f(d(\mathbf{x}_i, \mathbf{x}_j)) \\ -2\mathbf{x}_i^T\mathbf{x}_i^{(2)} + \lambda f(d(\mathbf{x}_i, \mathbf{x}_j)) \\ \vdots \\ -2\mathbf{x}_i^T\mathbf{x}_i^{(K)} + \lambda f(d(\mathbf{x}_i, \mathbf{x}_j)) \\ 2\mathbf{x}_i^T\mathbf{x}_i^{(1)} + \lambda f(d(\mathbf{x}_i, \mathbf{x}_j)) \\ \vdots \\ 2\mathbf{x}_i^T\mathbf{x}_i^{(K)} + \lambda f(d(\mathbf{x}_i, \mathbf{x}_j)) \end{bmatrix}$$

We have not explicitly defined the nonnegative function $f(d(\mathbf{x}_i, \mathbf{x}_j))$ which is a coefficient on the linear weight terms. This allows more flexibility for the user and allows tuning of the algorithm depending on data sets. Possible obvious functions to consider could include

$$f(d(\mathbf{x}_i, \mathbf{x}_j)) = ||\mathbf{x}_i - \mathbf{x}_j||_2^2$$

$$f(d(\mathbf{x}_i, \mathbf{x}_j)) = ||\mathbf{x}_i - \mathbf{x}_j||_2$$

$$f(d(\mathbf{x}_i, \mathbf{x}_j)) = ||\mathbf{x}_i - \mathbf{x}_j||_1$$

$$f(d(\mathbf{x}_i, \mathbf{x}_j)) = \frac{1}{||\mathbf{x}_i - \mathbf{x}_j||_2^2}$$

Note the first three functions favor the closest nearest neighbors while the final function penalizes these nearest neighbors. In practice, we have typically used the first or second function. Another possibility could be a unimodal function that favors the closest nearest neighbors and those that are farthest away, but penalizes nearest neighbors with mid-range distances. This would enforce sparsity but capture more global information. Various choices for this function will yield different, and possibly more appropriate results depending on the application.

In the next section, we will explore when this problem (or any quadratic program for that matter) has a solution.

5.3. Convex Optimization Problems

First, we need a few definitions and results regarding convexity in order to determine when a quadratic program has a solution. Let us observe the definitions of a convex set and function.

Definition 3. *A set C is a **convex set** if the line segment between any two points in C lies in C, i.e., if for any x_1, $x_2 \in C$ and any θ with $0 \leq \theta \leq 1$, we have*

$$\theta x_1 + (1 - \theta)x_2 \in C.$$

Definition 4. *A function $f : \mathbb{R}^n \to \mathbb{R}$ is a **convex function** if the domain of f is a convex set and if for any two points x and y in the domain of f, and any θ with $0 \leq \theta \leq 1$, we have*

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y).$$

Next, we need the definition of a positive semi-definite matrix.

Definition 5. *A symmetric matrix H is **positive semi-definite** if $\boldsymbol{x}^T H \boldsymbol{x} \geq 0$ for all $x \neq 0$. We write this as $H \succeq 0$.*

Now, when is a quadratic function convex? We will use the second-order condition described in [15]. The second-order condition is as follows:

Theorem 1 (Second Order Condition)**.** *Suppose f is twice differentiable. Then f is convex if and only if the domain of f is convex and its Hessian is positive semi-definite, i.e.*

$$\nabla^2 f(x) \succeq 0.$$

We must consider the Hessian of a quadratic function $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + c^T\mathbf{x}$. The gradient is

$$\nabla f(\mathbf{x}) = Qx + c$$

and the Hessian is

$$\nabla^2 f(\mathbf{x}) = Q.$$

Therefore, a quadratic function is convex if and only if Q is positive semi-definite (and concave if and only if Q is negative semi-definite). Now, we need to consider the definition of a convex optimization problem

Definition 6. *A **convex optimization problem** is one of the form*

$$\begin{aligned} &\textit{minimize} && f_0(\boldsymbol{x}) \\ &\textit{subject to} && f_i(\boldsymbol{x}) \leq 0 \textit{ for } i = 1, \ldots, m \\ &&& a_j^T \boldsymbol{x} = b_j \textit{ for } j = 1, \ldots, p \end{aligned} \tag{5.5}$$

where the $f_0, \ldots, f_m$ are convex functions.

Note that the equality constraints are required to be affine. Therefore, a quadratic program like 5.4 is convex if and only if Q is positive semi-definite. One of the key features of a convex optimization problem is that any local optimum is also a global optimum [15].

Let us now return to our problem described in the previous section with $Q = Q_i$ for each i defined as

$$Q_i = \begin{bmatrix} \tilde{Q}_i & -\tilde{Q}_i \\ -\tilde{Q}_i & \tilde{Q}_i \end{bmatrix}$$

where

$$\tilde{Q}_i = \begin{bmatrix} \mathbf{x}_i^{(1)T}\mathbf{x}_i^{(1)} & \mathbf{x}_i^{(1)T}\mathbf{x}_i^{(2)} & \ldots & \mathbf{x}_i^{(1)T}\mathbf{x}_i^{(K)} \\ \vdots & \ddots & & \vdots \\ \mathbf{x}_i^{(K)T}\mathbf{x}_i^{(1)} & \mathbf{x}_i^{(K)T}\mathbf{x}_i^{(2)} & \ldots & \mathbf{x}_i^{(K)T}\mathbf{x}_i^{(K)} \end{bmatrix}.$$

for each i. Notice that Q_i can be written as an outer product

$$Q_i = \mathbf{a}_i\mathbf{a}_i^T$$

where

$$\mathbf{a}_i = \begin{bmatrix} - & (\mathbf{x}_i^{(1)})^T & - \\ - & (\mathbf{x}_i^{(2)})^T & - \\ & \vdots & \\ - & (\mathbf{x}_i^{(K)})^T & - \\ - & -(\mathbf{x}_i^{(1)})^T & - \\ - & -(\mathbf{x}_i^{(2)})^T & - \\ & \vdots & \\ - & -(\mathbf{x}_i^{(K)})^t & - \end{bmatrix}.$$

Finally, notice that any matrix written as an outer product is positive semi-definite as

$$\mathbf{x}^T Q \mathbf{x} = \mathbf{x}^T \mathbf{a}\mathbf{a}^T \mathbf{x} = ||\mathbf{a}^T \mathbf{x}||_2^2 \geq 0$$

for all $x \neq 0$. Therefore, the problem in consideration is a convex optimization problem.

5.4. Solving Sparse LLE

As mentioned previously, determining the global optimal solutions can be achieved by finding local solutions. Thus, by forming the Lagrangian of our optimization problem, computing the gradiant and equating to zero (the first order optimality conditions), and solving for the roots using some approach such as Newton's Method, we may find not only the local optimum(s) of this program but also the global optimum(s). Note if Q is positive definite (and thus full rank) the solution is unique [15]. We will see in Chapter 6 one possible method to solve our convex quadratic optimization problem known as the Primal Dual Interior Point (PDIP) algorithm. Another method, which we will briefly introduce here, is the Split Bregman Algorithm as discussed in [64], [69], [40].

The Split Bregman Algorithm can solve problems of the form

$$\begin{aligned} &\underset{\mathbf{u}}{\text{minimize}} && E(\mathbf{u}) \\ &\text{subject to} && H(\mathbf{u}) = \mathbf{0} \end{aligned} \tag{5.6}$$

where $\mathbf{u} \in \mathbb{R}^n$, E and H are both convex, and H is differentiable. As E is not required to be differentiable, this algorithm can solve ℓ_1-regularized problems directly. We convert this problem to one that is unconstrained by

$$\underset{\mathbf{u}}{\text{minimize}} \qquad E(\mathbf{u}) + \lambda H(\mathbf{u}) \tag{5.7}$$

Typically, this algorithm solves problems of the form

$$\begin{aligned} E(\mathbf{u}, \mathbf{d}) &= ||\mathbf{d}||_1 + F(\mathbf{u}) \text{ and} \\ H(\mathbf{u}, \mathbf{d}) &= ||\mathbf{d} - \Phi(\mathbf{u})||_2^2, \text{ i.e. } \mathbf{d} = \Phi(\mathbf{u}) \end{aligned}$$

for a differentiable function Φ. [40] suggests solving this problem iteratively for $\mathbf{u}$ and $\mathbf{d}$ by splitting the differentiable and the non-differentiable portions.

In order for us to use the Split Bregman formula, we need to modify our problem 5.1 slightly. Note, that since the Split Bregman algorithm can handle ℓ_1 regularization terms directly, we do not need to use optimization techniques as discussed above to remove this non-differentiable term from our objective function. For each data point $\mathbf{x}_i$, form

- F, the diagonal matrix with entries given by a nonnegative function f of distances between $\mathbf{x}_i$ and each of its K nearest neighbors, i.e. $f(d(\mathbf{x}_i, \mathbf{x}_i^{(\ell)})$ for $\ell = 1, \ldots, K$

- N_i the $D \times K$ matrix of nearest neighbors to point $\mathbf{x}_i$ with the points written as columns

Then, our decoupled problem can be reformulated as

$$\underset{\mathbf{w}}{\text{minimize}} \quad \lambda_1 ||F\mathbf{w}||_1 + \frac{\lambda_2}{2}||\mathbf{x}_i - N_i\mathbf{w}||_2^2 + \frac{\lambda_3}{2}(\mathbf{e}^T\mathbf{w} - 1)^2$$

where we are solving for the weights associated to the single point $\mathbf{x}_i$. Written in the Split Bregman formulation, we have

$$\underset{\mathbf{w,d,b}}{\text{minimize}} \quad \lambda_1 ||\mathbf{d}||_1 + \frac{\lambda_2}{2}||\mathbf{x}_i - N_i\mathbf{w}||_2^2 + \frac{\lambda_3}{2}(\mathbf{e}^T\mathbf{w} - 1)^2 + \frac{\lambda_4}{2}||\mathbf{d} - F\mathbf{w}||_2^2$$

which can be simplified by using the suggestions given in [40] as

$$(\mathbf{w}_{k+1}, \mathbf{d}_{k+1}) = \underset{\mathbf{w,d}}{\text{argmin}} \quad \lambda_1 ||\mathbf{d}||_1 + \frac{\lambda_2}{2}||\mathbf{x}_i - N_i\mathbf{w}||_2^2 + \frac{\lambda_3}{2}(\mathbf{e}^T\mathbf{w} - 1)^2 + \frac{\lambda_4}{2}||\mathbf{d} - F\mathbf{w} - \mathbf{b}_k||_2^2.$$

We iteratively solve for w_{k+1}, d_{k+1}, and b_{k+1} by using the following formulas

$$\begin{aligned}
\mathbf{w}_{k+1} &= \underset{\mathbf{w}}{\text{argmin}} \frac{\lambda_2}{2}||\mathbf{x}_i - N_i\mathbf{w}||_2^2 + \frac{\lambda_3}{2}(\mathbf{e}^T\mathbf{w} - 1)^2 + \frac{\lambda_4}{2}||\mathbf{d}_k - F\mathbf{w} - \mathbf{b}_k||_2^2 \\
\mathbf{d}_{k+1} &= \underset{\mathbf{d}}{\text{argmin}} \lambda_1 ||\mathbf{d}||_1 + \frac{\lambda_4}{2}||\mathbf{d} - F\mathbf{w} - \mathbf{b}_k||_2^2 \\
\mathbf{b}_{k+1} &= b_k + F\mathbf{w}_{k+1} - \mathbf{d}_{k+1}
\end{aligned}$$

Notice that there are 4 parameters in this derivation, λ_1 is associated with the ℓ_1 regularization, λ_2 is associated with the reconstruction error, λ_3 is attached to the sum to one criteria for the weights, and λ_4 is associated with the Bregman restriction. At optimality, the last two terms should be zero, the reconstruction error should be zero in the case when

$K > D$ and as small as possible otherwise, and thus, varying these last three parameters does not affect the solution greatly. However, varying the parameter λ_1 affecting the emphasis placed on inducing sparsity, much like varying λ in Equation 5.2, does change the optimal solution. We will see the affect of varying λ (in Equation 5.2) in the implementation sections.

As the Split Bregman algorithm is not our focus, we will not spend more time discussing precisely how to solve each step. We choose to use the PDIP algorithm to solve our optimization problem. Do note though that the computational complexity to solve problem 5.2 for all points $\mathbf{x}_i$, $i = 1, \ldots, p$, using Split Bregman is $\mathcal{O}(pK^3) + \mathcal{O}(ph(K^3 + KD^2 + K + 2K^3))$ where p is the number of data points, K is the number of nearest neighbors allowed, D is the dimension of our input data, and h is the average number of iterations to achieve a desired tolerance. The first term comes from an inverse computation while the second is matrix multiplication and addition.

5.5. The Algorithm

We summarize Sparse Locally Linear Embedding in the following Algorithm:

Sparse LLE Algorithm

(1) Determine a maximum value of K nearest neighbors to be allowed. Find the K nearest neighbors to each point $\mathbf{x}_i$ as ascertained by some metric, usually Euclidean distance.

(2) Find the sparse weights used to write each point as a linear combination of its nearest neighbors $\mathbf{x}_j$ by solving

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \lambda \sum_{j \in N_i} |w_j| f(d(\mathbf{x}_i, \mathbf{x}_j)) + \|\mathbf{x}_i - \sum_{j \in N_i} w_j \mathbf{x}_j\|_2^2 \\ \text{subject to} \quad & \sum_{j \in N_i} w_j = 1 \end{aligned}$$

for a given λ, typically very small, and each point $\mathbf{x}_i$.

(3) Zero out any weight value $w_{ij} < tol$, where tol is some tolerance such as 10^{-4} or possibly smaller. The number of nonzero weights determines, K_i, the number of nearest neighbors used to reconstruct point $\mathbf{x}_i$.

(4) Determine lower dimensional embedding vectors by minimizing

$$\phi(Y) = \sum_{i=1}^{p} \|\mathbf{y}_i - \sum_{j \in N_i} w_{ij} \mathbf{y}_j\|_2^2$$

subject to $YY^T = I$ and $\sum_{i=1}^{p} \mathbf{y}_i = \mathbf{0}$ as a sparse eigensolver.

In the remainder of this chapter we will apply Sparse LLE to a variety of data sets, observing its ability to automatically select an appropriate number of nearest neighbors for each data point and its ability to preserve the topological structure from the high dimensional space.

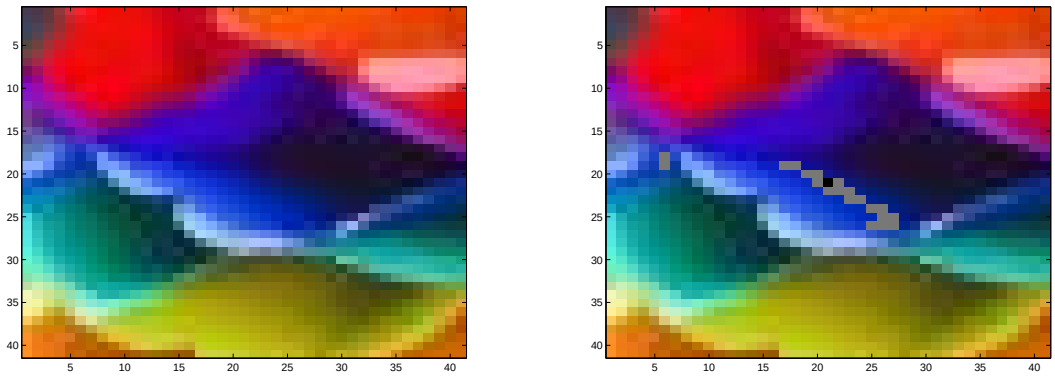
5.6. Sparsity Example

Let us consider the 3-dimensional RGB data from a square 41×41 color image, Figure 5.4a. For this example, we are primarily concerned with the sparsity induced on the weights

using Sparse LLE. This sparsity automatically selects an appropriate number of nearest neighbors K_i for each data point $\mathbf{x}_i$. We will focus on the central pixel within the image. Allowing $K = 20$ nearest neighbors–which is certainly more than necessary to represent this data in $\mathbb{R}^3$–we use a function of $f(d(\mathbf{x}_i, \mathbf{x}_j)) = ||\mathbf{x}_i - \mathbf{x}_j||_2$ and vary our parameter λ that enforces the importance of the sparsity term. Using a choice of $\lambda = 0$, our optimization problem is the same as the standard LLE problem to determine weights, and thus, all weights associated to the central pixel are nonzero. As we increase λ, sparsity is induced in the weights. Refer to Figure 5.4. Note in Figure 5.4b we see all pixels selected to be nearest neighbors to the central pixel, indicated as gray pixels. We use gray to denote those pixels associated to nonzero weights and white to denote zero weights, generated by increasing λ, Figures 5.4c and 5.4d. It is important to realize that these nearest neighbors are near in the spectral sense not the spatial sense.

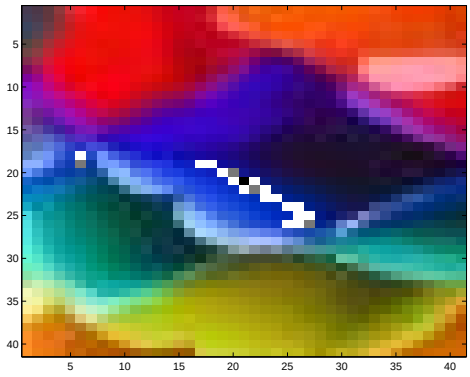
To observe the numerical values of the 20 weights associated to the central pixel for these three choices of parameter λ, see Figure 5.5. We see that as we increase λ, the number of nonzero weights dramatically decreases, Figure 5.6. As this data is in $\mathbb{R}^3$, it makes sense that 3 or 4 nearest neighbors would be necessary to reconstruct each data point, remembering that the weights associated to these nearest neighbors must sum to 1.

Our objective function in 5.2 involves two terms–the first we call the sparsity term, and the second is the reconstruction error. As more emphasis is placed on the first term, there is a deemphasis on minimizing the second term. Thus, there is a balancing act between the two terms. We notice as λ is increased (and the importance of the sparsity term is increased) the reconstruction error increases, as expected. See Figure 5.7. Finally, we see this balancing act explicitly in the pareto optimal curve that compares the value of which we

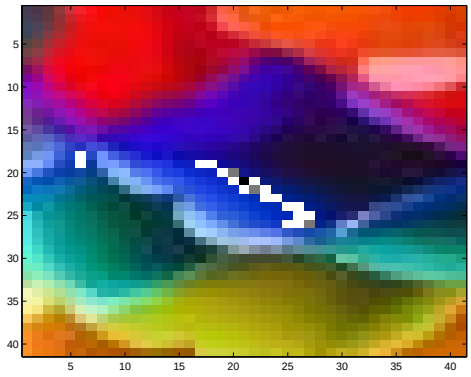


(A) Original Image

(B) $\lambda = 0$



(C) $\lambda = 0.02$



(D) $\lambda = 0.1$

FIGURE 5.4. Considering weights associated to the central (black) pixel. Using $K = 20$. Nearest neighbor pixels associated to nonzero weights are colored gray and those associated to zero weights are colored white.

will call the sparsity term the risk, $\sum_{j \in N_i} |w_j| f(d(\mathbf{x}_i, \mathbf{x}_j))$, and we will call the reconstruction error the reward $\|\mathbf{x}_i - \sum_{j \in N_i} w_j \mathbf{x}_j\|_2^2$, Figure 5.8. From all of these figures we can determine that increasing λ by a small amount will induce sparsity in the weights associated to nearest neighbors. However, we do not want to increase λ so much that the reconstruction error is large, and thus, each point is poorly reconstructed as a linear combination of its nearest neighbors.

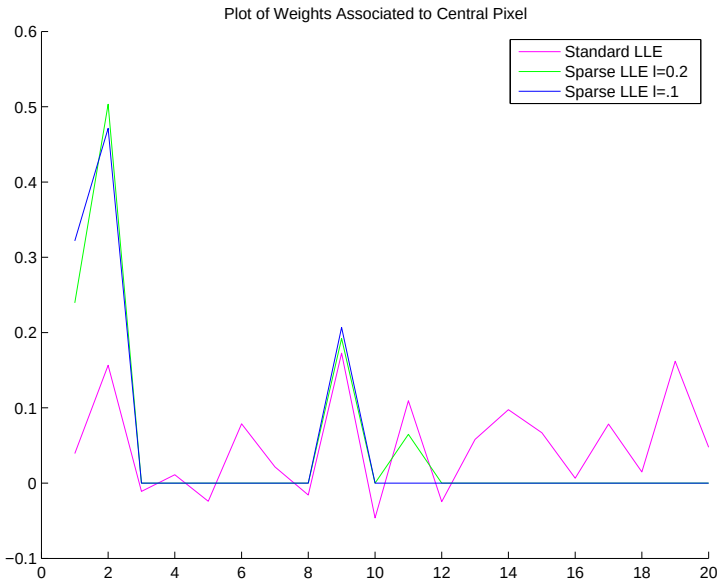


FIGURE 5.5. Plot of weights associated to central pixel, allowing $K = 20$ nearest neighbors and varying λ.

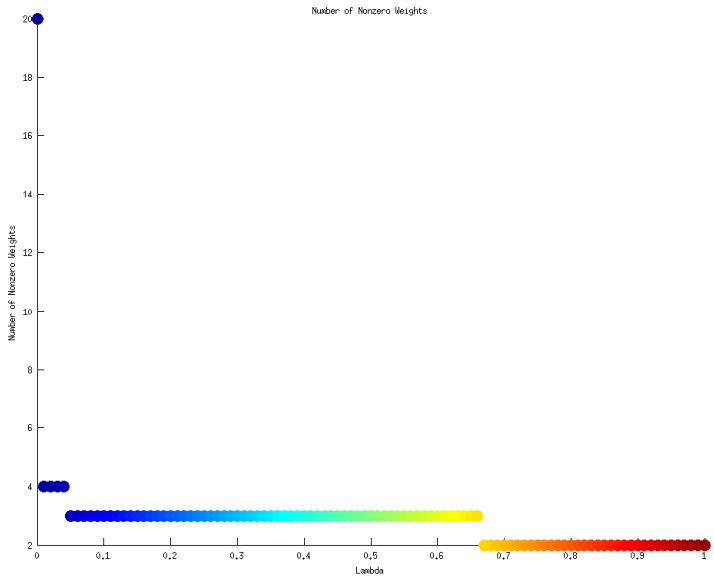


FIGURE 5.6. Plot of the number of nonzero weights associated to nearest neighbors of the central pixel versus λ.

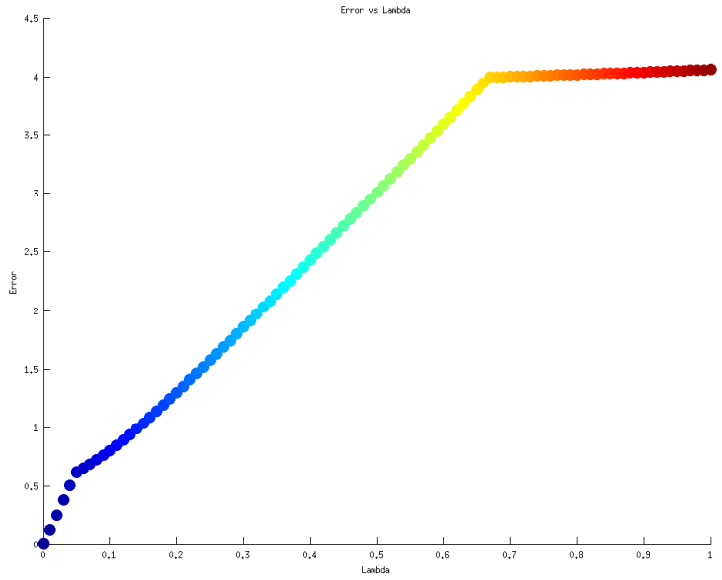


FIGURE 5.7. Plot of the reconstruction error versus λ.

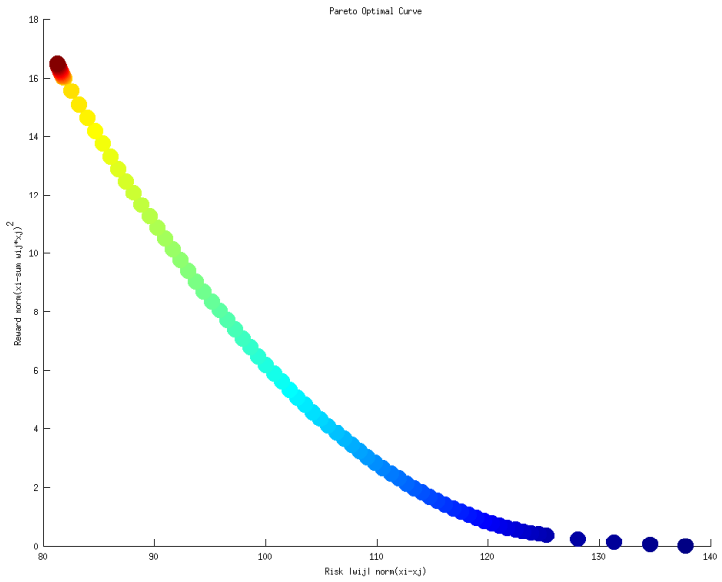
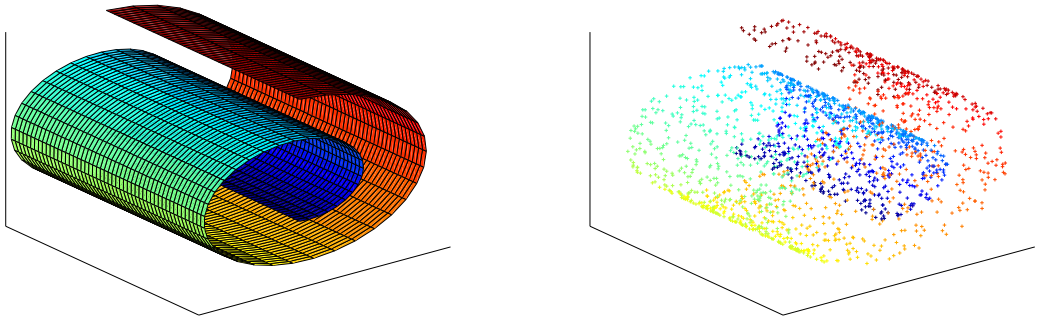


FIGURE 5.8. Plot of the sparsity term, the 'risk', $\sum_{j \in N_i} |w_j| f(d(\mathbf{x}_i, \mathbf{x}_j))$ versus the reconstruction error, the 'reward', $\|\mathbf{x}_i - \sum_{j \in N_i} w_j \mathbf{x}_j\|_2^2$. Note that blue corresponds to smaller λ values while red corresponds to larger λ values.



(A) Swiss Roll (B) Sampled Swiss Roll

FIGURE 5.9. 2000 randomly generated points along the swiss roll in $\mathbb{R}^3$.

5.7. Swiss Roll Example

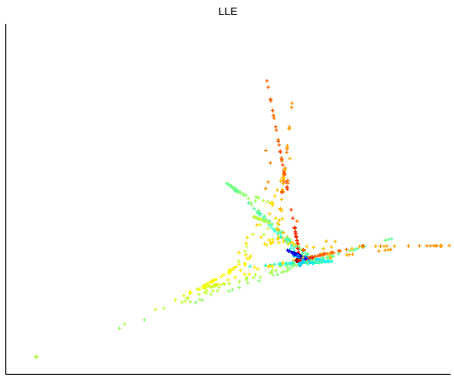
We will now consider the canonical swiss roll example found in many nonlinear dimensionality reduction texts [71], [3], [9]. The topological structure of this data set will not be captured with a linear dimensionality reduction technique such as PCA or MDS as it is highly nonlinear, but as we will see, it is able to be captured using the nonlinear technique of Sparse LLE. This data set consists of 2000 randomly generated points along the swiss roll in $\mathbb{R}^3$, Figure 5.9.

As this data is 3-dimensional it might be supposed that $K = 4$ would be an appropriate choice of nearest neighbors. However, using standard LLE with a choice of $K = 4$ nearest neighbors, the lower dimensional embedding in $\mathbb{R}^2$ is quite poor and does not reflect the topological structure in the slightest, Figure 5.10a. In fact, this choice of nearest neighbors disconnects the data, as evidenced by the eigenvalues of the Laplacian matrix M defined in Section 2.2.3 and further discussed in 4.2.2. From the source code on the Locally Linear Embedding homepage to generate the swiss roll, $K = 12$ nearest neighbors were selected [1]. We see that this choice of K yields a much better embedding, Figure 5.10b. Now, we

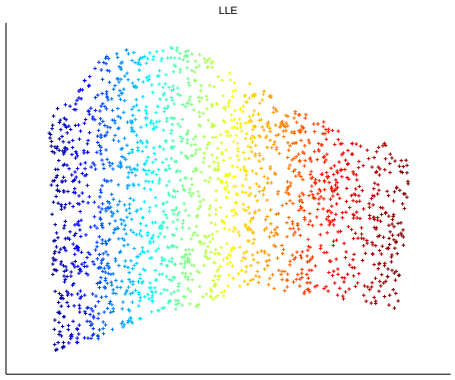
illustrate how Sparse LLE automatically determines an appropriate choice of K_i for each point $\mathbf{x}_i$ without needing to 'tune'. Let us use the Sparse LLE algorithm with $K = 20$ nearest neighbors allowed (although as we will see in the next example this is an arbitrary choice that does not affect the embedding if λ is in an appropriate range) with $\lambda = 0.01$, Figure 5.10. Note that Figure 5.10c was constructed using the full $K \times p$ matrix W in the eigenvector problem to determine the embedding vectors while 5.10d was generated by zeroing out those entries less than 10^{-4} in W and using the remaining 'nonzero' entries in W in the eigenvector problem. Recall that the weight problem is invariant to rotations, rescalings, and translations. Now, we consider a histogram for the number of nonzero weights associated to each point using both standard LLE and Sparse LLE, Figure 5.11. Note in both cases we consider weights $w_{ij} \geq 10^{-4}$ to be nonzero. Notice that nearly all of the weights in standard LLE are nonzero as $K = 12$ was selected, while only 2-6 weights are nonzero for each data point in Sparse LLE even though $K = 20$ nearest neighbors were allowed.

Finally, let us consider the average number of nonzero weights associated to nearest neighbors of data points sampled from the swiss roll as we vary λ, the parameter that affects sparsity in Sparse LLE, as well as K, see Figure 5.12a. Figure 5.12b is the standard deviation of this computation. Again, as the swiss roll data is in $\mathbb{R}^3$ a much smaller number of nearest neighbors is needed to reconstruct each data point that the $K = 20$ allowed in our Sparse LLE computations and the $K = 12$ selected in standard LLE.

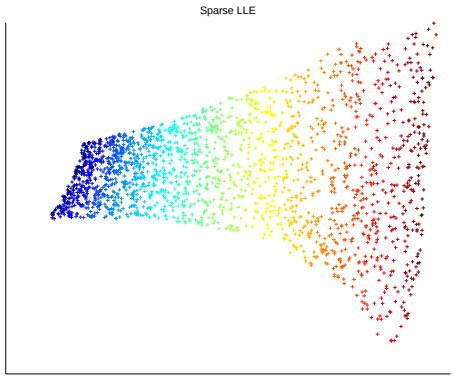
Therefore, through this experiment we see that Sparse LLE is able to not only preserve the topological structure of this highly nonlinear data set, but also automatically choose an appropriate number of nearest neighbors for each point, that agrees with intuition.



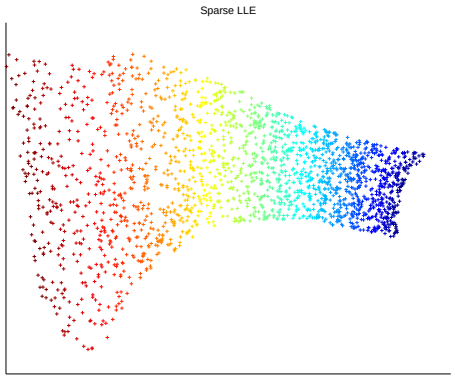
(A) LLE $K = 4$



(B) LLE $K = 12$



(C) Sparse LLE $K = 20$ $\lambda = 0.01$ Full

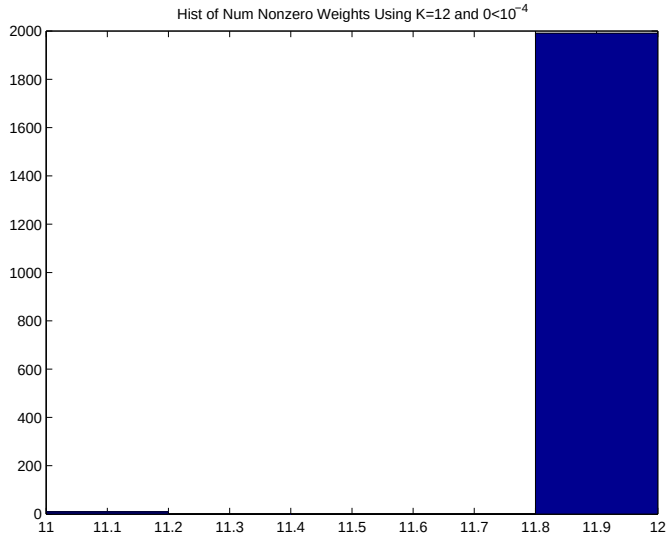


(D) Sparse LLE $K = 20$ $\lambda = 0.01$ Zeroed

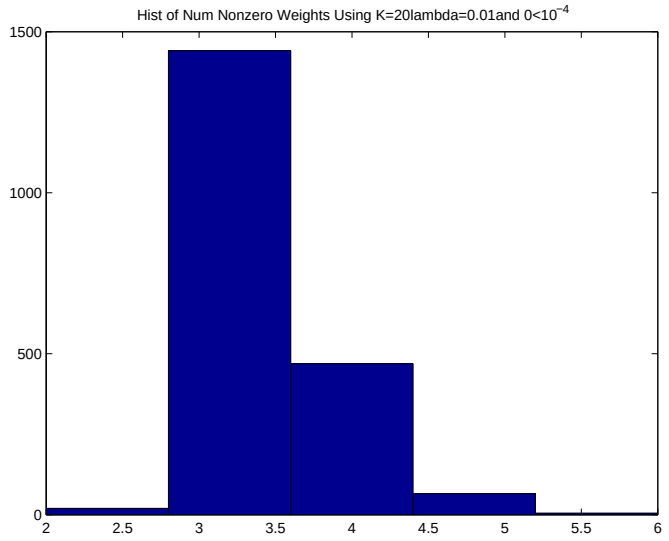
FIGURE 5.10. Points in $\mathbb{R}^3$ sampled from the swiss roll embedded into $\mathbb{R}^2$ using indicated algorithms and parameters, where Full indicates using the full $K \times p$ matrix W and Zeroed indicates zeroing those entries less than 10^{-4} in W.

5.8. Fabry-Perot Data Set

Field data collected by PSIs Range Test Validation System [26] was used as a sample data set to better understand Sparse LLE. This data set, which we will call the Fabry-Perot data set, was collected by an interferometer that measures the size of the elctromagnetic spectral radiance at certain wavelengths, namely those in the 8-11 micron range. It collects 20 images from 20 different wavelengths selected to maximize detection sensitivity of a particular



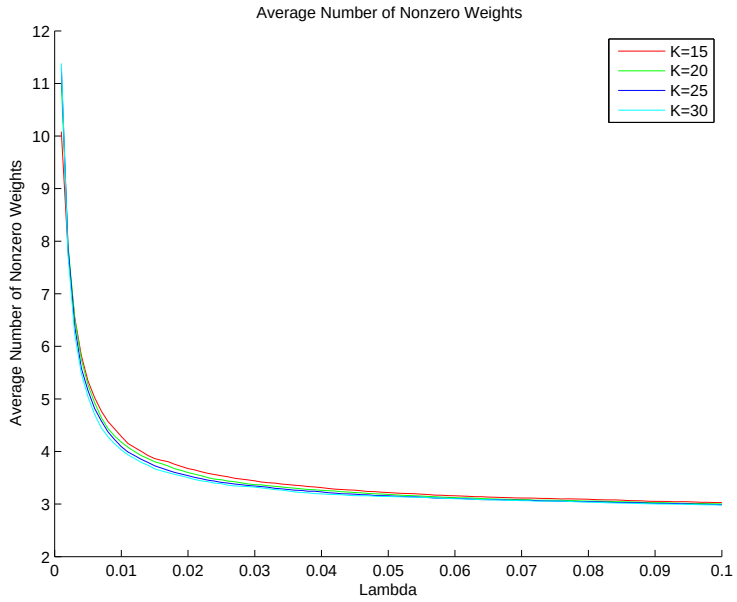
(A) LLE $K = 12$



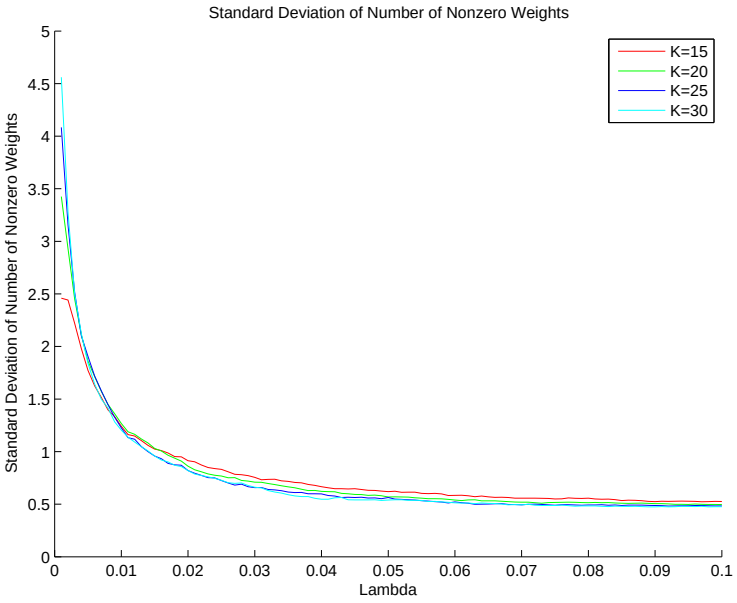
(B) Sparse LLE $K = 20$ $\lambda = 0.01$

FIGURE 5.11. Histograms of the number of nonzero weights associated to each point where the Sparse LLE weight matrix W had entries less than 10^{-4} zeroed out.

simulant. Each of these 20 images is of 256×256, and the 20 individual wavelength images–which we call sheets–form a single $256 \times 256 \times 20$ dimensional data cube representing a single moment in time. A predetermined quantity of a chemical simulant is released into the air



(A) Average



(B) Standard Deviation

FIGURE 5.12. Number of nonzero weights associated to nearest neighbors as both λ and K are varied.

to generate a cloud for detection against natural background. Data cubes are collected to record the event from 'pre-burst' to 'burst' to 'post-burst'. Three chemical simulants were

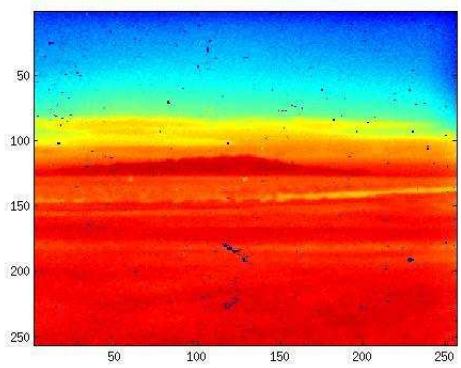


FIGURE 5.13. One sheet of a sample data cube.

released, and a series of data cubes were collected for the entirety of the event for each simulant. The three simulants are Glacial Acetic Acid (GAA), Methyl Salicylate (MeS), and Triethyl Phosphate (TEP). In this paper, we will focus on data cubes from the GAA data set.

5.8.1. PREPROCESSING. Each data cube is preprocessed by resizing, replacing missing data, and removing background. The data cubes are resized to to $26 \times 250 \times 20$ to focus on the area of interest, simply to speed up computations. As is evidenced by Figure 5.13, there is missing data that is replaced with the average pixel value. Finally, it should be noted that the chemical plume is not visible to the human eye, and thus, we remove the background of each data using the following process, see Figure 5.14:

(1) Consider a single 20-dimensional pixel through time $\mathbf{P}_i$ for frames known to be 'pre-burst', about $n = 50$ frames, $i = 1, \ldots, n$.

(2) Compute the mean of this pixel through time,

$$< P >= \frac{1}{n} \sum_{i=1}^{n} \mathbf{P}_i$$

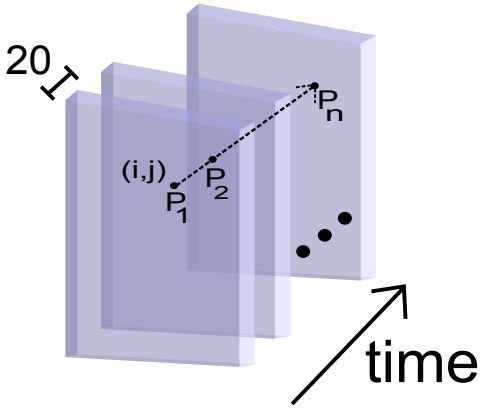


FIGURE 5.14. Illustration of considering a single pixel in $\mathbb{R}^{20}$ through time.

(3) Mean subtract this pixel at each time and arrange mean subtracted pixel at the columns of the matrix $\hat{P}$,

$$\hat{P} = \left[\hat{P}_1|\hat{P}_2|\dots|\hat{P}_n\right] = [P_1 - < P > |P_2 - < P > |\dots|P_n - < P >]$$

(4) Compute the singular value decomposition

$$\hat{P} = U\Sigma V^T$$

(5) Form the projection matrix onto the background basis

$$\mathbb{P}_k = \sum_{i=1}^{k} \mathbf{u}_i \mathbf{u}_i^T$$

(6) Project each pixel onto the null space, away from the projection basis

$$\hat{P}_i^R = (I - \mathbb{P}_k)\hat{P}_i$$

This is done beyond the 'pre-burst' data cubes to include all frames within the data set.

(7) Repeat all steps for each pixel.

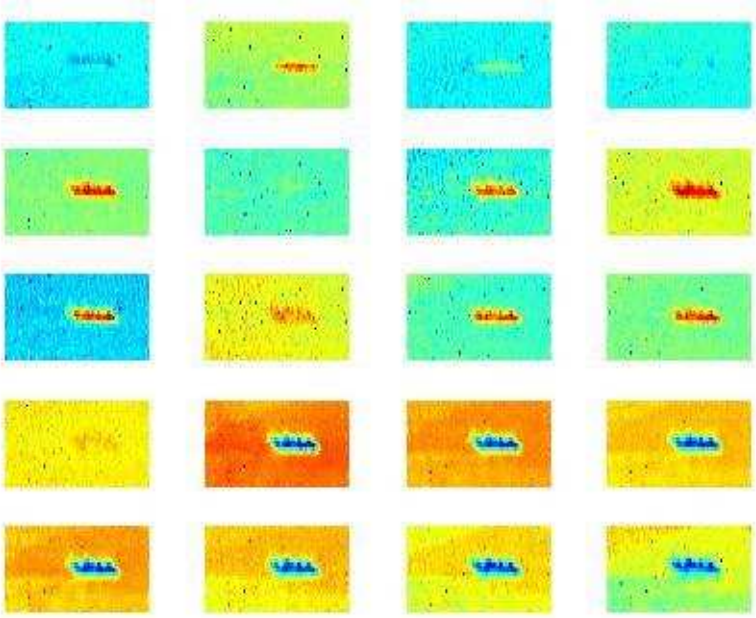


FIGURE 5.15. All 20 sheets of a pre-processed sample data cube.

This method results in the background being removed from all data cubes throughout the entirety of the event. See Figure 5.15 for a sample image of a preprocessed data cube. We now observe that the plume is visible.

As a final preprocessing step–that is used purely for visualization–we perform LBG clustering, as discussed in Chapter 2.1, on each of these pre-processed data cubes. We have selected three starting centers, one from the upper left, one from the lower right, and one from the plume. Observe a sample clustered data cube, GAA frame 70, 5.16. Each color represents an individual cluster with yellow representing chemical. This coloring will be used throughout the rest of this section.

5.8.2. VARYING PARAMETERS. We will now consider varying the parameters in both standard LLE and Sparse LLE, and then we will compare the reconstruction outputs. First, we reduce the dimension of GAA Frame 70 from $\mathbb{R}^{20}$ to $\mathbb{R}^2$ using both standard LLE and

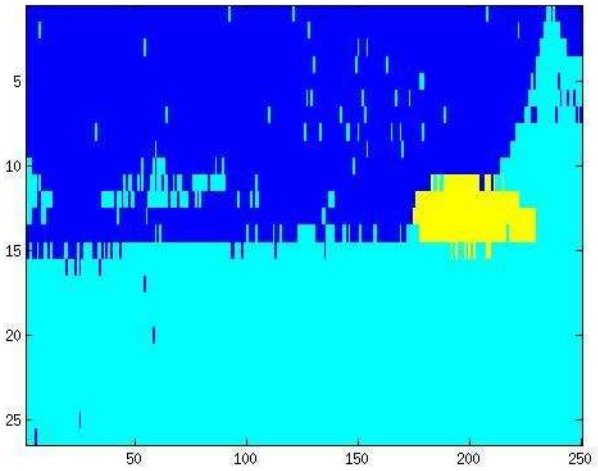
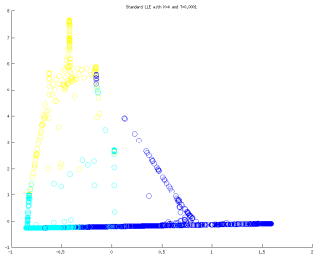


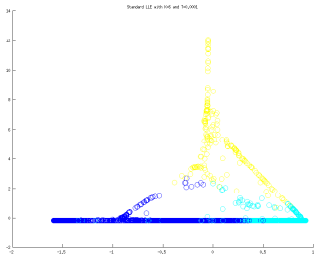
FIGURE 5.16. Frame 70 of GAA clustered using LBG.

Sparse LLE. In all experiments, we used $f(d(\mathbf{x}_i, \mathbf{x}_j)) = ||\mathbf{x}_i - \mathbf{x}_j||_2^2$. We implement standard LEE on this data set of size 20×6500 using a variety of choices of K. A few of these choices are displayed in Figure 5.17. We see that each of these choices of nearest neighbors yield drastically different results, except possibly $K = 6, 8$, keeping in mind that the problem is invariant to rotations, rescalings, and translations. There are some measures, such as the residual variance discussed in [57], that could be used to measure the reconstruction quality. However, by visual inspection, it seems that there is more separation between the chemical pixels and the background pixels when K is fairly small.

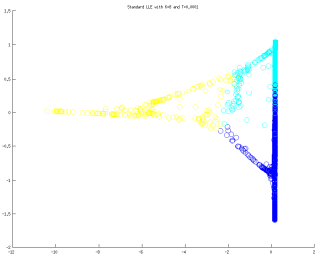
We will now focus on Sparse LLE. First, we fix $K = 30$ and vary λ, Figure 5.18. Notice that changing the parameter value λ appears to affect the reconstruction error. Now, we fix $\lambda = 0.05$ and vary K, see Figure 5.19. Notice that in this case, varying K also affects the output but there appears to be a bit more stability, especially when K is smaller. As we will shortly see, this is because not enough sparsity was induced in the weights. Finally, we again vary K, but fix $\lambda = 0.1$ which in turn will induce more sparsity, see Figure 5.20. Notice, that varying K in this case does not appear to affect the reconstruction output. This



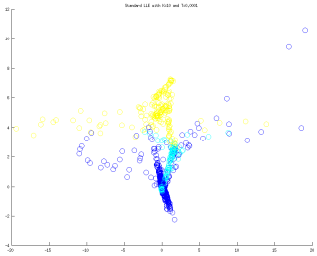
(A) LLE $K = 4$



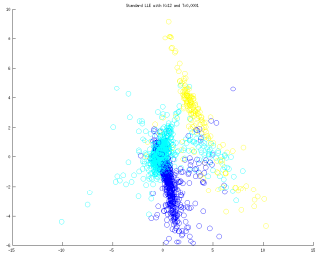
(B) LLE $K = 6$



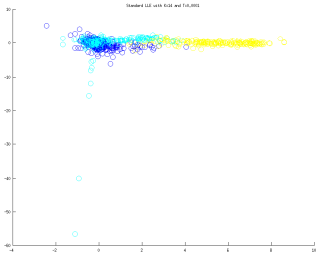
(C) LLE $K = 8$



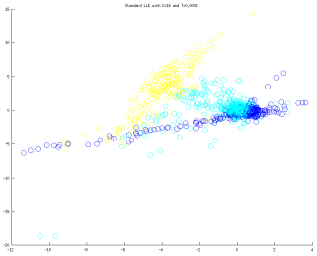
(D) LLE $K = 10$



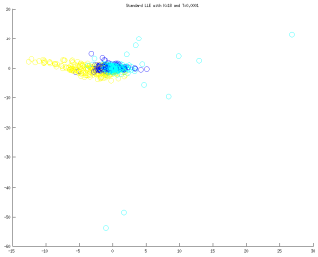
(E) LLE $K = 12$



(F) LLE $K = 14$

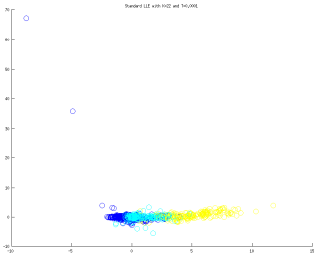


(G) LLE $K = 16$

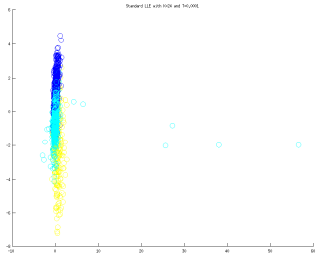


(H) LLE $K = 18$

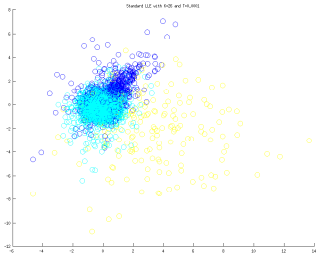
(I) LLE $K = 20$



(J) LLE $K = 22$

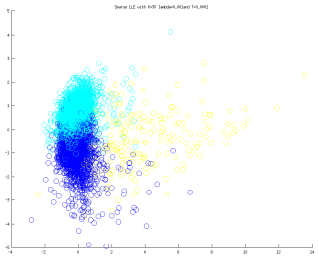


(K) LLE $K = 24$

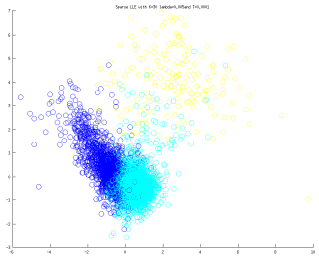


(L) LLE $K = 26$

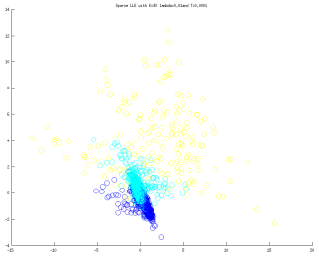
FIGURE 5.17. Reducing the 20-dimensional Fabry-Perot data down to $\mathbb{R}^2$ using standard LLE with varying K values.



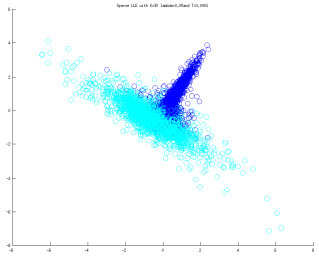
(A) SLLE $K = 30$ $\lambda = 0.001$



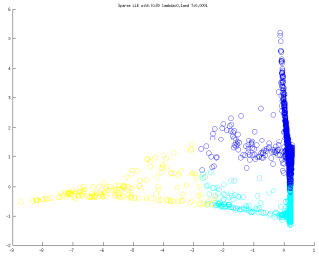
(B) SLLE $K = 30$ $\lambda = 0.005$



(C) SLLE $K = 30$ $\lambda = 0.01$



(D) SLLE $K = 30$ $\lambda = 0.05$

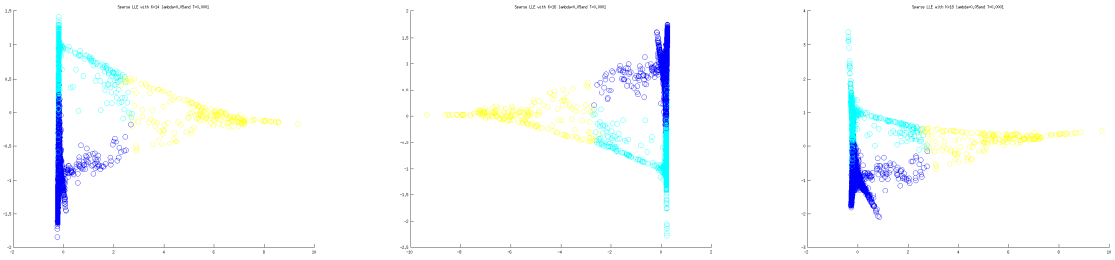


(E) SLLE $K = 30$ $\lambda = 0.1$

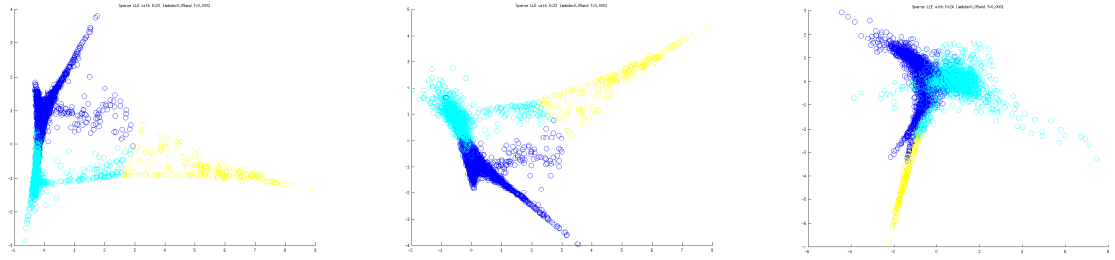
FIGURE 5.18. Reducing the 20-dimensional Fabry-Perot data down to $\mathbb{R}^2$ using Sparse LLE with fixed K values and varying λ.

is because we have induced enough sparsity in the problem that our results are consistent across different choices of nearest neighbors.

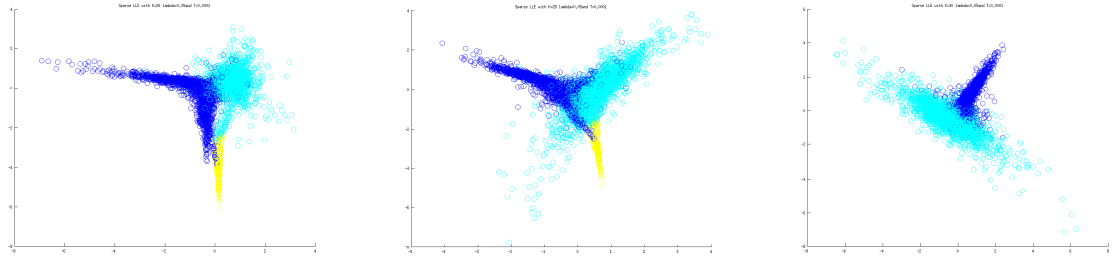
We should also note that all of these sets of images (including those for standard LLE) were created by zeroing out those weight values $w_{ij} < 10^{-4}$. In Figure 5.21, we fix the choice of nearest neighbors at $K = 30$, fix $\lambda = 0.1$ for Sparse LLE, but vary this tolerance to define zero values. Notice that in both cases the reconstructions appear to be very similar. However, this is for very different reasons. The weights in LLE are typically all larger than 10^{-4} while the unnecessary weights in Sparse LLE are often much less than this tolerance. We observe this in Table 5.1. Remember that there are $p \cdot K = 6500 \cdot 30 = 195000$ possible weights, so Sparse LLE is inducing sparsity in close to 60 percent of them, while standard LLE has virtually no zero weights.



(A) SLLE $K = 14$ (B) SLLE $K = 16$ (C) SLLE $K = 18$



(D) SLLE $K = 20$ (E) SLLE $K = 22$ (F) SLLE $K = 24$



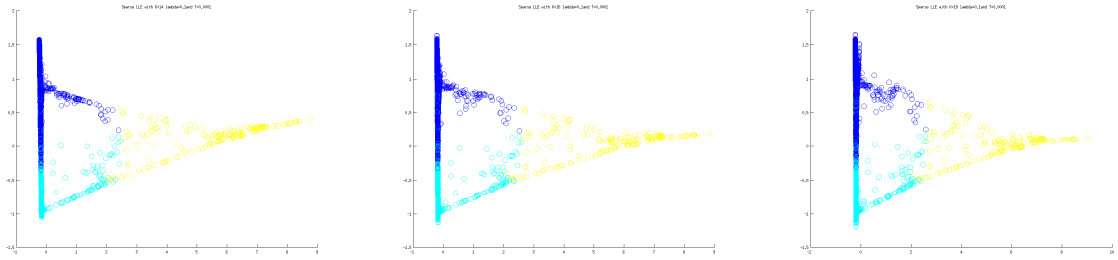
(G) SLLE $K = 26$ (H) SLLE $K = 28$ (I) SLLE $K = 30$

FIGURE 5.19. Reducing the 20-dimensional Fabry-Perot data down to $\mathbb{R}^2$ using Sparse LLE with varying K values and a fixed $\lambda = 0.05$.

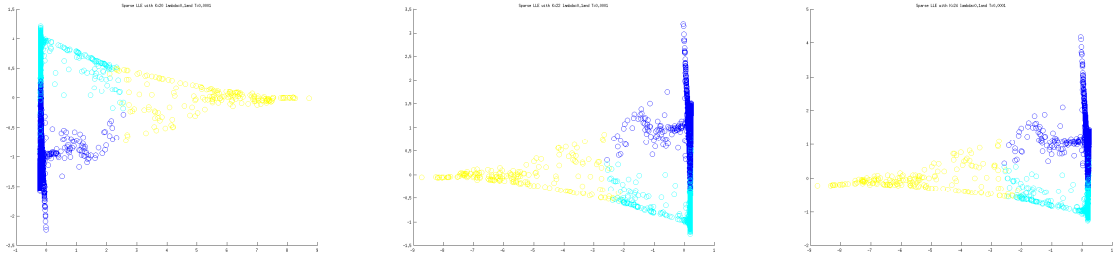
TABLE 5.1. Number of 'zero' weights as defined by those values less than the tolerance designated.

Tolerance	10^{-7}	10^{-6}	10^{-5}	10^{-4}
LLE	0	1	9	136
Sparse LLE	116430	119259	119492	119645

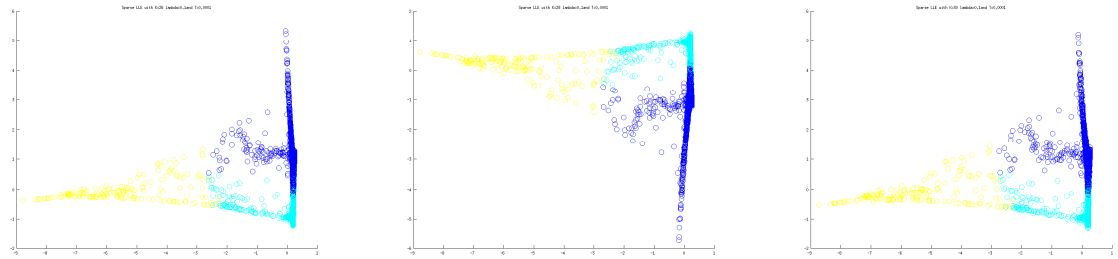
Similar to the experiment run for the swiss roll, let us now consider the average number of nonzero weights for all of the data points as we vary both λ and K. Notice that as λ increases, the average number of nonzero weights appears to level off between 8-10. Also,



(A) SLLE $K = 14$ (B) SLLE $K = 16$ (C) SLLE $K = 18$



(D) SLLE $K = 20$ (E) SLLE $K = 22$ (F) SLLE $K = 24$

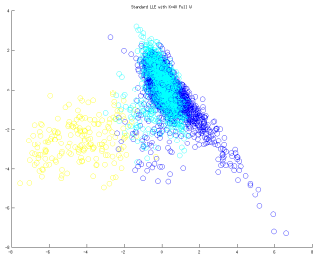


(G) SLLE $K = 26$ (H) SLLE $K = 28$ (I) SLLE $K = 30$

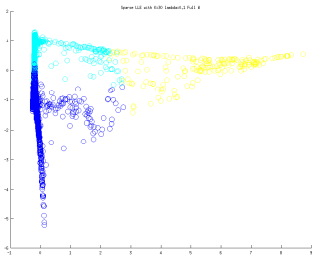
FIGURE 5.20. Reducing the 20-dimensional Fabry-Perot data down to $\mathbb{R}^2$ using Sparse LLE with varying K values and a fixed $\lambda = 0.1$.

notice that $K = 9$ in LLE produces similar results to Sparse LLE with $\lambda = 0.1$. Thus, Sparse LLE is able to automatically choose an appropriate number of nearest neighbors for this data set, which seems to be in the correct range that not only agrees with LLE reconstructions but more importantly reflects the likely topological structure of the data as all chemical points appear to be isolated from the background.

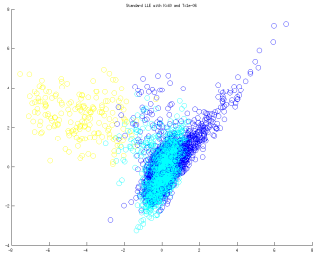
5.8.3. ANALYZING SEQUENCE OF DATA CUBES. In this experiment, we consider a sequence of successive data cubes from the GAA event. We look at the 50 frames, frame



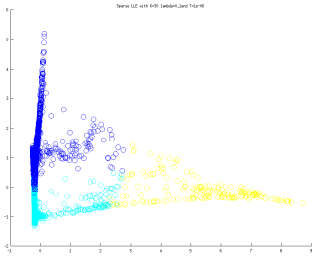
(A) LLE Full W



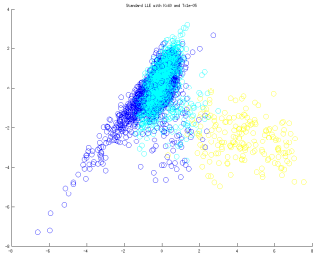
(B) Sparse LLE Full W



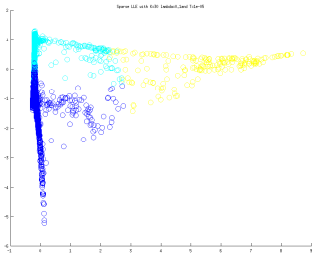
(C) LLE $tol = 10^{-6}$



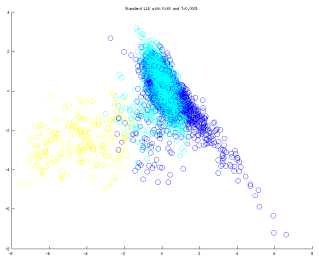
(D) Sparse LLE $tol = 10^{-6}$



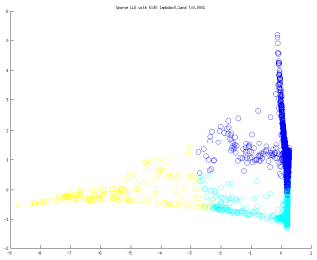
(E) LLE $tol = 10^{-5}$



(F) Sparse LLE $tol = 10^{-5}$

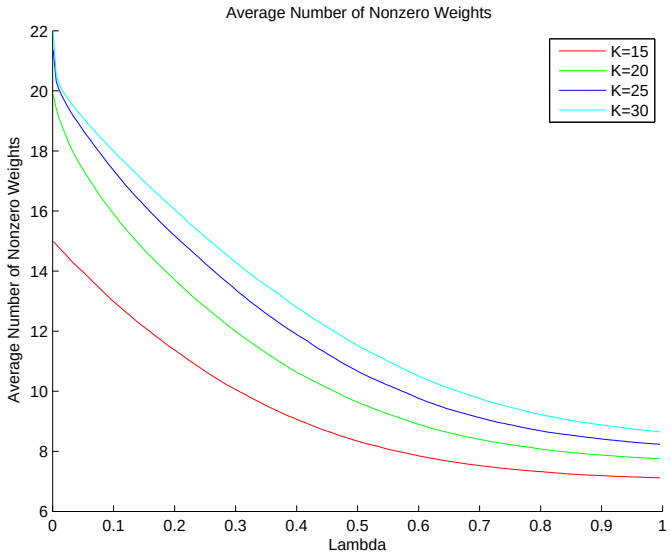


(G) LLE $tol = 10^{-4}$

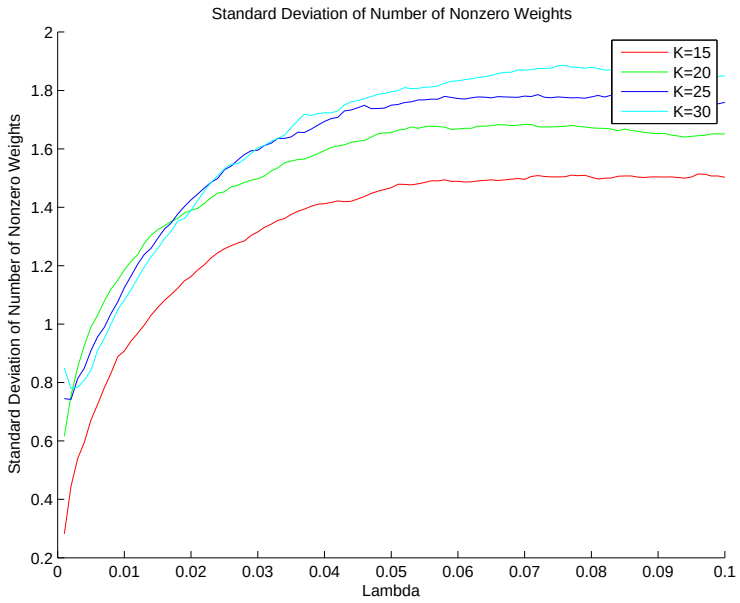


(H) Sparse LLE $tol = 10^{-4}$

FIGURE 5.21. Reducing the 20-dimensional Fabry-Perot data down to $\mathbb{R}^2$ using standard LLE and Sparse LLE with fixed $K = 40$ and $\lambda = 0.1$, but varying the cut off of zero, i.e. any $w_{ij} < tol$ will be zeroed out.



(A) Average



(B) Standard Deviation

FIGURE 5.22. Number of nonzero weights associated to nearest neighbors as both λ and K are varied.

numbers 70 through 119, which all occurred while the 'burst' was present in the scene. We have preprocessed each data cube as described above, including performing the clustering for visualization. Each of these frames can be viewed in the movie GAAmovie.avi.

Then, we implement Sparse LLE–with parameters $K = 20$ and $\lambda = 0.1$–on each of the data cubes individually. The 2-dimensional reconstructions can be viewed in the movie GAA_75Frames70to119SparseLLE.avi. Notice that as time progresses, it seems that the background pixels gradually start mixing with the chemical (yellow) pixels, which we would expect to happen as the chemical disperses into the atmosphere and becomes less pure.

5.9. Gene Expression Influenza Data

The final data set that we will consider is gene expression data from the Duke Influenza Study [85]. In this study, 17 volunteers were exposed to the H1N1 virus, and 19 volunteers were exposed to the H3N2 virus. Over a 24 hour period, 3 different measurements were collected from each volunteer. The measurements were gene expression data of dimension $D = 12023$. Therefore, our data set is 108×12023 dimensional. About half of all subjects produced symptoms while others remained healthy. Genes related to the immune system express themselves when an individual is ill. Therefore, individuals who contracted influenza during this 24 hour period will have larger values for genes relating to the immune system, indicating a more active gene. We have labels indicating those individuals who contracted influenza during these measurements and those who did not. These labels will be used later for visualization but were not used in analysis.

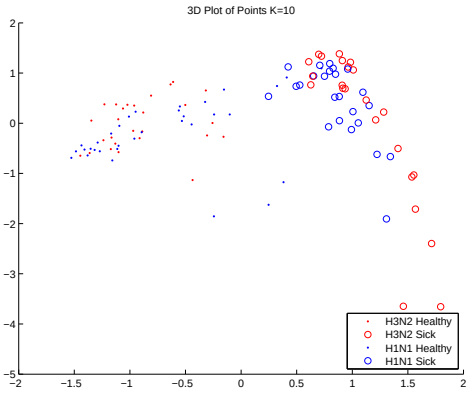
Now, it might be supposed that points in 12023-dimensional space representing those who were ill would tend to group together in this high dimensional space, while those who were not grouped together. Since we cannot visualize this data set in this space, we will reduce the dimension of this data set down to $\mathbb{R}^3$ and $\mathbb{R}^2$ using LLE and Sparse LLE. In this case, unlike in our other analysis, any choice of nearest neighbors will be much less than the dimension of our ambient space, i.e. $K < D$, and thus, our data points will be unable to be

perfectly reconstructed by their nearest neighbors. However, we will see that Sparse LLE is still able to remain invariant across the choice of nearest neighbors for an appropriately large λ.

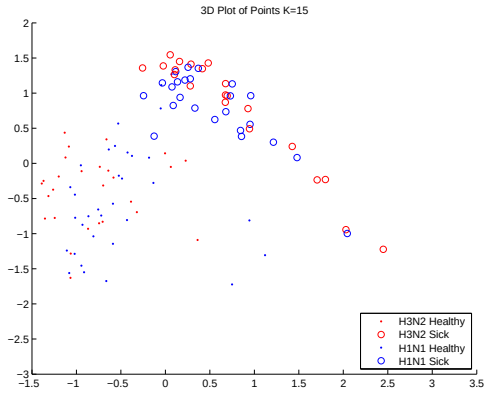
Consider the following. We vary the number of nearest neighbors K using standard LLE, and see that once again our reconstructions appear to differ depending on this choice, Figure 5.23. In visual inspection, it appears that $K = 6$ produces the most separation between sick and healthy individuals with only one misclassified point, Figure 5.24. However, without running through all choices of K, this particularly choice would not be apparent.

Instead, consider using Sparse LLE. We see that there appears to be a range of λ values for which varying K does not affect the output reconstructions, Figure 5.25. Notice, while there is not as much separation between the ill and healthy patients as in Figure 5.24, there is still fairly good separation, especially if the 3-dimension reconstructions are considered. A technician could analyze a new individual by running Sparse LLE again, which is very fast since the number of samples $p = 108$ to determine whether or not the patient is ill or healthy with fairly accurate results.

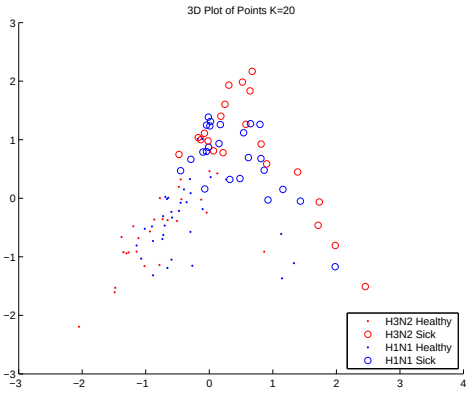
Finally, let us consider varying both K and λ and computing the average number of nonzero weights, Figure 5.26a, and the standard deviation, Figure 5.26b. We see that for K varying between 15 and 30 and $\lambda > 1$ the number of nonzero weights appears to level off in the range of 15 to 10. It should be noted that since $K < D$, larger choices of λ must be selected for a more consistent number of nonzero weights and there is more variability in this number.



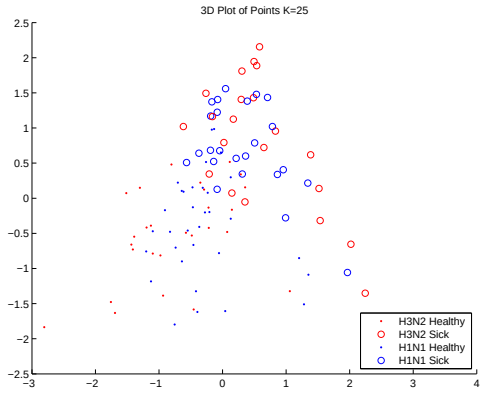
(A) LLE $K = 10$



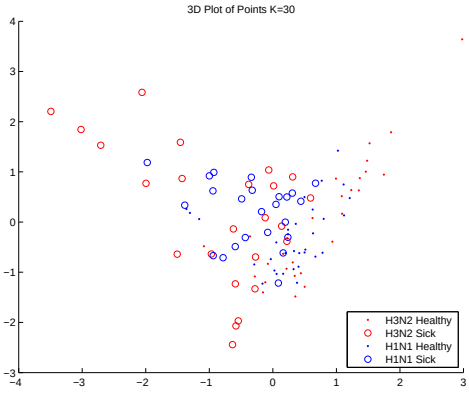
(B) LLE $K = 15$



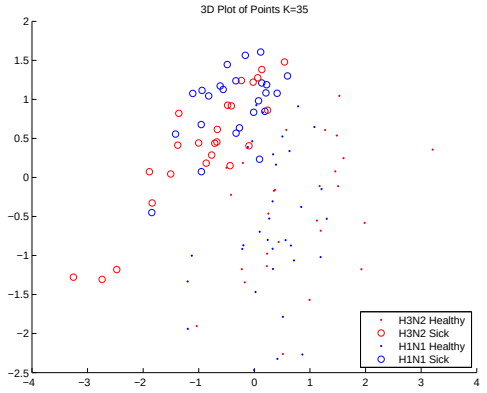
(C) LLE $K = 20$



(D) LLE $K = 25$



(E) LLE $K = 30$



(F) LLE $K = 35$

FIGURE 5.23. Reducing the 12023-dimensional data down to $\mathbb{R}^2$ using LLE while varying K.

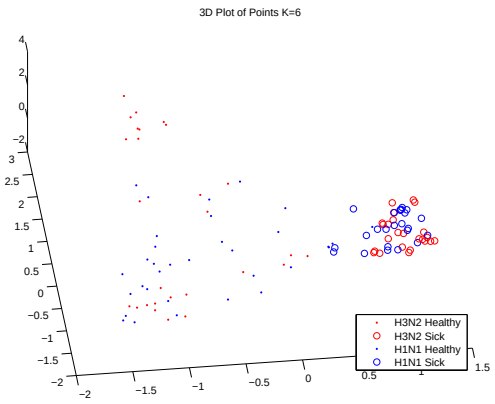
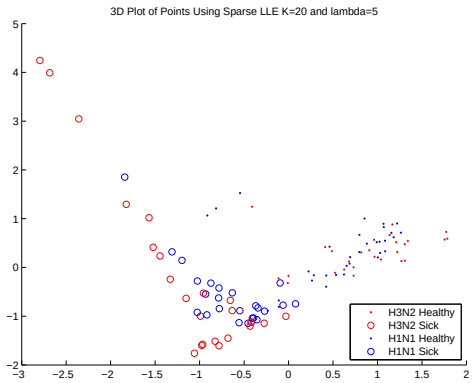


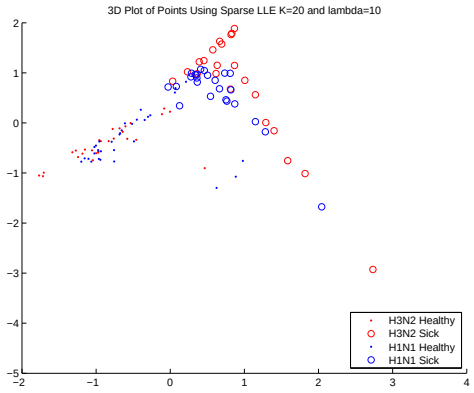
FIGURE 5.24. Reducing the 12023-dimensional data down to $\mathbb{R}^3$ using LLE with $K = 6$. By visual inspection, this choice of K appears to provide the best separation between sick and healthy individuals.

5.10. CONCLUSION

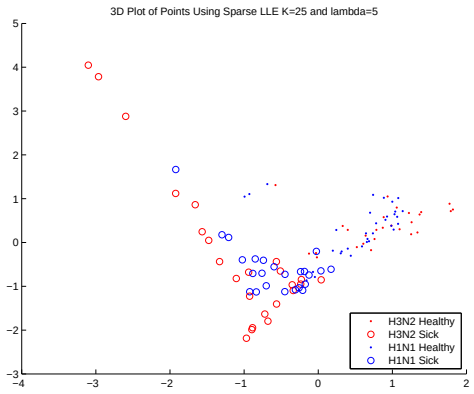
In this chapter, we have proposed a new algorithm, Sparse Locally Linear Embedding, that automatically determines an appropriate number of nearest neighbors K_i for each point $\mathbf{x}_i$. We see that this algorithm is robust to the choice of maximum K allowed if an appropriate parameter λ is selected. A variety of data sets have been analyzed including data where the number of nearest neighbors $K > D$ the ambient dimension of the data and vice versa.



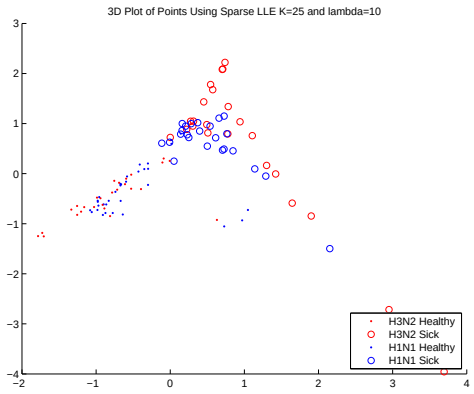
(A) SLLE $K = 20$ $\lambda = 5$



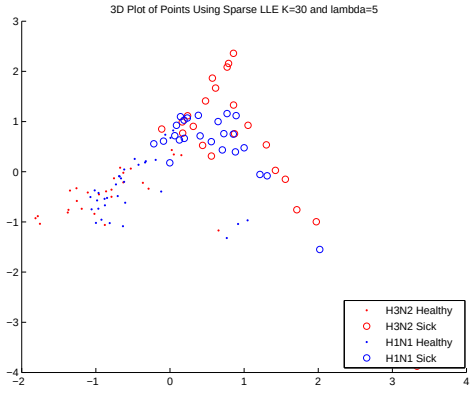
(B) SLLE $K = 20$ $\lambda = 10$



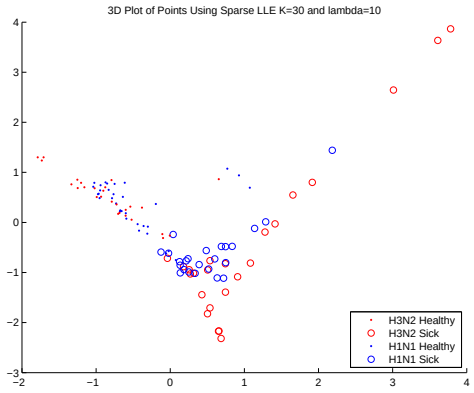
(C) SLLE $K = 25$



(D) SLLE $K = 25$ $\lambda = 10$

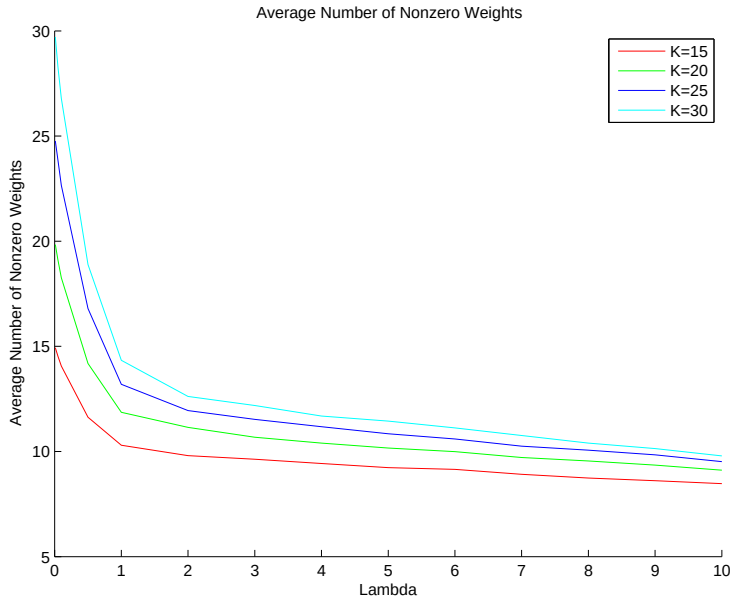


(E) SLLE $K = 30$

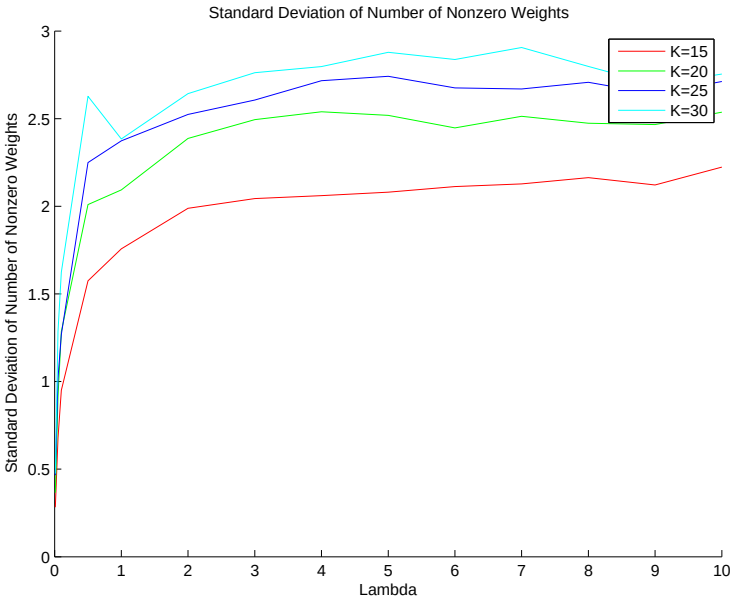


(F) SLLE $K = 30$ $\lambda = 10$

FIGURE 5.25. Reducing the 12023-dimensional data down to $\mathbb{R}^2$ using Sparse LLE while varying K and λ values.



(A) Average



(B) Standard Deviation

FIGURE 5.26. Number of nonzero weights associated to nearest neighbors as both λ and K are varied.

CHAPTER 6

Primal Dual Interior Point Method

6.1. Introduction

In this chapter, we will consider interior-point methods to solve convex optimization problems as discussed in [13], [15], [79]. We focus on quadratic programming problems with equality constraints where the decision variables are required to be nonnegative, namely

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{6.1}$$

Here, $\mathbf{x} \in \mathbb{R}^N$, Q is a symmetric positive semidefinite $N \times N$ matrix, $\mathbf{c} \in \mathbb{R}^N$, A is an $M \times N$ matrix representing M equality constraints, and $\mathbf{b} \in \mathbb{R}^M$. This is precisely the set-up of the Sparse LLE objective function 5.3 discussed in Section 5.2.

6.2. Central Path

We will solve problem 6.1 by reducing it to a sequence of linear, equality constrained problems and then applying Newton's method. Thus, we must rewrite our problem as an equality constrained problem by replacing the inequality constraints with a new term in the objective function that models the same behavior. Consider replacing each inequality constraint $x_i \geq 0$ by a term that is infinity when x_i is negative and zero otherwise such as

below:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} + \sum_i^n I(x_i) \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{b} \end{aligned} \tag{6.2}$$

where $I : \mathbb{R} \to \mathbb{R}$ is the indicator function

$$I(x_i) = \begin{cases} \infty & : x_i \leq 0 \\ 0 & : x_i > 0 \end{cases}$$

Keeping in mind that the goal is to find the optimal solution(s), it would be convenient to differentiate, set equal to zero, and use Newton's method to determine the roots, the optimal solutions. This new objective function is not differentiable as there is a sharp discontinuity. Instead, consider replacing this indicator function with a continuous function that is infinity when $x_i < 0$, finite for $x_i > 0$, and approaches infinity as x_i approaches zero. One such function is the *logarithmic barrier function*, $-\mu \sum_{i=1}^{n} \log x_i$ for a parameter $\mu > 0$, whose domain is the set of points satisfying strict inequalities, $x_i > 0$. Thus, adding this term to our objective function,

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} - \mu \sum_{i=1}^{n} \log x_i \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{b} \end{aligned} \tag{6.3}$$

we see that this new program approximates 6.1. Upon inspection, it seems that as μ tends to 0, the quality of this new formulation improves.

Notice that the parameter μ indexes a family of *barrier problems*,

$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimize}} && B_\mu(\mathbf{x}) \\ &\text{subject to} && A\mathbf{x} = \mathbf{b} \end{aligned} \tag{6.4}$$

where $B_\mu = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} - \mu \sum_{i=1}^{n} \log x_i$. Each barrier problem has a unique optimal solution, $x^*(\mu)$, for each μ [15]. We call the set of these optimal solutions the *central path*. As μ gets sufficiently close to 0, the solution to the barrier problem is arbitrarily close to the solution of the original quadratic program 6.1 [79].

6.3. Lagrange Multipliers

In order to solve each of these nonlinear barrier problems indexed by μ, we implement the method of Lagrange multipliers on our equality-constrained optimization problem. The Lagrangian for this problem is

$$L(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} - \mu \sum_i \log x_i + \mathbf{y}^T(A\mathbf{x} - \mathbf{b})$$

where the y_j for $j = 1, \ldots, M$ are our Lagrange multipliers.

Differentiating our Lagrangian with respect to each of our variables, $\mathbf{x}$ and $\mathbf{y}$, we obtain the first-order optimality conditions:

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{x}} &= Q\mathbf{x} + \mathbf{c} - \mu X^{-1}\mathbf{e} + A^T\mathbf{y} \\ \frac{\partial L}{\partial \mathbf{y}} &= A\mathbf{x} - \mathbf{b} \end{aligned}$$

Here, $\mathbf{e}$ denotes the vector of all ones, and X is the diagonal matrix of the vector $\mathbf{x}$ (this notation will be used throughout this chapter). If we let $\mathbf{z} = \mu X^{-1}\mathbf{e}$, then we obtain the following Karush-Kuhn-Tucker (KKT) conditions:

$$Q\mathbf{x} + \mathbf{c} - \mathbf{z} + A^T\mathbf{y} = 0 \tag{6.5}$$

$$A\mathbf{x} = \mathbf{b} \tag{6.6}$$

$$X\mathbf{z} = \mu\mathbf{e} \tag{6.7}$$

Notice that since $\mu > 0$ and all entries of X are nonnegative, then $\mathbf{z}$ is also required to be nonnegative. This condition (Equation 6.7) is referred to as the complementary slackness condition. As μ tends to 0, then $X\mathbf{z}$ tends to 0, a condition that must be satisfied at optimality. We see there is no restriction on $\mathbf{y}$, and thus, it is unconstrained.

6.4. The Dual

Let us now determine the dual function by considering the KKT system of equations. We see that the second equation, 6.6, is the primal equality constraint. As mentioned previously, from Equation 6.7, we see that there is a vector $\mathbf{z}_{N\times 1}$ that is complementary to the vector of primal variables $\mathbf{x}_{N\times 1}$. Thus, $\mathbf{z}$ is constrained to be nonnegative in the dual problem as $\mathbf{x} \geq \mathbf{0}$ and $\mu > 0$. The first equation 6.5 is also a dual constraint.

To determine the dual objective function, we recall the weak duality theorem.

Theorem 2 (Weak Duality Theorem)**.** *If f is the primal objective function, $\boldsymbol{u}$ is feasible for the primal problem, g is the dual objective function, and $\boldsymbol{v}$ is feasible for the dual, then*

$$g(\boldsymbol{v}) \leq f(\boldsymbol{u})$$

Therefore, the dual function gives us a lower bound on the optimal solution of our primal problem 6.1. Now, using equations 6.5 and 6.6, consider

$$\begin{aligned}
\mathbf{y}^T(A\mathbf{x}-\mathbf{b}) &= \mathbf{0} \\
\Rightarrow \mathbf{y}^T A\mathbf{x} &= \mathbf{y}^T\mathbf{b} \\
\Rightarrow (A^T\mathbf{y})^T\mathbf{x} &= (-\mathbf{c}+\mathbf{z}-Q\mathbf{x})^T\mathbf{x} \\
&= -\mathbf{c}^T\mathbf{x}+\mathbf{z}^T\mathbf{x}-\mathbf{x}^TQ\mathbf{x} \\
\Rightarrow \mathbf{y}^T\mathbf{b} &= -\mathbf{c}^T\mathbf{x}+\mathbf{z}^T\mathbf{x}-\mathbf{x}^TQ\mathbf{x}
\end{aligned}$$

Now, since $\mathbf{z}, \mathbf{x} \geq \mathbf{0}$, we know $\mathbf{z}^T\mathbf{x} \geq \mathbf{0}$. Thus,

$$\begin{aligned}
0 &\leq \mathbf{z}^T\mathbf{x} \\
&= \mathbf{c}^T\mathbf{x}+\mathbf{x}^TQ\mathbf{x}+\mathbf{b}^T\mathbf{y} \\
&= \mathbf{c}^T\mathbf{x}+\frac{1}{2}\mathbf{x}^TQ\mathbf{x}-(-\mathbf{b}^T\mathbf{y}-\frac{1}{2}\mathbf{x}^TQ\mathbf{x})
\end{aligned}$$

Then, $-\mathbf{b}^T\mathbf{y}-\frac{1}{2}\mathbf{x}^TQ\mathbf{x} \leq \mathbf{c}^T\mathbf{x}+\frac{1}{2}\mathbf{x}^TQ\mathbf{x}$, and thus, our dual problem must be

$$\begin{aligned}
\underset{\mathbf{x}}{\text{maximize}} \quad & -\mathbf{b}^T\mathbf{y}-\frac{1}{2}\mathbf{x}^TQ\mathbf{x} \\
\text{subject to} \quad & -Q\mathbf{x}+\mathbf{z}-A^T\mathbf{y} = \mathbf{c} \\
& \mathbf{z} \geq \mathbf{0}
\end{aligned}$$

As mentioned in [79], the $N \times 1$ vector $\mathbf{x}$ appears in the dual problem. This vector should have no connection to the variable $\mathbf{x}$ in the primal problem, except at optimality, they will be equal.

6.5. Primal Dual Interior Point Method

Let us now recall Newton's Method [79], used to find a root of a system of equations, i.e. $\mathbf{u}^* \in \mathbb{R}^n$ for which a system of n equations $F(\mathbf{u}^*) = \mathbf{0}$. This is an iterative method where step directions $\Delta\mathbf{u}$ are computed such that $F(\mathbf{u} + \Delta\mathbf{u}) = \mathbf{0}$. We do this by approximating F, which is potentially nonlinear, by the $\Delta\mathbf{u}$ linear terms of its Taylor's series expansion

$$F(\mathbf{u} + \Delta\mathbf{u}) \approx F(\mathbf{u}) + F'(\mathbf{u})\Delta\mathbf{u}.$$

Therefore, we solve

$$F'(\mathbf{u})\Delta\mathbf{u} = -F(\mathbf{u}).$$

Given a starting solution $\mathbf{u}$, Newton's method updates by replacing $\mathbf{u}$ with $\mathbf{u}+\Delta\mathbf{u}$, iterating until $F(\mathbf{u}) \approx \mathbf{0}$ as designated by some tolerance.

We will use Newton's method to solve the KKT system of equations that was derived above

$$Q\mathbf{x} + \mathbf{c} - \mathbf{z} + A^T\mathbf{y} = 0 \tag{6.8}$$

$$A\mathbf{x} = \mathbf{b} \tag{6.9}$$

$$X\mathbf{z} = \mu\mathbf{e} \tag{6.10}$$

We may start with an arbitrary choice of values for all primal and dual variables, requiring $\mathbf{x}, \mathbf{z} > 0$, but not necessarily requiring these choices to be feasible. These variables are iteratively updated by taking a single Newton's step followed by a reduction in the value of the barrier parameter μ. Thus, we must determine the step directions $(\Delta\mathbf{x}, \Delta\mathbf{y}, \Delta\mathbf{z})$ pointing

approximately at the point $(\mathbf{x}_\mu, \mathbf{y}_\mu, \mathbf{z}_\mu)$ on the central path. The new point $(\mathbf{x} + \Delta\mathbf{x}, \mathbf{y} + \Delta\mathbf{y}, \mathbf{z} + \Delta\mathbf{z})$ will then lie approximately on the primal-dual central path.

We must determine these step directions by replacing $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ with $(\mathbf{x} + \Delta\mathbf{x}, \mathbf{y} + \Delta\mathbf{y}, \mathbf{z} + \Delta\mathbf{z})$. The system of equations 6.8 to 6.10 becomes

$$\begin{aligned} Q(\mathbf{x} + \Delta\mathbf{x}) + \mathbf{c} - (\mathbf{z} + \Delta\mathbf{z}) + A^T(\mathbf{y} + \Delta\mathbf{y}) &= \mathbf{0} \\ A(\mathbf{x} + \Delta\mathbf{x}) &= \mathbf{b} \\ (\mathbf{X} + \Delta\mathbf{X})(\mathbf{z} + \Delta\mathbf{z}) &= \mu\mathbf{e} \end{aligned}$$

We drop the nonlinear terms to obtain a linear system and solve for the unknowns, $(\Delta\mathbf{x}, \Delta\mathbf{y}, \Delta\mathbf{z})$.

$$\begin{aligned} Q\Delta\mathbf{x} - \Delta\mathbf{z} + A^T\Delta\mathbf{y} &= -\mathbf{c} - Q\mathbf{x} + \mathbf{z} - A^T\mathbf{y} \\ A\Delta\mathbf{x} &= \mathbf{b} - A\mathbf{x} \\ Z\Delta\mathbf{x} + X\Delta\mathbf{z} &= \mu\mathbf{e} - X\mathbf{z} \end{aligned}$$

Rewriting this system of equations in matrix form, we obtain

$$\begin{bmatrix} Q & A^T & -I \\ A & 0 & 0 \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \\ \Delta\mathbf{z} \end{bmatrix} = - \begin{bmatrix} \mathbf{c} + Q\mathbf{x} - \mathbf{z} + A^T\mathbf{y} \\ -\mathbf{b} + A\mathbf{x} \\ -\mu\mathbf{e} + X\mathbf{z} \end{bmatrix} \tag{6.11}$$

and solve for $(\Delta\mathbf{x}, \Delta\mathbf{y}, \Delta\mathbf{z})$ using, for example, Gaussian Elimination.

Note that it is important to choose μ so that it is not too large or too small. We would like to reflect the current values of our variables in our choice of μ. As $X\mathbf{z} = \mu\mathbf{e}$, then μ

could be the average

$$\mu = \frac{\mathbf{x}^T\mathbf{z}}{N}.$$

This would assume that our variables lie exactly on the central path. However, this is most likely not the case. We reduce each value of μ by a factor in order to produce a new solution that is closer to the optimal solution on the central path, i.e.

$$\mu = \delta\frac{\mathbf{x}^T\mathbf{z}}{N}.$$

In implementation, we have used $\delta = \frac{1}{10}$ as recommended by [79].

We now must compute a step length parameter θ in order to update our solution along the step direction $(\Delta\mathbf{x}, \Delta\mathbf{y}, \Delta\mathbf{z})$ i.e.

$$\hat{\mathbf{x}} = \mathbf{x} + \theta\Delta\mathbf{x}$$

$$\hat{\mathbf{y}} = \mathbf{y} + \theta\Delta\mathbf{y}$$

$$\hat{\mathbf{z}} = \mathbf{z} + \theta\Delta\mathbf{z}$$

If θ was chosen arbitrarily, updating the solutions may lead to a new solution where $\mathbf{x}$ and $\mathbf{z}$ do not remain positive. It is therefore necessary to determine a step length parameter that will not violate these conditions, i.e.

$$x_i + \theta\Delta x_i > 0 \text{ for } i = 1, \ldots, n.$$

We assume θ to be positive and the primal constraint requires, $x_i > 0$, so

$$\frac{1}{\theta} = -\frac{\Delta x_i}{x_i} \text{ for } i = 1, \ldots, n.$$

We do the same for the dual variables $\mathbf{z}$, and see

$$\frac{1}{\theta} = \max_i \left\{ -\frac{\Delta x_i}{x_i}, -\frac{\Delta z_i}{z_i} \right\}$$

To guarantee that our inequality holds, we multiply by a factor, α close to but less than one

$$\theta = \alpha \frac{1}{\max_i \left\{ -\frac{\Delta x_i}{x_i}, -\frac{\Delta z_i}{z_i} \right\}}.$$

In implementation, we have chosen $\lambda = 0.9$. Typically, we choose θ to be the minimum of either this computed value or 1.

It should be noted that although we require positivity of $\mathbf{x}$ and $\mathbf{z}$ throughout the algorithm, optimal solutions may occur at 0. This can be achieved "in the limit" as described by [79]. For convergence analysis, please refer to this text as well.

We summarize the Primal-Dual Interior Point (PDIP) Method in the following Algorithm:

PDIP Algorithm

(1) Initialize values for all primal and dual variables $\mathbf{u} = (\mathbf{x}, \mathbf{y}, \mathbf{z})$ with $\mathbf{x}, \mathbf{z} > 0$.

(2) Estimate a value for μ (i.e. smaller than the 'current' value) using, for example,

$$\mu = \delta \frac{\mathbf{x}^T \mathbf{z}}{N}.$$

(3) Compute step direction $(\Delta\mathbf{x}, \Delta\mathbf{y}, \Delta\mathbf{z})$ pointing in direction of central path by solving

$$\begin{bmatrix} Q & A^T & -I \\ A & 0 & 0 \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \\ \Delta\mathbf{z} \end{bmatrix} = - \begin{bmatrix} \mathbf{c} + Q\mathbf{x} - \mathbf{z} + A^T\mathbf{y} \\ -\mathbf{b} + A\mathbf{x} \\ -\mu\mathbf{e} + X\mathbf{z} \end{bmatrix}.$$

(4) Update variables $\mathbf{u} + \theta\Delta\mathbf{u}$ where θ is an appropriate step length such as

$$\theta = \min\left(\alpha \frac{1}{\max_i \left\{-\frac{\Delta x_i}{x_i}, -\frac{\Delta z_i}{z_i}\right\}}, 1\right).$$

(5) Repeat 2-4 until optimality, $F(\mathbf{u}) \approx 0$.

6.6. Reducing the KKT System

The KKT equations 6.11 are a $(2N + M) \times (2N + M)$ linear system in $(2N + M)$ unknowns. Solving this system of equations is the most computationally intensive step of the PDIP Algorithm. Using basic linear algebra, we may eliminate variables from this system, resulting in a smaller linear system to solve.

Recall our system of equations,

$$Q\Delta\mathbf{x} + A^T\Delta\mathbf{y} - \Delta\mathbf{z} = -A^T\mathbf{y} + \mathbf{z} - \mathbf{c} - Q\mathbf{x} \tag{6.12}$$

$$A\Delta\mathbf{x} = \mathbf{b} - A\mathbf{x} \tag{6.13}$$

$$Z\Delta\mathbf{x} + X\Delta\mathbf{z} = \mu\mathbf{e} - X\mathbf{z} \tag{6.14}$$

As $\Delta\mathbf{z}$ is isolated in 6.12, it is natural to solve for this variable first.

$$Q\Delta\mathbf{x} + A^T\Delta\mathbf{y} - \Delta\mathbf{z} = -A^T\mathbf{y} + \mathbf{z} - \mathbf{c} - Q\mathbf{x} \tag{6.15}$$

$$\Delta\mathbf{z} = Q\Delta\mathbf{x} + A^T\Delta\mathbf{y} + A^T\mathbf{y} - \mathbf{z} + \mathbf{c} + Q\mathbf{x} \tag{6.16}$$

Now, we substitute this formula into 6.14

$$Z\Delta\mathbf{x} + X\Delta\mathbf{z} = \mu\mathbf{e} - X\mathbf{z} \tag{6.17}$$

$$Z\Delta\mathbf{x} + X(Q\Delta\mathbf{x} + A^T\Delta\mathbf{y} + A^T\mathbf{y} - \mathbf{z} + \mathbf{c} + Q\mathbf{x}) = \mu\mathbf{e} - X\mathbf{z} \tag{6.18}$$

$$(Z + XQ)\Delta\mathbf{x} + XA^T\Delta\mathbf{y} = \mu\mathbf{e} - XA^T\mathbf{y} - X\mathbf{c} - XQ\mathbf{x} \tag{6.19}$$

Putting this together with 6.13, we have a reduced $(N + M) \times (N + M)$ linear system in $(N + M)$ unknowns

$$\begin{bmatrix} Z + XQ & XA^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \end{bmatrix} = \begin{bmatrix} \mu\mathbf{e} - XA^T\mathbf{y} - X\mathbf{c} - XQ\mathbf{x} \\ \mathbf{b} - A\mathbf{x} \end{bmatrix}. \tag{6.20}$$

This system is often referred to as the reduced system.

We may further reduce our system down to only one variable by solving for $\Delta\mathbf{x}$ in 6.19

$$(Z + XQ)\Delta\mathbf{x} + XA^T\Delta\mathbf{y} = \mu\mathbf{e} - XA^T\mathbf{y} - X\mathbf{c} - XQ\mathbf{x} \tag{6.21}$$

$$\Delta\mathbf{x} = (Z + XQ)^{-1}(\mu\mathbf{e} - XA^T\mathbf{y} - X\mathbf{c} - XQ\mathbf{x} - XA^T\Delta\mathbf{y})$$

Now, substitute into 6.13

$$A\Delta\mathbf{x} = \mathbf{b} - A\mathbf{x}$$

$$A(Z + XQ)^{-1}(\mu\mathbf{e} - XA^T\mathbf{y} - X\mathbf{c} - XQ\mathbf{x} - XA^T\Delta\mathbf{y})) = \mathbf{b} - A\mathbf{x}$$

to obtain

$$-A(Z + XQ)^{-1}XA^T\Delta\mathbf{y} = \mathbf{b} - A\mathbf{x} - A(Z + XQ)^{-1}(\mu\mathbf{e} - XA^T\mathbf{y} - X\mathbf{c} - XQ\mathbf{x})$$

$$A(Z + XQ)^{-1}XA^T\Delta\mathbf{y} = -A((Z + XQ)^{-1}(-\mu\mathbf{e} + XA^T\mathbf{y} + X\mathbf{c} + XQ\mathbf{x}) - \mathbf{x}) - \mathbf{b}. \quad (6.22)$$

This system is often referred to as the normal system.

Equation 6.22 is now an $M \times M$ linear system in M variables, but in order to formulate this linear system the inverse of $(Z + XQ)$ must be computed. Let us now consider an alternative to solving for this inverse which may be numerically unstable and may require more computational time. Instead, let

$$W = A(Z + XQ)^{-1}.$$

Now, A is likely not a square matrix, so take the transpose

$$W^T = (A(Z + XQ)^{-1}))^T = ((Z + XQ)^{-1})^T A^T = ((Z + XQ)^T)^{-1}A^T.$$

Therefore,

$$(Z + XQ)^T W^T = A^T.$$

We solve this system of equations and substitute W in equation 6.22

$$WXA^T\Delta\mathbf{y} = -W(-\mu\mathbf{e} + XA^T\mathbf{y} + X\mathbf{c} + XQ\mathbf{x}) - \mathbf{x}) - \mathbf{b}. \tag{6.23}$$

We call this the normal without inverses system of equations.

On a final note, [79] formulates the KKT conditions in a slightly different manner and also computes inverses in each of the reduction steps. We discuss this formulation in the next section.

6.7. Vanderbei PDIP Formulation

As mentioned in Section 6.6, [79] formulates the KKT system and in turn, the reduced systems slightly differently. We will explore this formulation in this section. Consider the quadratic program defined with equality constraints and nonnegativity constraints on the decision variables. Note [79] actually formulates using both equality constraints and inequality constraints, but we have taken the approach that all inequality constraints can be written as equality constraints with slack variables added to the problem.

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

Now, let's form the Lagrangian

$$L(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} - \mu\sum_i \log x_i + \mathbf{y}^T(\mathbf{b} - A\mathbf{x})$$

Differentiating our Lagrangian with respect to each of our variables, $\mathbf{x}$ and $\mathbf{y}$, we obtain the first-order optimality conditions:

$$\frac{\partial L}{\partial \mathbf{x}} = Q\mathbf{x} + \mathbf{c} - \mu X^{-1}\mathbf{e} - A^T\mathbf{y} \tag{6.24}$$

$$\frac{\partial L}{\partial y} = \mathbf{b} - A\mathbf{x} \tag{6.25}$$

We let $\mathbf{z} = \mu X^{-1}\mathbf{e}$, then then we obtain the following KKT conditions:

$$\begin{aligned} Q\mathbf{x} + \mathbf{c} - \mathbf{z} - A^T\mathbf{y} &= 0 \\ A\mathbf{x} &= \mathbf{b} \\ X\mathbf{z} &= \mu\mathbf{e} \end{aligned}$$

We will use Newton's method to solve this system of equations by replacing $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ with $(\mathbf{x} + \Delta\mathbf{x}, \mathbf{y} + \Delta\mathbf{y}, \mathbf{z} + \Delta\mathbf{z})$. Then, our system becomes

$$\begin{aligned} Q(\mathbf{x} + \Delta\mathbf{x}) + \mathbf{c} - (\mathbf{z} + \Delta\mathbf{z}) - A^T(\mathbf{y} + \Delta\mathbf{y}) &= \mathbf{0} \\ A(\mathbf{x} + \Delta\mathbf{x}) &= \mathbf{b} \\ (\mathbf{X} + \Delta\mathbf{X})(\mathbf{z} + \Delta\mathbf{z}) &= \mu\mathbf{e} \end{aligned}$$

We drop the nonlinear terms to obtain a linear system, and solve for the unknowns, $(\Delta\mathbf{x}, \Delta\mathbf{y}, \Delta\mathbf{z})$.

$$Q\Delta\mathbf{x} - \Delta\mathbf{z}) - A^T\Delta\mathbf{y} = -\mathbf{c} - Q\mathbf{x} + \mathbf{z} + A'\mathbf{y} \tag{6.26}$$

$$A\Delta\mathbf{x} = \mathbf{b} - A\mathbf{x} \tag{6.27}$$

$$Z\Delta\mathbf{x} + X\Delta\mathbf{z} = \mu\mathbf{e} - X\mathbf{z} \tag{6.28}$$

Rewriting this linear system of equations in matrix form, we obtain

$$\begin{bmatrix} -Q & A^T & I \\ A & 0 & 0 \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \\ \Delta\mathbf{z} \end{bmatrix} = - \begin{bmatrix} \mathbf{c} + Q\mathbf{x} - \mathbf{z} + A^T\mathbf{y} \\ -\mathbf{b} + A\mathbf{x} \\ -\mu\mathbf{e} + X\mathbf{z} \end{bmatrix} \tag{6.29}$$

Let us now consider the dual of this problem

$$\begin{aligned} \mathbf{y}^T(A\mathbf{x} - \mathbf{b}) &= \mathbf{0} \\ \Rightarrow \mathbf{y}^T A\mathbf{x} &= \mathbf{y}^T\mathbf{b} \\ \Rightarrow (A^T\mathbf{y})^T\mathbf{x} &= (\mathbf{c} - \mathbf{z} + Q\mathbf{x})^T\mathbf{x} \\ &= \mathbf{c}^T\mathbf{x} - \mathbf{z}^T\mathbf{x} + \mathbf{x}^T Q\mathbf{x} \\ \Rightarrow \mathbf{y}^T\mathbf{b} &= \mathbf{c}^T\mathbf{x} - \mathbf{z}^T\mathbf{x} + \mathbf{x}^T Q\mathbf{x} \end{aligned}$$

Now, since $\mathbf{z}, \mathbf{x} \geq \mathbf{0}$, we know $\mathbf{z}^T\mathbf{x} \geq \mathbf{0}$. Thus,

$$\begin{aligned} 0 &\leq \mathbf{z}^T\mathbf{x} \\ &= \mathbf{c}^T\mathbf{x} + \mathbf{x}^T Q\mathbf{x} - \mathbf{b}^T\mathbf{y} \\ &= \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T Q\mathbf{x} - (\mathbf{b}^T\mathbf{y} - \frac{1}{2}\mathbf{x}^T Q\mathbf{x}) \end{aligned}$$

Then, $\mathbf{b}^T\mathbf{y} - \frac{1}{2}\mathbf{x}^T Q\mathbf{x} \leq \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T Q\mathbf{x}$, and thus, our dual problem must be

$$\begin{aligned} \underset{\mathbf{x}}{\text{maximize}} \quad & \mathbf{b}^T\mathbf{y} - \frac{1}{2}\mathbf{x}^T Q\mathbf{x} \\ \text{subject to} \quad & -Q\mathbf{x} + \mathbf{z} - A^T\mathbf{y} = \mathbf{c} \\ & \mathbf{z} \geq \mathbf{0} \end{aligned}$$

Notice that this is the same as the original formulation, except the first term is not negated agreeing with the formulation presented in [79]. Therefore, the solutions of the Lagrange multipliers $\mathbf{y}$ will have opposite signs in each of the formulations.

Now, we will consider eliminating variables from this system of equations in order to solve a smaller linear system. Solve for $\Delta\mathbf{z}$ in Equation 6.28

$$Z\Delta\mathbf{x} + X\Delta\mathbf{x} = \mu\mathbf{e} - \mathbf{xz}$$

$$\Delta\mathbf{z} = X^{-1}(\mu\mathbf{e} - X\mathbf{z} - Z\Delta\mathbf{x} = \mu X^{-1}\mathbf{e} - \mathbf{z} - X^{-1}Z\Delta\mathbf{x}$$

We substitute this formula into Equation 6.26

$$\begin{aligned} -Q\Delta\mathbf{x} + A^T\Delta\mathbf{y} + \Delta\mathbf{z} &= \mathbf{c} + Q\mathbf{x} - A^T\mathbf{y} - \mathbf{z} \\ -Q\Delta\mathbf{x} + A^T\Delta\mathbf{y} + \mu X^{-1}\mathbf{e} - \mathbf{z} - X^{-1}Z\Delta\mathbf{x} &= \mathbf{c} + Q\mathbf{x} - A^T\mathbf{y} - \mathbf{z} \\ -(Q + X^{-1}Z)\Delta\mathbf{x} + A^T\Delta\mathbf{y} &= \mathbf{c} + Q\mathbf{x} - A^T\mathbf{y} - \mu X^{-1}\mathbf{e} \end{aligned} \tag{6.30}$$

Putting this together with Equation 6.27, we have a reduced $(N + M) \times (N + M)$ linear system in $(N + M)$ unknowns.

$$\begin{bmatrix} -(Q + X^{-1}Z) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{c} + Q\mathbf{x} - A^T\mathbf{y} - \mu X^{-1}\mathbf{e} \\ \mathbf{b} - A\mathbf{x} \end{bmatrix} \tag{6.31}$$

This system of equations is referred to as the reduced system.

We may further reduce our system down to only one variable by solving for $\Delta\mathbf{x}$ in Equation 6.30.

$$\begin{aligned} -(Q + X^{-1}Z)\Delta\mathbf{x} + A^T\Delta\mathbf{y} &= \mathbf{c} + Q\mathbf{x} - A^T\mathbf{y} - \mu X^{-1}\mathbf{e} \\ \Delta\mathbf{x} &= -(Q + X^{-1}Z)^{-1}(\mathbf{c} + Q\mathbf{x} - A^T\mathbf{y} - \mu X^{-1}\mathbf{e} - A^T\Delta\mathbf{y} \end{aligned}$$

Now we substitute into our final equation

$$\begin{aligned} A\Delta\mathbf{x} &= \mathbf{b} - A\mathbf{x} \\ A(-(Q + X^{-1}Z)^{-1}(\mathbf{c} + Q\mathbf{x} - A^T\mathbf{y} - \mu X^{-1}\mathbf{e} - A^T\Delta\mathbf{y}) &= \mathbf{b} - A\mathbf{x} \end{aligned}$$

to obtain

$$A((Q + X^{-1}Z)^{-1}A^T\Delta\mathbf{y}) = \mathbf{b} - A\mathbf{x} + A(Q + X^{-1}Z)^{-1}(\mathbf{c} + Q\mathbf{x} - A^T\mathbf{y} - \mu X^{-1}\mathbf{e}. \tag{6.32}$$

This system of equations is referred to as the normal system.

Equation 6.32 is now an $M \times M$ linear system in M variables, but in order to formulate this linear system the inverse of $(Q + X^{-1}Z)$ must be computed. Let us now consider an alternative to solving for this inverse which may be numerically unstable and may require more computational time. Instead, let

$$V = A(Q + X^{-1}Z)^{-1}.$$

Now, A is likely not a square matrix, so take the transpose

$$V^T = (A(Q + X^{-1}Z)^{-1})^T = ((Q + X^{-1}Z)^{-1})^T A^T = ((Q + X^{-1}Z)^{-1}A^T.$$

Therefore,

$$(Q + X^{-1}Z)^T V^T = A^T.$$

We solve this system of equations for V^T and substitute V into Equation 6.32

$$VA^T\Delta\mathbf{y} = \mathbf{b} - A\mathbf{x} + V(\mathbf{c} + Q\mathbf{x} - A^T\mathbf{y} - \mu X^{-1}\mathbf{e}. \tag{6.33}$$

We will call this system the normal without inverses system of equations.

We will compare all 8 formulations (the original discussed in Section 6.5 and the three reduced systems discussed in 6.6 as well as all four derivations in this section) of the Primal-Dual Interior Point algorithm in terms of complexity in the next section.

6.8. Complexity Analysis

In the formulations of the PDIP described above, solving a system of equations must be performed. The most straightforward way to do this is with Gaussian elimination which takes $\frac{2}{3}n^3 + \mathcal{O}(n^2)$ floating point operations (flops) if the system is of size $n \times n$ [77]. Let us recall the dimensions of our problem: $A_{M\times N}$,$\mathbf{b}_{M\times 1}$, $\mathbf{c}_{N\times 1}$, $Q_{N\times N}$, primal variable $\mathbf{x}_{N\times 1}$, dual slack variable $\mathbf{z}_{N\times 1}$, dual variable $\mathbf{y}_{M\times 1}$.

Now, solving the full $(2M+N)\times(2M+N)$ systems of equations 6.11 and 6.29 will require $\frac{2}{3}(2M+N)^3$ flops. The PDIP algorithm will be dominated by this computation, and thus, to fully solve a quadratic program using the PDIP requires $\mathcal{O}((2M+N)^3 h)$ flops where h is the average number of iterations. We will see shortly, how the number of iterations typically grows with the size of the problem.

To solve the reduced $(M+N)\times(M+N)$ systems of equations 6.20 and 6.31 will require $\frac{2}{3}(M+N)^3$ flops. Again, the computations will be dominated by solving this system, so to fully solve a quadratic program using these reduced systems requires $\mathcal{O}((M+N)^3 h)$ flops. Do note that solving 6.31 does involve computing the inverse of the diagonal matrix X which can be performed by taking

$$X^{-1} = \begin{bmatrix} \frac{1}{x_1} & 0 & 0 & \dots & 0 \\ 0 & \frac{1}{x_2} & 0 & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \dots & & 0 & \frac{1}{x_N} \end{bmatrix}.$$

However, more matrix multiplcation is required to solve 6.20.

Finally, we consider the case where we have reduced the system down to only the one variable $\Delta\mathbf{y}$, which solves an $M \times M$ linear system. In this case, the size of M and N play a large part in the computational complexity. Note that $M < N$. Now, fully solving the quadratic program will likely not be dominated by solving the linear systems 6.22 and **??** (and analogously 6.23 and 6.33). Therefore, we must also consider solving for the inverses (or other linear systems) in the problem as well as matrix multiplication. Thus, the computational complexity to fully solve this system is $\mathcal{O}((N^3 + M^3 + M^2N + N^2M)h)$

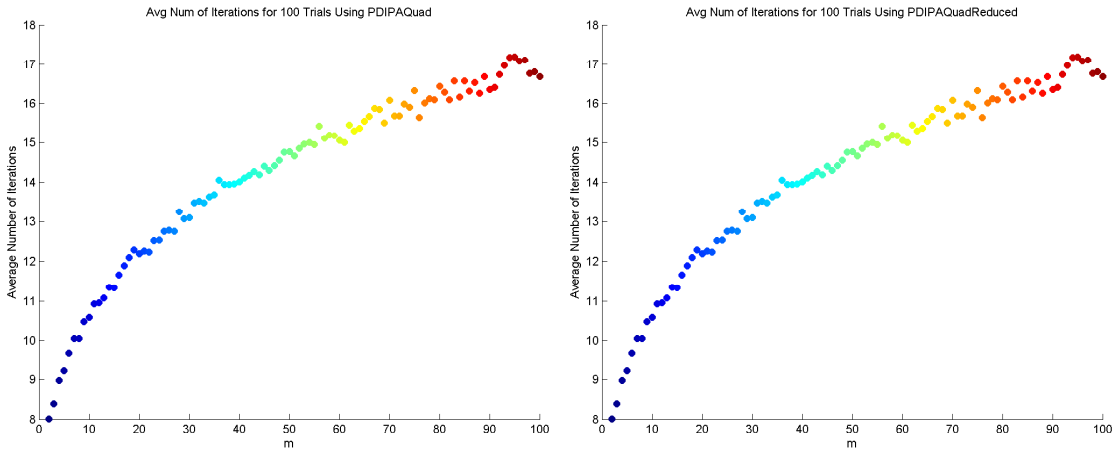
To test these computational complexity arguments, an experiment for varying size systems was considered. We construct a quadratic program where

$$A = \begin{bmatrix} R_{m\times m} & | & I_{m\times m} \end{bmatrix}$$

(with R an $m \times m$ random matrix), $\mathbf{b}_{m\times 1}$ with random entries, $\mathbf{c}_{2m\times 1}$ where the first m entries are chosen randomly and the remaining are 0,

$$\hat{Q}_{m\times m} = \begin{bmatrix} R'_{m\times m} & 0_{m\times m} \\ 0_{m\times m} & 0_{m\times m} \end{bmatrix}$$

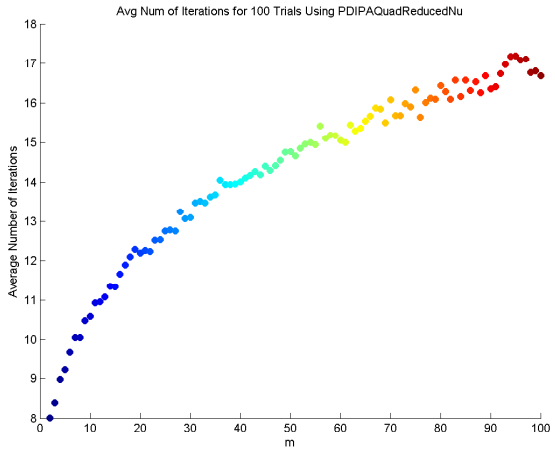
and $Q = \hat{Q}^T\hat{Q}$, a symmetric positive semi-definite matrix. We have constructed this experiment in this manner in order to pad random inequality constraints with slack variables, making the constraints all equality constrained. We vary m from 2 to 100, and run 100 trials for each choice of m. For each trial and each PDIP derivation, we record the number of iterations required to solve and the CPU time in order to determine the averages. See Figures 6.1, 6.2, 6.3, and 6.4 to compare the average number of iterations and CPU time. We see that across all methods the number of iterations is comparable, Figure 6.5a,



(A) PDIP

(B) Reduced

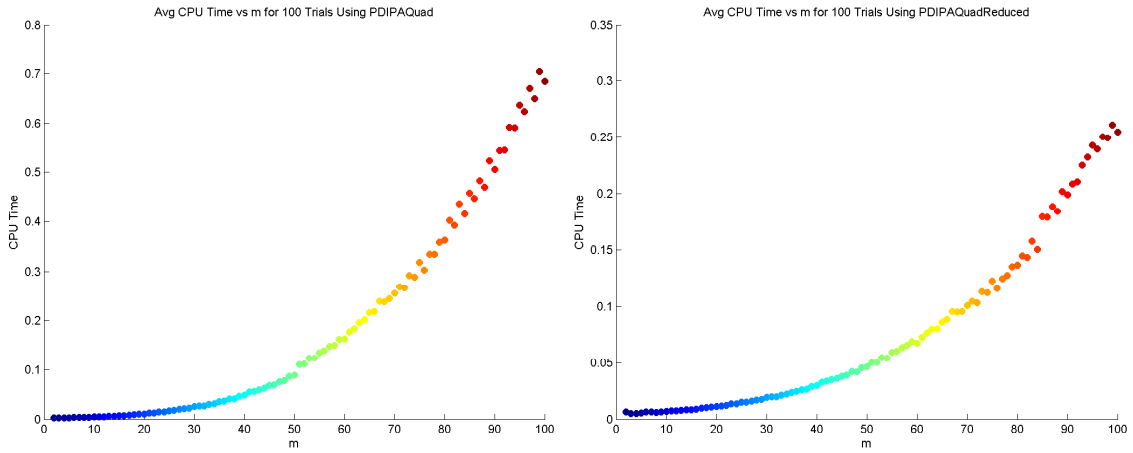
(C) Normal



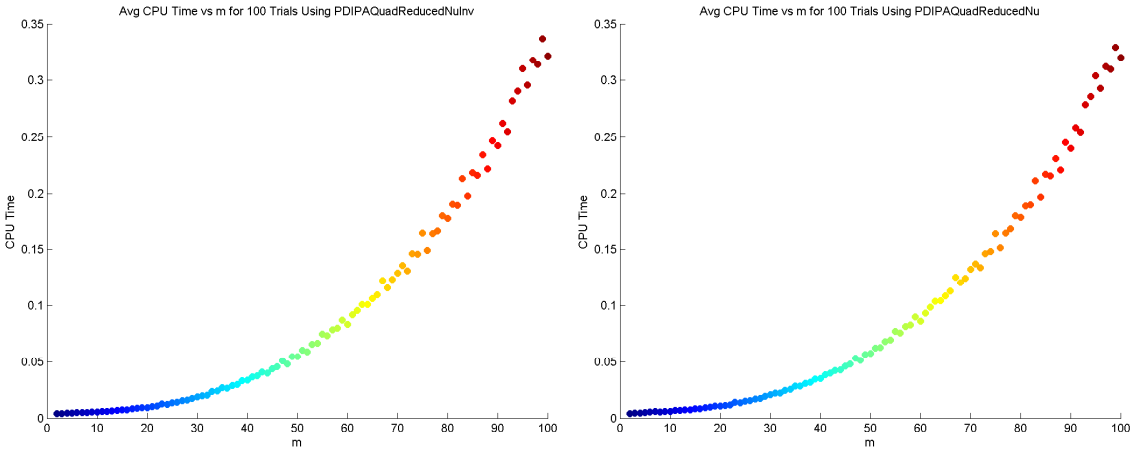
(D) Normal without Inverses

FIGURE 6.1. Average number of iterations for 100 trials, varying the size of system solves to be $m \times 2m$, with derivations as described in Section 6.6.

while the reduced and normal cases have the fastest time. Notice that the normal version avoiding inverses discussed in Section 6.7 seems to be slightly faster than all other methods, Figure 6.5b. It should be noted, of course, that the most stable method is the full cases, and while there may be speed-up from using the reduced and particularly the normal cases, some instability may arise.



(A) PDIP (B) Reduced

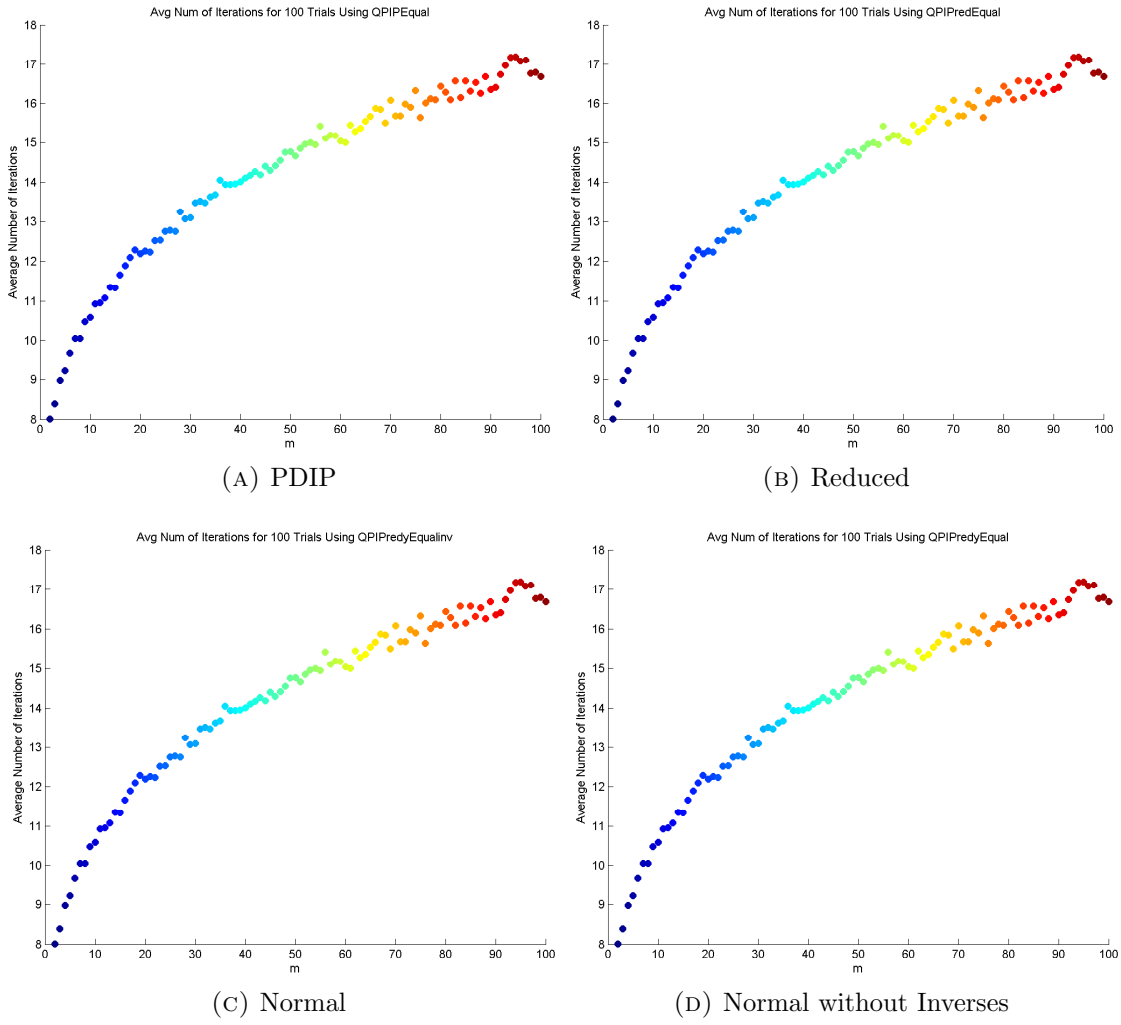


(C) Normal (D) Normal without Inverses

FIGURE 6.2. Average CPU time for 100 trials, varying the size of system solved to be $m \times 2m$, with derivations as described in Section 6.6.

6.9. Computational Complexity of Sparse LLE

We will now return to our consideration of Sparse LLE, but first, we consider the computational complexity of standard LLE as discussed in [71]. In standard LLE, the computational complexity to determine the nearest neighbors scales in the worst case as $\mathcal{O}(Dp^2)$ where D is the ambient dimension of our input data and p is the number of data points. The complexity to solve the least squares problems to determine the weights requires $\mathcal{O}(DpK^3)$ number of operations required to solve the $K \times K$ set of linear equations for all points. The most



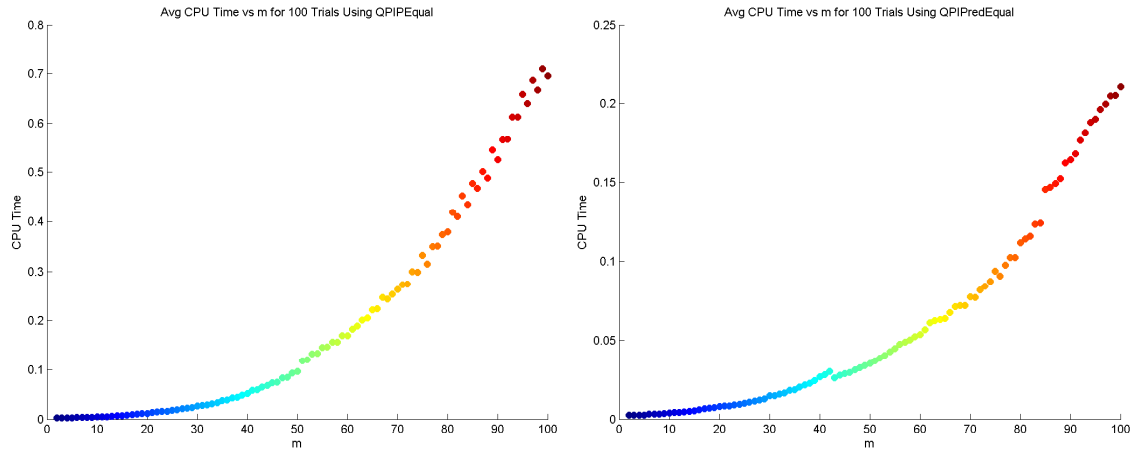
(A) PDIP (B) Reduced

(C) Normal (D) Normal without Inverses

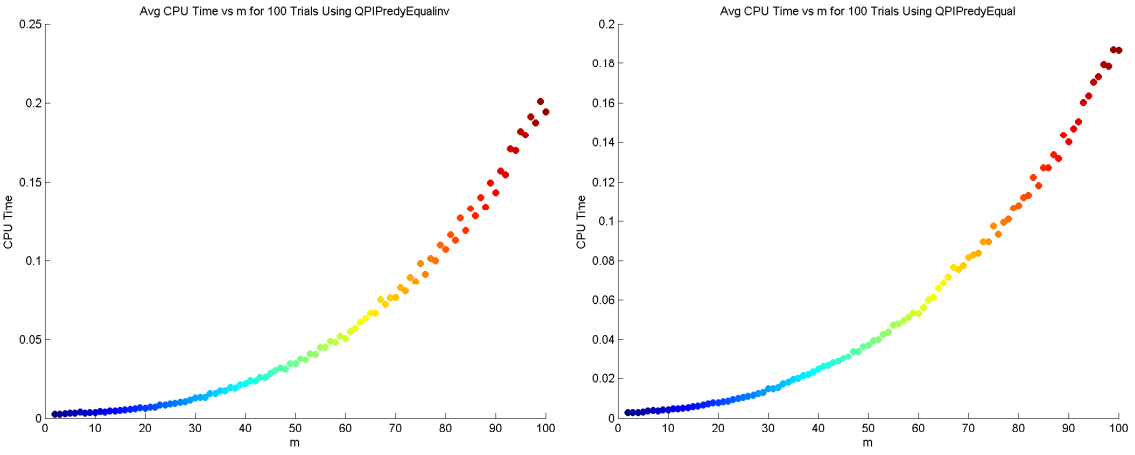
FIGURE 6.3. Average number of iterations for 100 trials, varying the size of system solved to be $m \times 2m$, with derivations as described in Section 6.7.

computationally expensive step in LLE is to solve the eigenvector problem which scales as $\mathcal{O}(dp^2)$ where d is the dimension of the embedding data. Methods to solve sparse symmetric eigenproblems, however, reduce complexity to subquadratic in p.

In terms of the computational complexity of Sparse LLE, the first step remains the same and the third step also remains the same, simply a fewer number of nonzero weights are involved in the eigensolver. Thus, using sparse eigensolvers, Sparse LLE will have a smaller computational cost than standard LLE for the final step. Our focus, is on the second step to



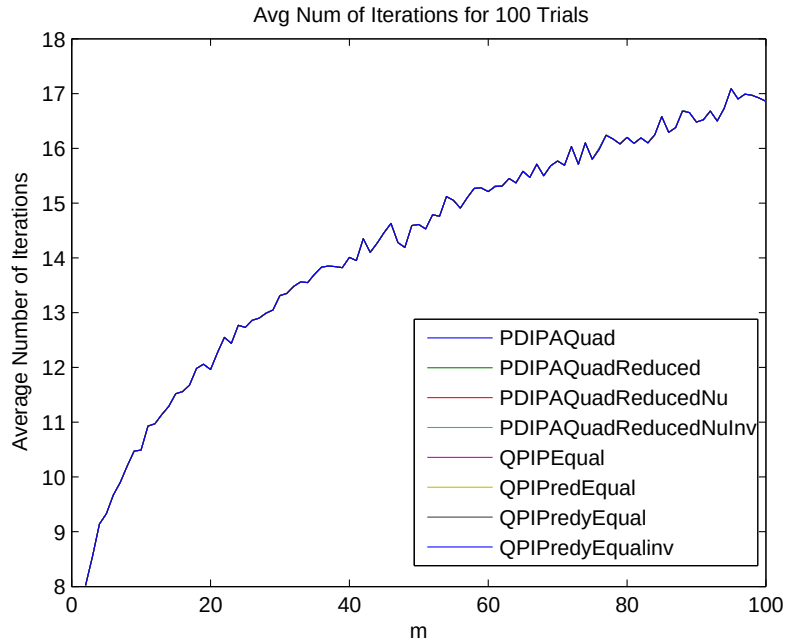
(A) PDIP (B) Reduced



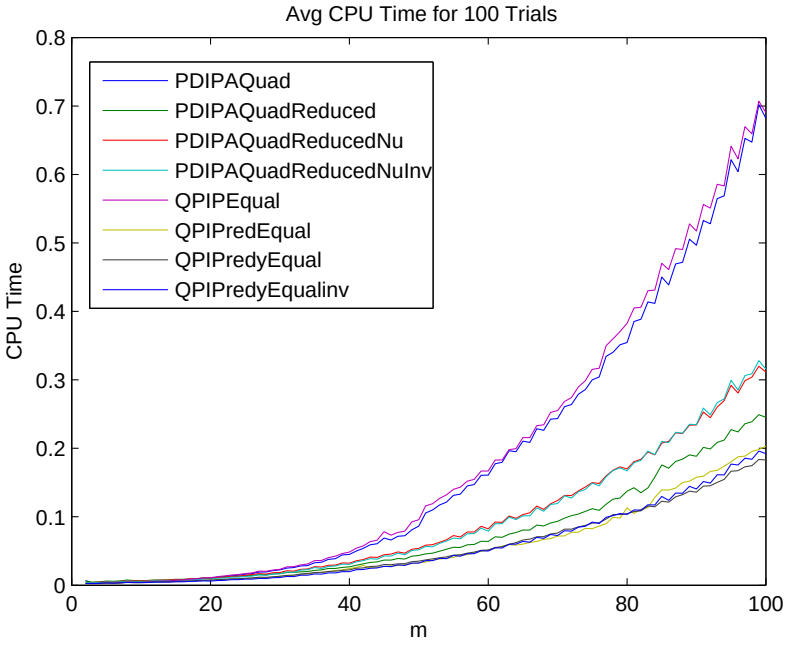
(C) Normal (D) Normal without Inverses

FIGURE 6.4. Average CPU time for 100 trials, varying the size of system solved to be $m \times 2m$, with derivations as described in Section 6.7.

determine the weights. In sparse LLE, the number of decision variables to solve the quadratic program for each i is $N = 2K$ and the number of equality constraints is $M = 1$, the sum to one constraint. Therefore, the full system of equations formed in LLE is $(2M + N) \times (2M + N) = (4K + 1) \times (4K + 1)$. Therefore, the number of operations required to solve for all of the weights is $\mathcal{O}(Dp(4K + 1)^3 h)$, where h is the average number of iterations required to solve each problem. Although this is more computationally complex than standard LLE,



(A) Iterations



(B) CPU

FIGURE 6.5. Average number of iterations and CPU time for 100 trials, varying the size of system solved to be $m \times 2m$. Comparing all derivations discussed in Sections 6.6 and 6.7.

Sparse LLE has the benefit that nearest neighbors are chosen appropriately for each nearest neighbor.

CHAPTER 7

Detecting the Vertices of a Convex Hull Encasing a Point Cloud of Data

7.1. Introduction

The convex hull of a set of points in Euclidean space is defined as the smallest convex set (or polytope) containing the points. Finding efficient algorithms to compute the convex hull is one of the fundamental problems in computational geometry. Its influential role is due to its wide range of uses in both pure and applied mathematics. This includes applications in areas as diverse as algebraic geometry, number theory, combinatorics, pattern recognition, end member detection, data visualization, path planning, and geographical information systems [63]. While computing the set of vertices in a convex hull can be carried out fairly efficiently, describing the full collection of facets in a convex hull becomes impractical for high dimensional data sets due to their rapid growth. In many applications, extremal and nearly extremal data points are the most fundamental and provide clues as to the nature of the data. For instance, the vertices of a convex hull are useful in the sense that any point in the data set can be written as their convex combination. In addition, the vertices are the extrema of linear functionals on the data and potentially capture novel observations in the data. However, under the presence of noise, the true vertex set may be difficult to determine and one should expand the list of extremal candidates to points lying near the boundary of the convex hull.

The problem of determining the vertices of the convex hull of a finite set of points has a long history in computational geometry. The planar case is well-studied and includes methods developed since 1970 such as Gift Wrapping, Quickhull, Graham's Algorithm, Divide and Conquer, and the Incremental Algorithm to name a few [63]. Several of these algorithms may be extended to the 3-dimensional case as well as to higher dimensions. The Quickhull algorithm described in [5] can be used to determine convex hulls in n-dimensions. The problem becomes much more difficult as the dimension increases because the hull itself may have a very large size [55]. The worst case complexity estimate for the Quickhull algorithm grows exponentially with the number of dimensions, largely due to the number of facets contained in the convex hull [5]. In practice, we observe this algorithm failing to finish, e.g., on 100 points in 14 dimensions. Thus, in this chapter, we present an optimization problem and algorithm for stratifying high dimensional point cloud data based on proximity to the boundary of the data's convex hull.

7.2. Thought Experiment

We will first develop a geometric intuition regarding a convex hull. The formal definition is as follows:

Definition 7. *The convex hull of a set C is the set of all convex combinations of points in C:*

$$\mathcal{H}(C) = \{\theta_1 \boldsymbol{x}_1 + \cdots + \theta_k \boldsymbol{x}_k | \boldsymbol{x}_i \in C, \theta_i \geq 0, i = 1, \ldots, k, \theta_1 + \cdots + \theta_k = 1\}$$

where k is the cardinality of the set C.

Another interpretation of a convex combination is a weighted average of the points with θ_i representing the proportion of x_i present [15]. The convex hull, $\mathcal{H}(C)$, is the smallest convex

set that contains C. Geometrically, a set is convex if for any two points in the set, the line segment joining them is also in the set. In 2-dimensions, the convex hull of a point cloud of data will be a convex polygon whose vertices are some points in the data set. Analogously in 3-dimensions, it is a convex polyhedron, and in general, the convex hull of a finite set of data points is a convex polytope.

In what follows it will be useful to work with extreme points of a convex hull of a point cloud of data. The set of extreme points of a polyhedron is equivalent to the set of vertices [13].

Definition 8. *A vector $\boldsymbol{x} \in \mathcal{H}(C)$ is an extreme point if we cannot find two vectors $\boldsymbol{y}, \boldsymbol{z} \in \mathcal{H}(C)$, both different from $\boldsymbol{x}$, and a scalar $\lambda \in [0, 1]$, such that $\boldsymbol{x} = \lambda \boldsymbol{y} + (1 - \lambda)\boldsymbol{z}$.*

To illustrate this further, observe a point cloud of data such as in Figure 7.1a. We will consider perfectly rewriting each data point as a linear combination of its K nearest neighbors. This is possible if K is chosen such that $K \geq D$, where D is the dimension of the data set and there are D linearly independent neighbors. If we arbitrarily choose $K = 7$ and consider a point, $\mathbf{x}$, in the interior of this point cloud of data, then we see that this point falls within the convex hull of its nearest neighbors, see Figure 7.1b. Therefore, we can perfectly reconstruct point $\mathbf{x}$ as a convex combination of its nearest neighbors, i.e. all coefficients (or weights) associated to the nearest neighbors are positive. However, if we choose to reconstruct the vertex point $\mathbf{y}$, we observe that it does not fall within the convex hull of its nearest neighbors, see Figure 7.1c. Therefore, in order to perfectly reconstruct a vertex point by its nearest neighbors at least one of the coefficients of the linear combination will need to be negative. This results in the corollary following from the above definition

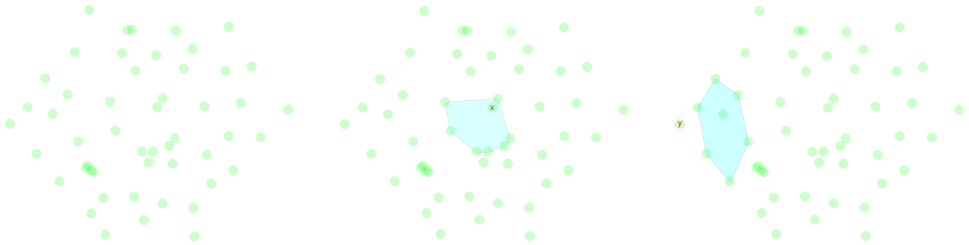


FIGURE 7.1. Illustration of reconstructing an interior point versus a vertex by a set of nearest neighbors.

Corollary 1. *A point is a vertex of a convex hull if and only if perfectly rewriting this point as a linear combination of its $K = p - 1$ nearest neighbors requires at least one negative coefficient where p is the number of points in the data set.*

Using this result, an algorithm to determine the vertices of a convex hull is presented in the next section.

7.3. Optimization Problem

In this section, we discuss an algorithm to determine the vertices of a convex hull from a point cloud of data. To do this, we propose to solve the optimization problem

$$\begin{aligned} &\underset{\mathbf{w}_i}{\text{minimize}} && \gamma\|\mathbf{w}_i\|_2^2 + \lambda\|\mathbf{w}_i\|_1 + \|\mathbf{x}_i - \sum_{j\in N_i} w_{ij}\mathbf{x}_j\|_2^2 \\ &\text{subject to} && \sum_{j\in N_i} w_{ij} = 1 \end{aligned} \tag{7.1}$$

for each point $\mathbf{x}_i$. The solution consists of a representation of each point as a K-dimensional weight vector $\mathbf{w}_i$. N_i is the index set associated to the K nearest neighbors of point $\mathbf{x}_i$ where,

if possible, $K \geq D$. We will describe how this optimization problem generates vertices of the convex hull of the data by analyzing the role of each term in the objective function.

7.3.1. Data Representation. To begin consider the related optimization problem

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \|\mathbf{x} - \sum_{j \in N} w_j \mathbf{x}_j\|_2^2 \\ \text{subject to} \quad & \sum_{j \in N} w_j = 1 \end{aligned} \tag{7.2}$$

where, for simplicity, we have surpressed the notation expressing dependence on data point i.

The effect of this optimization problem is to write each point as the sum of other points in the convex hull. In particular, we want to express each data point as a linear combination of its nearest neighbors. At optimality, the residual (objective function of Equation 7.2) will be as small as possible. If $K \geq D$, then we may perfectly reconstruct any $\mathbf{x} \in C$, but this solution is not unique, i.e., there is an infinite family of weights that all produce zero error.

From this family of solutions that minimize the residual, we add further terms to the objective function to uncover solutions with desriable properties.

7.3.2. POSITIVITY. Consider the optimization problem

$$\begin{aligned} &\underset{\mathbf{w}}{\text{minimize}} && \|\mathbf{w}\|_1 \\ &\text{subject to} && \sum_{j \in N_i} w_j = 1 \end{aligned} \tag{7.3}$$

which requires that a set of weights that sum to one must have a minimum ℓ_1 norm.

A necessary condition for a vector w to satisfy this optimzation problem is for it to be non-negative. Below, it will be seen that this term serves to restrict the representation of x by its neighbors to be a convex combination of points in C. In other words, positivity induces convexity.

To see why the ℓ_1 norm favors positivity we use a standard trick in optimization to remove the absolute value from our objective function by introducing nonnegative variables $w_j^+ \geq 0$ and $w_j^- \geq$ such that $w_j = w_j^+ - w_j^-$ and $|w_j| = w_j^+ + w_j^-$. Then the optimization problem can be rewritten as

$$\begin{aligned} &\underset{\mathbf{w}}{\text{minimize}} && \sum_{j \in N} w_j^+ + w_j^- \\ &\text{subject to} && \sum_{j \in N} w_j^+ - w_j^- = 1 \\ &\text{and} && w_j^+ \geq 0, w_j^- \geq 0 \end{aligned}$$

Rewriting the equality constraint as

$$\sum_{j \in N} w_j^+ = 1 + \sum_{j \in N} w_j^-$$

the objective function may be rewritten as

$$\underset{\mathbf{w}}{\text{minimize}} \quad 1 + 2 \sum_{j \in N} w_j^- .$$

Then, we see that the optimal solution to this problem has the property that $w_j^- = 0$ for all $j \in N$, and hence, the solution $w_j = w_j^+ \geq 0$.

Notice in optimization problem 7.1, a parameter λ is included in this ℓ_1 regularization term. For any point that can be written as a convex combination, only the constant λ is added to the objective function as the 1-norm of positive weights summing to one will of course be one. Hence, for the case $K > D$ the solution remains non-unique.

In the context of compressed sensing, adding an ℓ_1 term to the reconstruction objective function is sparsity inducing because it serves as a proxy for the ℓ_0 norm [19]. Also, we have seen that one can also induce sparsity for this local representation problem if a data-weighted scaling is included, see Chapter 5.

7.3.3. Uniformity. Now consider the optimization problem

(7.4)

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \|\mathbf{w}\|_2^2 \\ \text{subject to} \quad & \sum_{j \in N} w_j = 1 \end{aligned}$$

Using Lagrange multipliers, we will show that the solutions to this optimization problem satisfy weights with a uniform distribution. We form the Lagrangian

$$L(\mathbf{w}, \lambda) = w_1^2 + w_2^2 + \ldots + w_K^2 - \lambda(w_1 + w_2 + \ldots + w_K - 1)$$

where $\lambda > 0$. Then differentiating with respect to each variable and equating to zero we obtain

$$\frac{\partial L}{\partial w_i} = 2w_i - \lambda = 0 \text{ for } i = 1, \ldots, K \tag{7.5}$$

$$\frac{\partial L}{\partial \lambda} = w_1 + w_2 + \ldots + w_K - 1 = 0 \tag{7.6}$$

Thus, each $w_i = \frac{\lambda}{2}$ and substituting into Equation 7.6, we obtain

$$K(\lambda/2) - 1 = 0.$$

Therefore, we see that the vector that minimizes the square Euclidean norm is the one with entries all $1/K$. Thus,

$$w_j = \kappa$$

where κ is fixed for all j, and hence we say that the weights are uniform in size.

The addition of this ℓ_2 norm term further provides the important result that the solution is unique.

7.3.4. Putting it all together. We now perform a balancing act with the optimization problem. We seek to represent a point by its nearest neighbors but such that all the weights are positive, if possible, and favor solutions with a uniform distribution of weights.

We suggest using parameters $\gamma << \lambda << 1$. With these parameter choices, the most emphasis is placed on reconstructing each data point as best as possible (right term), then finding weights reflecting a convex combination if possible (middle term), and finally from all possible solutions that perform the first two steps use the left term to regularize. Therefore, each term in our objective function enforces weights favoring (from left to right) uniformity, convexity, and proximity, see Equation 7.7.

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \overbrace{\gamma\|\mathbf{w}\|_2^2}^{\text{Uniformity}} + \overbrace{\lambda\|\mathbf{w}\|_1}^{\text{Convexity}} + \overbrace{\|\mathbf{x} - \sum_{j\in N} w_j \mathbf{x}_j\|_2^2}^{\text{Proximity}} \\ \text{subject to} \quad & \underbrace{\sum_{j\in N} w_j = 1}_{\text{Convex Combination}} \end{aligned} \tag{7.7}$$

Therefore, for an appropriate choice of parameters, any point that has a negative weight cannot be represented in any other manner, and thus, a point with at least one negative weight must be a vertex of the convex hull.

To solve this objective function, we rewrite using the technique described in Section 5.2 to remove the absolute value from our objective function by introducing nonnegative variables w_j^+ and w_j^- such that $w_j = w_j^+ - w_j^-$ and $|w_j| = w_j^+ + w_j^-$.

$$\gamma\|\mathbf{w}|_2^2 + \lambda\|\mathbf{w}\|_1 + \|\mathbf{x}_i - \sum_{j\in N_i} w_j\mathbf{x}_j\|_2^2$$

$$= \gamma\mathbf{w}^T\mathbf{w} + \lambda\sum_{j\in N_i}||w_j| + < \mathbf{x}_i - \sum_{j\in N_i} w_j, \mathbf{x}_i - \sum_{k\in N_i} w_k^{(i)} >$$

$$= \gamma < \mathbf{w}^+ - \mathbf{w}^-, \mathbf{w}^+ - \mathbf{w}^- > +\lambda\sum_{j\in N_i}(w_j^+ + w_j^-) + \|\mathbf{x}_i - \sum_{j\in N_i}(w_j^+ - w_j^-)\mathbf{x}_j\|_2^2$$

$$= \gamma(< \mathbf{w}^+, \mathbf{w}^+ > - < \mathbf{w}^+, \mathbf{w}^- > - < \mathbf{w}^-, \mathbf{w}^+ > + < \mathbf{w}^-, \mathbf{w}^- >) + \lambda\sum_{j\in N_i}(w_j^+ + w_j^-) +$$

$$< \mathbf{x}_i - \sum_{j\in N_i}(w_j^+ - w_j^-)\mathbf{x}_j, \mathbf{x}_i - \sum_{j\in N_i}(w_j^+ - w_j^-)\mathbf{x}_j >$$

$$= \gamma\left(\sum_{j\in N_i}(w_j^+)^2 - 2\sum_{j\in N_i} w_j^+w_j^- + \sum_{j\in N_i}(w_j^-)^2\right) + \lambda\sum_{j\in N_i}(w_j^+ + w_j^-) + \mathbf{x}_i^T\mathbf{x}_i -$$

$$2\sum_{j\in N_i}(w_j^+ - w_j^-)\mathbf{x}_i^T\mathbf{x}_j + \sum_{j,k\in N_i}(w_j^+ - w_j^-)(w_k^+ - w_k^-)\mathbf{x}_j^T\mathbf{x}_k$$

$$= \gamma\left(\sum_{j\in N_i}(w_j^+)^2 - 2\sum_{j\in N_i} w_j^+w_j^- + \sum_{j\in N_i}(w_j^-)^2\right) + \sum_{j,k\in N_i}\mathbf{x}_j^T\mathbf{x}_k(w_j^+w_k^+ - w_j^+w_k^- - w_j^-w_k^+ + w_j^-w_k^-)$$

$$+ \sum_{j\in N_i}\left(-2\mathbf{x}_i^T\mathbf{x}_j + \lambda)\right)(w_j^+ - w_j^-)$$

$$+ \mathbf{x}_i^T\mathbf{x}_i$$

Therefore, we see this objective function is a quadratic function, and thus, we can rewrite 7.1 as a quadratic program of the form

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{w}^TQ\mathbf{w} + \mathbf{c}^T\mathbf{w} \\ \text{subject to} \quad & A\mathbf{w} = \mathbf{b} \\ & \mathbf{w} \geq 0 \end{aligned} \tag{7.8}$$

where the constant term is dropped. The optimization problem 7.8 is in $n = 2K$ variables with equality constraint, $A\mathbf{w} = \mathbf{b}$, an $m \times n$ system where $m = 1$. More explicity,

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}^+ \\ \mathbf{w}^- \end{bmatrix} = \begin{bmatrix} w_i^{(1)+} \\ w_i^{(2)+} \\ \cdots \\ w_i^{(K)+} \\ w_i^{(1)-} \\ w_i^{(2)-} \\ \cdots \\ w_i^{(K)-} \end{bmatrix}$$

where we use the notation $w_i^{(\ell)}$ to denote the weight associated from point $\mathbf{x}_i$ to its ℓth nearest neighbor for $\ell = 1, \ldots, K$. Then, $Q = Q_i$ for each i with

$$Q_i = \begin{bmatrix} \tilde{Q}_i & -\tilde{Q}_i \\ -\tilde{Q}_i & \tilde{Q}_i \end{bmatrix} + E$$

where

$$\tilde{Q}_i = \begin{bmatrix} \mathbf{x}_i^{(1)T}\mathbf{x}_i^{(1)} & \mathbf{x}_i^{(1)T}\mathbf{x}_i^{(2)} & \ldots & \mathbf{x}_i^{(1)T}\mathbf{x}_i^{(K)} \\ \vdots & \ddots & & \vdots \\ \mathbf{x}_i^{(K)T}\mathbf{x}_i^{(1)} & \mathbf{x}_i^{(K)T}\mathbf{x}_i^{(2)} & \ldots & \mathbf{x}_i^{(K)T}\mathbf{x}_i^{(K)} \end{bmatrix}$$

end

$$E = \gamma \begin{bmatrix} I_{K\times K} & -I_{K\times K} \\ -I_{K\times K} & I_{K\times K} \end{bmatrix}.$$

Finally,

$$A = \begin{bmatrix} 1 & \dots & 1 & -1 & \dots & -1 \end{bmatrix}$$

$$\mathbf{b} = 1$$

$$\mathbf{c} = \begin{bmatrix} -2\mathbf{x}_i^T\mathbf{x}_i^{(1)} + \lambda \\ -2\mathbf{x}_i^T\mathbf{x}_i^{(2)} + \lambda \\ \vdots \\ -2\mathbf{x}_i^T\mathbf{x}_i^{(K)} + \lambda \\ 2\mathbf{x}_i^T\mathbf{x}_i^{(1)} + \lambda \\ \vdots \\ 2\mathbf{x}_i^T\mathbf{x}_i^{(K)} + \lambda \end{bmatrix}$$

We choose to solve this problem by using the Primal Dual Interior Point method as describe in Chapter 6 and [13, 15, 79].

This algorithm is summarized below.

Algorithm to Detect Vertices of a Convex Hull

(1) Select a point $\mathbf{x}_i$.

(2) Determine K nearest neighbors. If possible choose $K > D$.

(3) For an appropriate choice of parameters $\gamma << \lambda << 1$, solve

$$\begin{aligned} &\underset{\mathbf{w}}{\text{minimize}} && \gamma\|\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_1 + \|\mathbf{x}_i - \sum_{j\in N_i} w_j\mathbf{x}_j\|_2^2 \\ &\text{subject to} && \sum_{j\in N_i} w_j = 1 \end{aligned} \tag{7.9}$$

(4) Points with at least one negative weight are vertices of the convex hull.

(5) Apply this to each point in C either sequentially or in parallel.

The computational complexity of this algorithm is as follows. Determining the nearest neighbors scales in the worst case as $\mathcal{O}(Dp^2)$ where D is the ambient dimension of our input data and p is the number of data points. To formulate the quadratic program as a linear system of equalities is on the order of $\mathcal{O}(K^2Dp)$. Solving the full $(2M+N)\times(2M+N) = (4K+1)\times(4K+1)$ system of equations requires a computational complexity of $\mathcal{O}(p(4K+1)^3h)$, where h is the average number of iterations required to solve each problem. Thus, this step is on the order of $\mathcal{O}(Dp(4K+1)^3h)$.

We observe that the computational complexity of this algorithm is linear in the number of data points and the ambient dimension of the data. Further note, that as optimization problem 7.1 is solved for each point $\mathbf{x}_i$, this step can be trivially parallelized.

7.4. Implementation

In this section, we implement the algorithm on several point cloud data sets to illustrate the features of the optimization problem described above.

7.4.1. Toy Example. In the first experiment, 50 random points are generated inside the unit square. Using MATLAB's convex hull function, we determine the vertices of the convex hull containing this data set. An additional 10 points were then placed along the boundary of the convex hull of the original 50 points in order to make the data potentially more complicated to analyze; see Figure 7.2. Typically, a larger choice for the number of nearest neighbors, K, is better, particularly if the sample is not dense, in order to eliminate the possibility of interior points forming a clique and becoming isolated or causing confusion

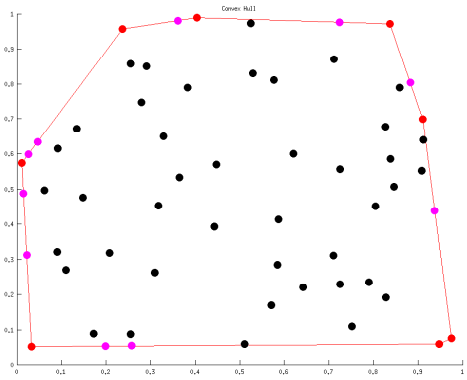
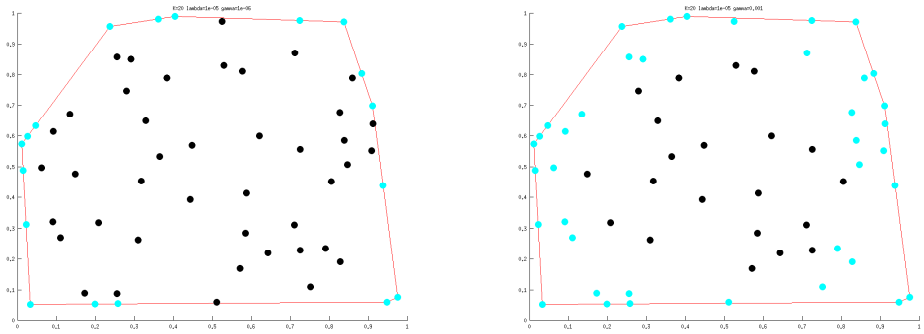


FIGURE 7.2. The data set consists of an initial 50 random points augmented by an additional 10 boundary points (magenta). The vertices (red) are determined by the Quickhull algorithm.

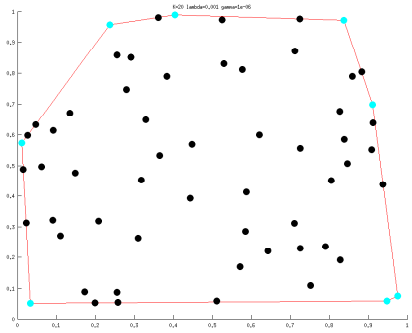
with the actual boundary of the data. A choice of $K = 20$ nearest neighbors was found to work well for this data set. We will see, in this example, that both λ and γ need to be quite small and that λ does in fact need to be larger than γ in order to greater penalize negative weights. Depending on application (i.e. desire to determine boundary of a data set or vertices of a convex hull), different choices for these parameters will be needed.

In Figure 7.3, we observe the effect of varying both γ and λ. We have chosen to denote those points with a negative weight as cyan and those points with only positive weights as black. For the relatively small choices of $\lambda = 10^{-5}$ and $\gamma = 10^{-6}$, we see in Figure 7.3a that all of the boundary points have negative weights, and thus, our algorithm was able to determine the boundary of the data set. Note that if a thicker boundary is desired, then a choice of γ and λ as seen in Figure7.3b would be appropriate. As more emphasis was placed on uniformity in the weights than on convexity, more weights are negative. Note that in Figure 7.3c, only the vertices of the convex hull were represented using negative weights, corresponding precisely with those vertices determined by MATLAB. Finally, note that if λ is chosen to be too large, then too much emphasis will be placed on avoiding negative

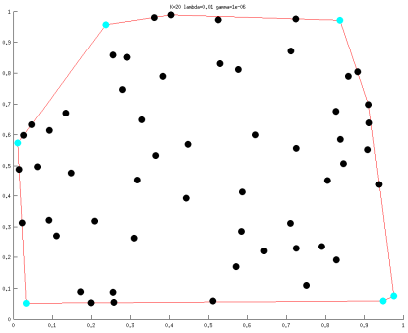


(A) $\lambda = 10^{-5}, \gamma = 10^{-6}$

(B) $\lambda = 10^{-5}, \gamma = 10^{-3}$



(C) $\lambda = 10^{-3}, \gamma = 10^{-6}$



(D) $\lambda = 10^{-2}, \gamma = 10^{-6}$

FIGURE 7.3. Implementation of the algorithm with various parameter choices. Cyan points have a negative weight for the given parameter choices.

weights, and some of those points who should be represented in this manner will not be, see Figure 7.3d.

7.4.2. WEIGHTS ENCODE GEOMETRIC INFORMATION. The structure of the optimization problem results in a very useful property that allows us to infer a considerable amount of information about the relative location of the data points based on the norm of their associated weight vectors. As described above, we have observed above that negative weights are a signature for a vertex. This arises from the ℓ_1-norm constraint in the optimization problem. But there is also additional geometric structure imposed by the ℓ_2-norm given its propensity to drive the components of a weight vector to have a uniform distribution.

The ℓ_2-norm of a weight vector is a measure of the distance of the associated point from the boundary. Hence, we find that the less uniform the weights are, the closer the point is to the boundary. Similarly, the more uniform the weights are, the farther from any boundary the point is.

As previously mentioned, if $\lambda >> \gamma$, then more emphasis is placed on rewriting each data point as a convex combination of its nearest neighbors, whenever possible, and then choosing weights with a uniform distribution. In summary, vertex points having negative weights associated to their nearest neighbors will not yeild uniformly distributed weights and thus, will have a larger 2-norm magnitude in the weights. Similarly, reconstructing points that fall along boundaries by their nearest neighbors will not be represented by uniform weights. Yet, points centered nicely within the convex hull of its nearest neighbors can be reconstructed with uniform weights. This yields a heuristic to determine those points that are vertices of the convex hull as well as the boundaries of the data set.

In Figure 7.4, we observe a plot of each point in our data set colored according to the magnitude of the two norm where blue corresponds to the smallest values and red corresponds to the largest values. Thus, we see that the vertices have the largest Euclidean norm, followed by points along the boundary of the convex hull, and then finally interior points. Note that in practice the Euclidean norms of the weights are robust with respect to the optimization parameters.

7.4.3. Convex Hull Detection of Hypercubes. The next data set we consider is 2000 randomly selected points inside the unit cube and the 8 additional vertices of the cube added to the data set as well. We have chosen $K = 200$ nearest neighbors to represent each point, fixed $\gamma = 10^{-5}$, and explore the effect of varying λ. Cyan indicates those points with

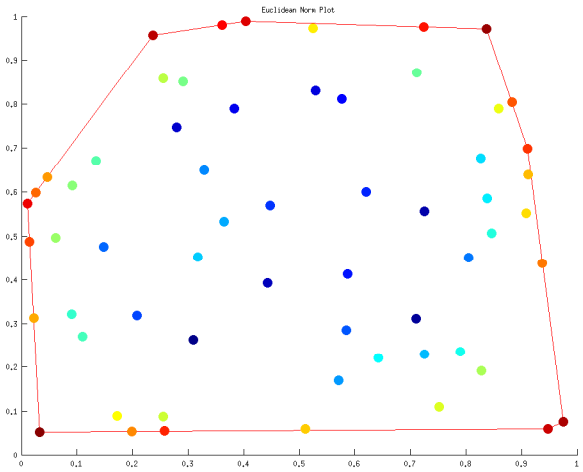
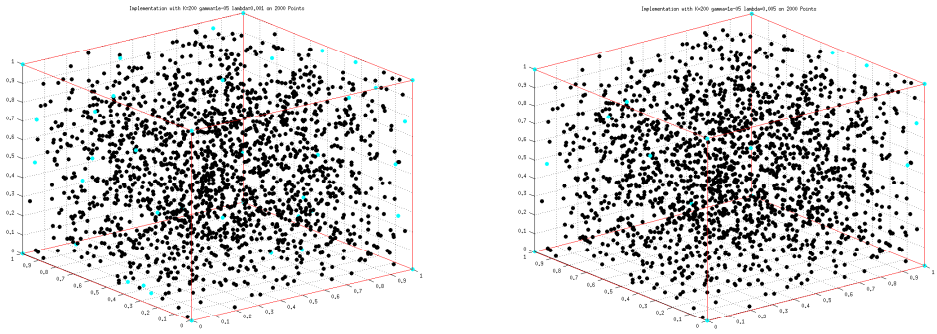


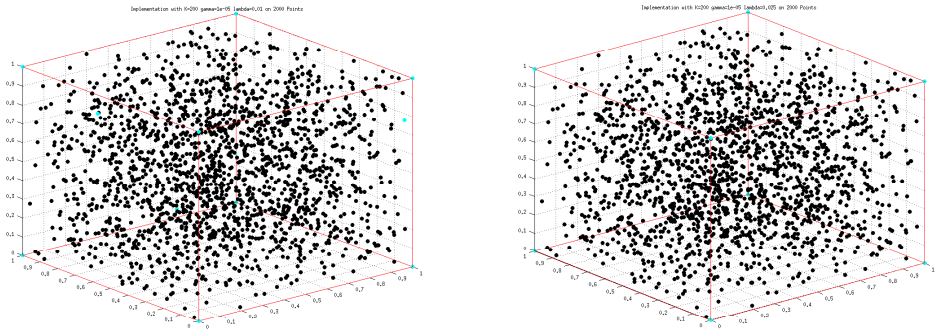
FIGURE 7.4. Data points colored according to magnitude of the two norm where blue corresponds to the smallest values and red corresponds to the largest values.

negative weights, see Figure 7.5. Note for $\lambda = 0.001$, there are 40 points with at least one negative weight, and these fall closest to the vertices, edges, and faces of the cube. As λ is increased the number of points with negative weights decreased, i.e. $\lambda = 0.005$ yields 18, $\lambda = 0.1$ yields 12, and $\lambda = 0.25$ yields 8, the number of vertices of the convex hull of this data set. Similar experiments have been implemented on hypercubes in higher dimensions, and for appropriate parameter choices, the correct number of vertices were determined.

Now, we consider a plot of the Euclidean norms of the weights associated to each point, Figure 7.6. Note that the first points in the data set are the vertices of the cube which correspond to the largest Euclidean norms. Other large norms correspond to points close to the edges and faces of the cube. This is apparent in Figure 7.7 which plots the distance to the boundary against the value of the Euclidean norm of the associated weight vector. We observe that for this data set the 8 vertices, indicated in cyan, have the largest magnitudes. To see this data another way we have plotted each data point in the cube split into 6 color



(A) $\lambda = 0.001$ (B) $\lambda = 0.005$



(C) $\lambda = 0.01$ (D) $\lambda = 0.025$

FIGURE 7.5. Plot of 2000 points randomly selected points with the 8 additional vertices of the cube added. Cyan points indicate negative weight, varying parameter λ.

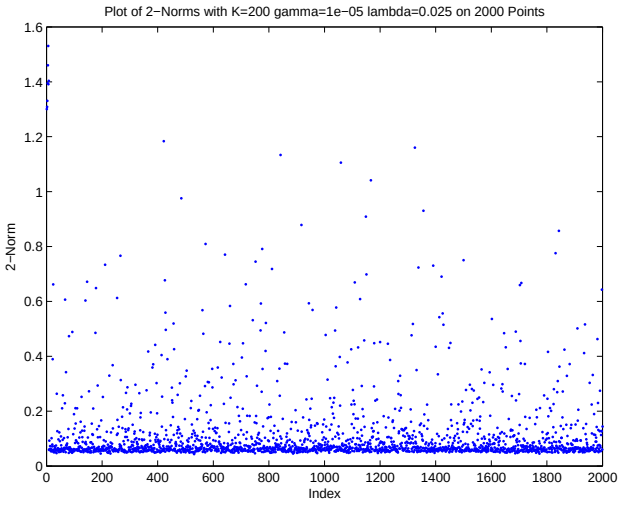


FIGURE 7.6. Plot of Euclidean norm of weights associated to each point with parameters $\gamma = 10^{-5}$ and $\lambda = 0.025$.

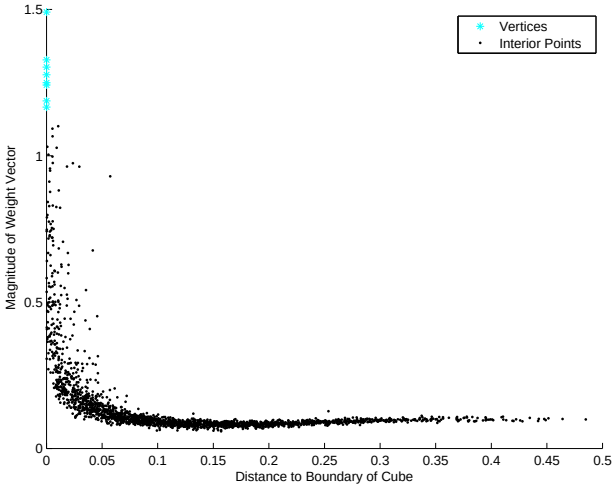


FIGURE 7.7. Plot of distance to the boundary of the cube versus the Euclidean norm of weights associated to each point with parameters $\gamma = 10^{-5}$ and $\lambda = 0.025$.

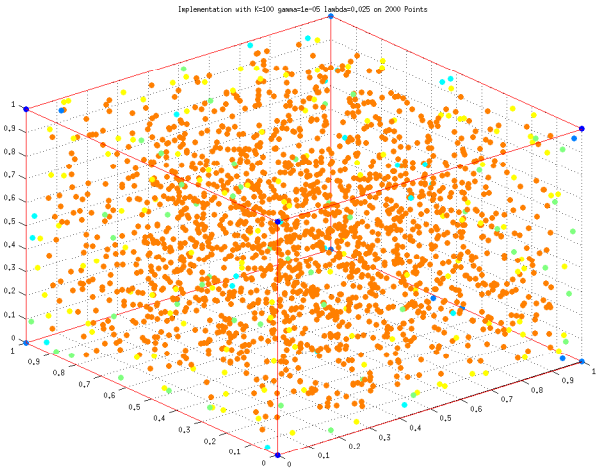


FIGURE 7.8. Plot of each point in the cube colored according to magnitude of the corresponding weight vector with $0 \leq ||\mathbf{w}||_2^2 \leq 0.25$ orange, $0.25 \leq ||\mathbf{w}||_2^2 \leq 0.5$ yellow, $0.5 \leq ||\mathbf{w}||_2^2 \leq 0.75$ green, $0.75 \leq ||\mathbf{w}||_2^2 \leq 1$ cyan, $1 \leq ||\mathbf{w}||_2^2 \leq 1.25$ light blue, $1.25 \leq ||\mathbf{w}||_2^2 \leq 1.5$.

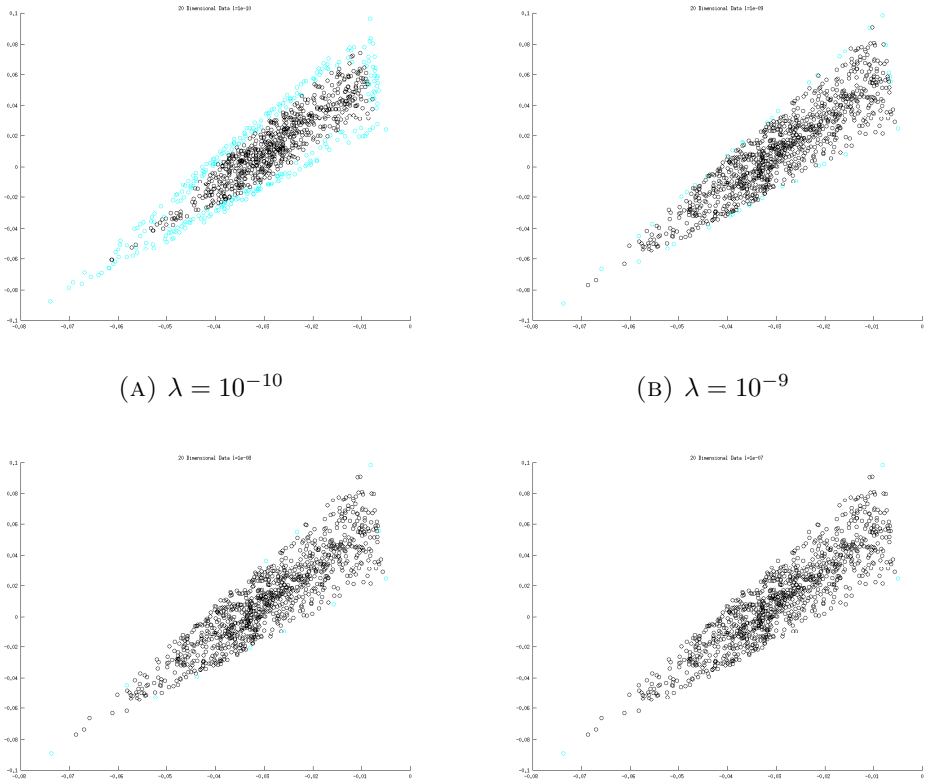
chunks corresponding to the magnitude of the weight vector: $0 \leq ||\mathbf{w}||_2^2 \leq 0.25$ orange, $0.25 \leq ||\mathbf{w}||_2^2 \leq 0.5$ yellow, $0.5 \leq ||\mathbf{w}||_2^2 \leq 0.75$ green, $0.75 \leq ||\mathbf{w}||_2^2 \leq 1$ cyan, $1 \leq ||\mathbf{w}||_2^2 \leq 1.25$ light blue, $1.25 \leq ||\mathbf{w}||_2^2 \leq 1.55$ blue. See Figure 7.8.

7.4.4. Convex Combination of Chemical Signatures. The final data set we consider in this study contains the three 20-dimensional signatures of three chemical simulants, Glacial Acetic Acid (GAA), Methyl Salicylate (MeS), and Triethyl Phosphate (TEP). A data set of 1003 points was created by randomly sampling 1000 points inside the convex hull of these three data points (i.e. forming a convex combination) and including the three points themselves. The data set then consists of a triangle in 20-dimensional space. We implement our algorithm designed to find the boundaries and vertices of the convex hull, and observe that even in this higher-dimensional space, for appropriate parameter choices, our algorithm is able to uncover this information. We have selected $K = 50$ nearest neighbors and $\gamma = 10^{-10}$ and again vary λ.

See Figure 7.9. Using PCA, we project the data set down into $\mathbb{R}^2$ for visualization purposes. Again cyan points indicate those with negative weights. We see those points with negative weights indicate for a choice of $\lambda = 10^{-10}$ a thick boundary of the data set, $\lambda = 10^{-9}$ a thinner boundary of the data set, $\lambda = 10^{-8}$ a small sampling of points along the boundary of the data set, and $\lambda = 10^{-7}$ precisely the vertices of the convex hull. Thus, even in this higher dimension, our algorithm is able to uncover both the boundaries of a data set and the vertices of a convex hull correctly.

7.5. Conclusion

We have proposed an optimization problem that can be used to identify vertices of the convex hull of a data set in high dimensions. The optimization problem consists of three components. The first term concerns representing a point by its neighbors. The other two components, ℓ_1 and ℓ_2 penalty terms, are used to encode geometric structure into the weight



(A) $\lambda = 10^{-10}$ (B) $\lambda = 10^{-9}$

(C) $\lambda = 10^{-8}$ (D) $\lambda = 10^{-7}$

FIGURE 7.9. Implementation of algorithm with various parameter choices on convex combination created using chemical signatures. Points represented in cyan have a negative weight.

vector $\mathbf{w}$ associated with a data point $\mathbf{x}$. We observe that the ℓ_2-norm of a weight vector is a measure of the distance of the associated point from the boundary.

In the extreme, a point cannot be a vertex unless it has a negative weight. By adusting the parameters, we can reduce the set of weights with negative components until only the vertices remain. The computational complexity of the algorithm grows only linearly with dimension and number of data points.

CHAPTER 8

Strengthening of Topological Signals in Persistent Homology through Vector Bundle Based Maps

8.1. Introduction

The need to efficiently extract critical information from large data sets has been growing for decades and is central to a variety of scientific, engineering and mathematical challenges. In many settings, underlying constraints on the data allow it to be considered as a sampling of a topological space. It is a fundamental problem in topological data analysis to develop theory and tools for recovering a topological space from a noisy, discrete sampling. The tools that one might choose to use on a given problem depend on the density, quality, and quantity of the data, on the ambient space from where the sampling is drawn, and on the complexity of the topological space as a sub-object of an ambient space. In this paper, we will focus on data consisting of points sampled from an algebraic variety (the zero locus of a system of polynomials). The data points are obtained using the tools of numerical algebraic geometry. Derived from techniques in homotopy continuation, numerical algebraic geometry allows one to use numerical methods to cheaply sample a large collection of low-noise points from an algebraic set. Persistent homology (PH) allows one to use such a sample to gain insight into the topological structure of the algebraic variety. Implementations of persistent homology are readily available and have been used in a variety of applications, ranging from the analysis of experimental data to analyzing the topology of an algebraic variety.

However, as with any algorithm, there are computational limitations. Generally, the time and space required for the persistence computation grows rapidly with the size of the input sample, so the maximum size of a sample is limited. Often, applications of PH start with noisy, real-world data, which may also be limited in size [59]. However, our consideration begins with effectively unlimited, arbitrarily accurate data. Experience shows that as one increases the sample size of a fixed space, the quality of the topological signals produced by PH improves. Since the computational complexity of persistent homology limits the size of a sample, methods of preprocessing data that improve the topological signal, without increasing the sample size, are desirable.

In this paper, we consider how topological re-embeddings affect the topological signal obtained from persistent homology. First, a construction of PH and the inherent challenges of interpreting its output is briefly introduced. Then, we will provide details about the setting in which we have applied this embedding technique, using computational topology to analyze projective algebraic varieties. Lastly, results for a specific example are displayed and interpreted.

8.2. Background

8.2.1. Persistent Homology. Beginning with a finite set of data points, which are viewed as a noisy sampling of a topological space, assume one has a way of building the matrix of pairwise distances between points in the data set. From this distance matrix, one constructs a nested sequence of simplicial complexes indexed by a parameter t. Fixing a field $\mathbb{K}$, for each simplicial complex, one builds an associated chain complex of vector spaces over $\mathbb{K}$. The i^{th} homology of the chain complex is a vector space and its dimension corresponds to the i^{th} Betti number, $\beta_i(\mathbb{K})$, of the corresponding topological space. For each pair $t_1 < t_2$,

there is a pair of simplicial complexes, S_{t_1} and S_{t_2}, and an inclusion map $j : S_{t_1} \hookrightarrow S_{t_2}$. This inclusion map induces a chain map between the associated chain complexes which further induces a linear map between the corresponding i^{th} homology vector spaces. For each i, the totality of the collection of i^{th} homology vector spaces and induced linear maps can be encoded as a graded $\mathbb{K}[t]$-module known as the persistence module. The i^{th} bar code is a way of presenting the invariant factors of the persistence module. As the invariant factors of the persistence module directly relate to the Betti numbers, from the bar code one can visualize the Betti numbers as a function of the scale, t, and can visualize the number of independent homology classes that persist across a given time interval $[t_i, t_j]$. For foundational material and overviews of computational homology in the setting of persistence, see [20, 33, 34, 39, 51, 86, 87].

One commonly used method for building a nested sequence of simplicial complexes from a distance matrix is through a *Vietoris-Rips* complex [39]. This is done by first building the 1-skeleton of the simplicial complex then determining the higher dimensional faces as the clique complex of the 1-skeleton. More precisely, fix $t > 0$, a collection of points X, and a metric, $d(x_i, x_j)$ for x_i, $x_j \in X$. The 1-skeleton of the Vietoris-Rips complex, $C_t(X)$, is defined by including the edge $x_i x_j \in C_t(X)$ if $d(x_i, x_j) \leq t$. A higher dimensional face is included in $C_t(X)$ if all of its lower dimensional sub-faces are in $C_t(X)$. In other words, the abstract k-simplices of $C_t(X)$ are given by unordered $(k+1)$-tuples of sample points whose pairwise distances do not exceed the parameter t.

Given a collection of data points, the resulting Vietoris-Rips complex, and its homology, is highly dependent on the choice of parameter t. To reconcile this ambiguity, persistence exploits that if $t_1 < t_2$ then C_{t_1} is a sub simplicial complex of C_{t_2}. In other words, as t grows

so do the Vietoris-Rips complexes, giving an inclusion from earlier complexes to those which appear later. The idea then is to not only consider the homology for a single specified choice of parameter, but rather track topological features through a range of parameters [39]. Those which persist over a large range of values are considered signals of underlying topology, while the short lived features are taken to be noise inherent in approximating a topological space with a finite sample [35].

For clarity, consider 4 points in the plane with distance matrix

$$\begin{bmatrix} 0 & t_2 & t_5 & t_3 \\ t_2 & 0 & t_1 & t_6 \\ t_5 & t_1 & 0 & t_4 \\ t_3 & t_6 & t_4 & 0 \end{bmatrix}.$$

We label the points a, b, c and d and build the sequence of Vietoris-Rips simplicial complexes up to $\mathbf{C}_{t_5}$. Table 8.1 shows the Betti information (where β_i is the dimension of the i^{th} homology vector space) for the example illustrated in Figure 8.1 over the range of parameter values $t \geq 0$.[1]

TABLE 8.1. Persistent homology data

Filtration Times (t)	β_0	β_1
$0 \leq t < t_1$	4	0
$t_1 \leq t < t_2$	3	0
$t_2 \leq t < t_3$	2	0
$t_3 \leq t < t_4$	1	0
$t_4 \leq t < t_5$	1	1
$t_5 \leq t$	1	0

Even in this simple example, the amount of information created by the persistent homology computation is non-trivial. Furthermore, an effective rendering of the complexes in

[1]For finite data there will only be finitely many parameter values where the simplicial complex changes.

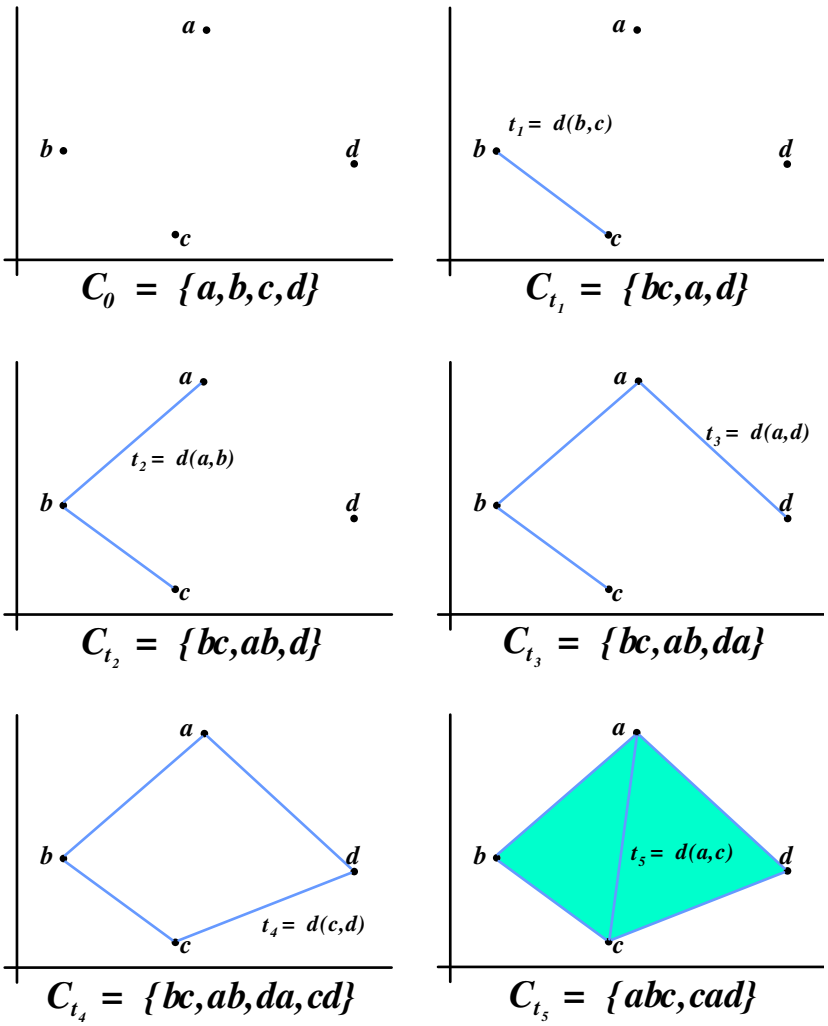


FIGURE 8.1. A sequence of Vietoris-Rips simplicial complexes shown geometrically and abstractly along with their maximal faces.

Figure 8.1 is only possible because there are very few points in the example. In the 4-point example, at time t_6 the simplicial complex $\mathbf{C}_{t_6}$ becomes three-dimensional. As the vertex set or the dimension of the ambient space grows, visualizing the sequence of complexes is not practical.

The *barcode* is a visual method for presenting some of the homological information in a sequence of chain maps. In particular, it displays the structure of the invariant factors of

the i^{th} persistence module. Figure 8.2 is the barcode corresponding to the example of the four points in the plane described in Figure 8.1.

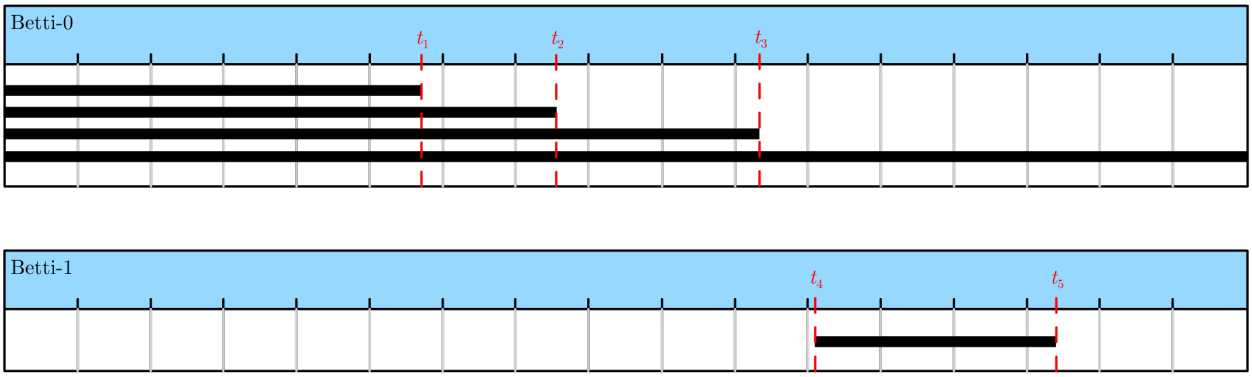


FIGURE 8.2. Barcodes corresponding to Figure 8.1

The computational requirements of the persistence computation is related to the sample size. It is often the case that computing the persistent homology using the Rips filtration is impractical. There is an alternative construction, introduced by Carlsson and de Silva, called the *witness complex* [29, 41]. Starting with a large sample set X, one picks a distinguished subset $L \subset X$ of *landmark points*. The witness complex is a family of simplicial complexes built on L using information from the entire set X.

To build the witness complex, first use the landmark set to assign to each point $x \in X$ the numbers $m_k(x)$ corresponding to the distance from x to its $(k+1)$-th nearest landmark point. For each integer k $(0 < k < |X|)$ and vertices $\{l_{j_i} | 0 \leq i \leq k\} \subset L$, include the k-simplex $[l_{j_0} l_{j_1} ... l_{j_k}]$ in the complex (at time t) if there exists a point $x \in X$ such that $\max\{d(l_{j_i}, x) | 0 \leq i \leq k\} \leq t + m_k(x)$, and if all of its faces are in the complex [2].

The output of the witness filtration is sensitive to the choice of landmark set. One technique for choosing a landmark set, called sequential maxmin, is implemented in the freely distributed persistent homology software package JPlex [**?**]. The procedure for using sequential maxmin is to first pick a point $l_0 \in X$ then inductively choose the i-th landmark

point from X by choosing the point furthest from the set of $(i-1)$ points already chosen. In practice, this seems to produce a stronger topological signal than choosing L randomly, so it is the method we will utilize.

8.2.2. Algebraic Varieties and Numerical Algebraic Geometry. A motivating problem for this paper is the computation of the Betti numbers of a complex projective algebraic variety from numerically obtained sample points. The method we use to obtain sample points derive from several algorithms in numerical algebraic geometry.

The term *numerical algebraic geometry* is often used to describe a wide ranging set of numerical methods to extract algebraic and geometric information from polynomial systems. The field includes a diverse collection of algorithms (both numeric and numeric-symbolic). The class of numerical algorithms that we use are rooted in homotopy continuation. The idea of homotopy continuation is to link a pair of polynomial systems through a deformation and to relate features of the two systems through this deformation. For example, one can track known, isolated, complex solutions of one polynomial system to unknown, complex solutions of a second polynomial system through a deformation of system parameters.

Let Z be the complex algebraic variety associated to an ideal in $\mathbb{C}[z_1, \ldots, z_N]$. With numerical homotopy continuation methods combined with monodromy breakup, it is practical to produce sets of numerical data points which numerically lie on each of the irreducible components of Z [73, 74].

There are several important features of the methods of numerical algebraic geometry that are worth highlighting. The first feature is the ability to refine sample points to arbitrarily high precision via Newton's method. A second feature is the ability to produce an arbitrary number of sample points on any given component. A third feature is the parallelizability of

these numerical methods. For instance, 10,000 processors could be used in parallel to track 10,000 paths and could be used in parallel to refine the accuracy of each sample point to arbitrarily high precision. The basic algorithms of numerical algebraic geometry (including monodromy breakup) are implemented in the freely available software package, Bertini [8].

It is important to note that sampling is computationally inexpensive, so obtaining large sample sets does not pose a significant challenge. However, it is not clear that this sampling technique will provide points that are well distributed for the purpose of persistent homology computations.

8.3. Main Idea

8.3.1. Theory. By its very nature, persistent homology characterizes intrinsic topological features which should be relatively insensitive to the metric used to build a pairwise distance matrix. However, experiments show that the signal *strength* is impacted by the choice of metric. In our experience, even if the topological features remain the same, the ability to correctly interpret information from a barcode depends on the strength of the signal. We will consider the barcode signal strength of mappings of an algebraic variety into various Grassmannians.

The Grassmannian $Gr(n, k)$ is a manifold parametrizing all k dimensional subspaces of a fixed n dimensional vector space. The Grassmann manifold $Gr(n + 1, 1)$ is the projective space $\mathbb{P}^n$, and from this vantage point, Grassmannians can be viewed as generalizations of projective spaces. These manifolds can be given a topological structure, a differential structure and even the structure of a projective variety (e.g. via the Plucker embedding).

Points in an n-dimensional projective space correspond to 1-dimensional subspaces of a fixed $(n + 1)$-dimensional vector space. A natural notion of distance is given by the smallest

angle between the subspaces. We would like to define the distance between points on other Grassmannians by extending this definition. As a starting point, it can be shown that any unitarily invariant metric on a Grassmannian can be written in terms of the principal angles between the corresponding subspaces. The principal angles between a pair of subspaces A, B in $\mathbb{C}^n$ can be determined as follows. First, determine matrices M and N whose columns form orthonormal bases for A and B. Next, determine the singular value decomposition $M^*N = U\Sigma V^*$. The singular values of M^*N are the diagonal entries of Σ. These singular values are the cosines of the principal angles between A and B (see [14]). If A and B are k-dimensional, then there will be principal angles $\Theta(A, B) = (\theta_1, \theta_2, \ldots, \theta_k)$ with $0 \leq \theta_1 \leq \theta_2 \leq \cdots \leq \theta_k \leq \pi/2$. There are many common metrics computed as functions of the principal angles [6]. For instance, the Fubini-Study metric induced by the Plucker embedding is $d_{FS}(A, B) = \cos^{-1}\left(\prod_{i=1}^{k} \cos\theta_i\right)$. We have found that the Fubini-Study metric does not, in general, yield a strong signal, and instead, we restrict our attention to the geodesic distance

$$d(A, B) = \sqrt{\theta_1^2 + \ldots + \theta_k^2}.$$

Since we wish to compare the effect of considering a sample in various Grassmannian embeddings, it remains to define what we mean by relative topological signal strength. Imagine we know that our sample was taken from a topological space whose i^{th} Betti number is b_i. Assuming that the b_i longest segments in the barcode represent these topological features, we will measure signal strength as the ratio of the sum of the lengths of the b_i longest segments to the sum of the total length of all the segments in the i^{th} Betti barcode, including noise. Note that noise consisting of many segments of total length m and noise consisting of a single segment of length m cannot be distinguished by this statistic. To cope

with this limitation we also consider the ratio of the length of the b_i^{th} longest segment to the $(b_i + 1)^{th}$ longest segment in the barcode.

8.3.2. Embeddings into the Grassmannian. Consider a complex projective curve, $C \subset \mathbb{P}^2$, defined by the zero locus of a homogenous polynomial $F(x, y, z)$. When we think of the zero set as a projective variety, then each point, $[x : y : z]$ on C, corresponds to a 1-dimensional subspace of $\mathbb{C}^3$ (note that the homogeneity of the equation leads to the conclusion that if (x, y, z) is a solution then so is (cx, cy, cz) for any $c \in \mathbb{C}$). Thus, points on a projective variety correspond to one-dimensional subspaces of $\mathbb{C}^3$ constrained to lie on the vanishing locus of a homogeneous polynomial. From this point of view, C is a sub-object of $\mathbb{P}^2 = Gr(3, 1)$. We can sample random points on C with several different methods. If we wish to build a distance matrix from these points, then we should consider the distance between a pair of points as the principal angle between the one dimensional spaces to which they correspond.

Consider the matrix

$$E(x, y, z) := \begin{bmatrix} 0 & z & -y \\ -z & 0 & x \\ y & -x & 0 \end{bmatrix},$$

and observe that for any point $(x, y, z) \neq (0, 0, 0)$, the rank of $E(x, y, z)$ is 2. This can be seen by observing that the determinant of $E(x, y, z)$ is identically zero and that the locus of conditions such that all 2×2 minors of $E(x, y, z)$ are zero force $x = y = z = 0$. For each

value of (x, y, z), we consider the row space of $E(x, y, z)$. Note also that

$$\begin{bmatrix} 0 & z & -y \\ -z & 0 & x \\ y & -x & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

As a consequence, the row space of $E(x, y, z)$ is the same as the row space of $E(cx, cy, cz)$, and $E(x, y, z)$ can be viewed as a rule for attaching a smoothly varying 2 dimensional subspace to each point of $\mathbb{P}^2$. In other words, $E(x, y, z)$ determines a rank two vector bundle on $\mathbb{P}^2$. For each one dimensional subspace of $\mathbb{C}^3$, we can determine a 2-dimensional subspace of $\mathbb{C}^3$ by mapping it to the row space of $E([x : y : z])$. If $\Phi_0 : \mathbb{P}^2 \to Gr(3, 2)$ denotes the image of this map, then by restriction this gives a map $\phi_0 : C \to Gr(3, 2)$.

For each integer $k > 0$, consider the set of monomials in x, y, z of degree k. We construct new matrices, $E_k(x, y, z)$, by concatenating matrices of the form $m_i \cdot E(x, y, z)$ for each degree k monomial m_i. For example, $E_1(x, y, z)$ is the matrix

$$\begin{bmatrix} 0 & xz & -xy & 0 & yz & -y^2 & 0 & z^2 & -zy \\ -xz & 0 & x^2 & -yz & 0 & yx & -z^2 & 0 & zx \\ xy & -x^2 & 0 & y^2 & -yx & 0 & zy & -zx & 0 \end{bmatrix}.$$

For each k, $E_k(x, y, z)$ has constant rank 2 on $\mathbb{P}^2$ and can be used to define a map $\phi_k : C \to Gr(N_k, 2)$ (where N_k is the number of columns of $E_k(x, y, z)$). Geometrically, the columns of $E_k(x, y, z)$ corresponds to a "spanning set for the space of sections of the twisted tangent bundle, $\mathbf{T}_{\mathbb{P}^2}(k-1)$". In this way, we can consider images of C, in increasingly large Grassmannians via the maps $\phi_0, \phi_1, \phi_2, \ldots$. It can be shown that for each k, Φ_k embeds $\mathbb{P}^2$ into $Gr(N_k, 2)$ and that ϕ_k embeds C into $Gr(N_k, 2)$.

8.3.3. EXAMPLE. Consider the complex projective elliptic curve, $C \subset \mathbb{P}^2$ defined by the equation

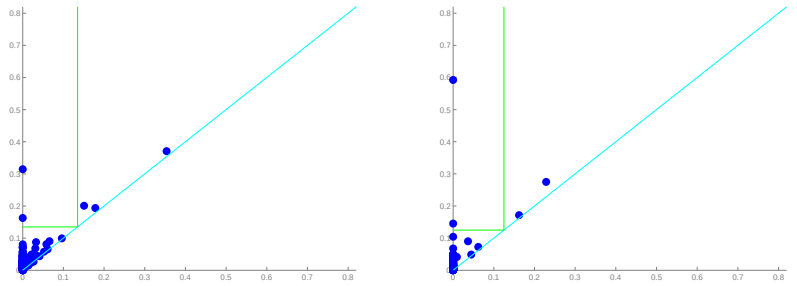
$$x^2y + y^2z + z^2x = 0. \tag{8.1}$$

Topologically, C is a torus whose Betti numbers are $\beta_0 = 1, \beta_1 = 2, \beta_2 = 1$.

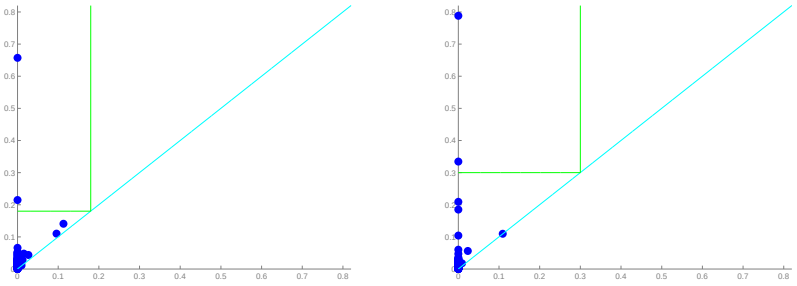
Using Bertini, we sampled 10,000 points satisfying Equation 8.1. We mapped each point to $Gr(N_k, 2)$ using ϕ_k, for $k = 1, \ldots, 10$. From these 10,000 points we fixed 100 landmark sets, L_i (of size 200) using the sequential maxmin algorithm with random initial points $l_{0,i}$ for $i = 1, \ldots, 100$. For each embedding $\phi_k(C)$ and for each of the fixed landmark sets, we compute the persistent homology barcodes for the zeroth and first Betti numbers using the witness complex construction.

Using the geodesic distance to measure distances between points, Figure 8.3 shows prototypical Betti-1 barcodes for the images of the 10,000 points in $Gr(N_k, 2)$. In the figure, each segment in the barcode is plotted as a point in the (x, y)-plane with the x-coordinate corresponding to the starting parameter and the y-coordinate corresponding to the ending parameter. Short segments (i.e. topological noise) appear near the $y = x$ line. Notice that as we move the elliptic curve to Grassmannians of higher degree, the two longest segments in the barcode grow in length while the number and lengths of the other segments decrease.

In Figure 8.4, we plot the relative signal strength of the Betti-1 barcodes, as measured by the ratio of the sum of the length of the two longest segments to the total sum of lengths of all segments, averaged over all landmark sets for each embedding. We observe an increase from approximately 10% to 55% of the total length of the barcodes being accounted for



(A) $k = 1$ (B) $k = 2$



(C) $k = 3$ (D) $k = 4$

FIGURE 8.3. Betti-1 barcodes for each of the four specified embeddings.

in the longest two segments. We also observe that the improvement of the relative signal strength levels-off after $k = 5$.

Figure 8.5 compares the second longest segment in the Betti-1 barcode (corresponding to a topological circle) to the third longest segment (representing topological noise). We notice a sharp increase in this measure of signal strength followed by a similarly steep decrease. Together with the content of Figure 8.4 this indicates that after $k = 5$, the relative length of the longest Betti-1 segment remains somewhat unchanged while the disparity in the lengths of the second and third longest segments is diminished.

It is worth noting that there is a diminished signal strength from the projective variety to the embeddings in the Grassmannian. Our aim, however, is to compare topological signal strength improvement across successive Grassmannian embeddings.

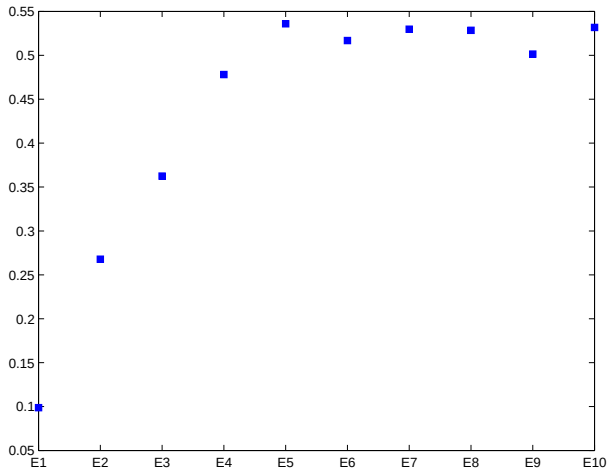


FIGURE 8.4. Average ratio of the sum of the longest two barcode lengths to the sum of the lengths of all barcodes for $k = 1, \ldots, 10$.

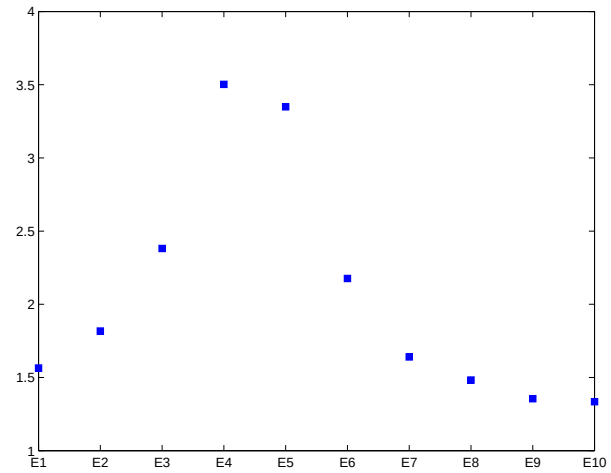


FIGURE 8.5. Average ratio of the second longest barcode to the third longest barcode for $k = 1, \ldots, 10$.

8.4. Conclusion

Using the techniques of numerical algebraic geometry, we can sample arbitrarily many points, to an arbitrary degree of accuracy, on any prescribed component of an algebraic set. Using twists of the tangent bundle to projective space, we can map these points to a sequence of Grassmann manifolds of increasing dimension. With techniques of computational

homology, we can build the persistence module and decompose the module into its invariant factors. A visual plot of the starting and ending points of the invariant factors aids in the understanding of the underlying variety as a topological space. Higher embeddings of the data seem to strengthen the topological signal.

For further research, we intend to develop improved sampling techniques for algebraic varieties. We will also conduct experiments to determine if alternate vector bundles or alternate metrics on the Grassmannian can be used to strengthen topological signals.

CHAPTER 9

Conclusion

In this dissertation, we have discussed several geometric, topological, and optimization techniques to analyze a variety of large data sets from the color data of an image to gene expression data.

After some background algorithms and metrics were discussed in Chapter 2, we discussed an application to landscape ecology in Chapter 3. A friendly graphical user interface called BLOSSM was developed and presented in this chapter. The function of BLOSSM is to aid ecologists (or any other scientists with images to analyze) in determining the ground cover of a landscape using the well-known clustering algorithm LBG. This GUI allows the user to determine various ecological variables from this analysis including: color abundance, number of flower clusters for each color, average size of flower clusters for each color, abundance of 'morphospecies' as characterized by color and cluster size, richness, and species diversity as measured by Shannon entropy. It was shown that the choice of color space and choice of metric can have a large impact on the reconstructions. We analyzed four different choices of color space RGB, Quantized RGB, Named Color, and CIELab as well as five different metrics ℓ_1, ℓ_2, ℓ_∞, Mahalanobis distance, and spectral angle. Depending on the application an appropriate choice of color space and metric may be determined to analyze an image.

In Chapter 4, we discussed a novel clustering algorithm, Locally Linear Embedding Clustering. The ability to characterize the color content of natural imagery is an important application of image processing. The pixel by pixel coloring of images may be viewed naturally

as points in color space, and the inherent structure and distribution of these points affords a quantization, through clustering, of the color information in the image. We presented a novel topologically driven clustering algorithm that permits segmentation of the color features in a digital image. The algorithm blends Locally Linear Embedding (LLE) and vector quantization by mapping color information to a lower dimensional space, identifying distinct color regions, and classifying pixels together based on both a proximity measure and color content. It is observed that these techniques permit a significant reduction in color resolution while maintaining the visually important features of images.

The LLE algorithm has proven to be a useful technique for revealing geometric structures in high dimensional data. The basic algorithm reconstructs each data point by a weighted average of its nearest neighbors, and the geometry obtained by these weights captures the lower-dimensional embedding. We observe that the embedding reconstruction is highly dependent on the parameter choice of the number of nearest neighbors, i.e., the geometric structure is not robust to parameter selection. In Chapter 5, we discussed modifications to the LLE optimization problem that address this shortcoming of standard LLE. This is accomplished by altering the objective function by introducing a data-weighted ℓ_1 regularization term. We observed that this new formulation has proven effective at automatically determining nearest neighbors using sparsity of numerical results. In Chapter 6, we discussed one algorithm, the Primal Dual Interior Point Algorithm, to solve this modified quadratic program. A variety of formulations were presented and complexity analysis was discussed.

The convex hull of a set of points in Euclidean space can help elucidate overall properties of the data and can help identify specific data points of high interest. In Chapter 7, we proposed a quadratic programming (QP) problem for the purpose of stratifying points in

a high dimensional data cloud based on proximity to the convex hull. A QP problem is solved for each data point to determine an associated weight vector. We showed that the weight vector of a given data point encodes geometric information concerning the point's relationship to the faces of the convex hull. For instance, we observe that the ℓ_2-norm of the weight vector is a measure of the distance of the associated point from the boundary. A necessary (but not sufficient) condition for a point to be a vertex is that it have weight components with negative values. However, by adjusting parameters in the QP, we can reduce the set of data points with negative weight components until only the vertices of the convex hull remain. It is noted that the weight vector computation can be carried out in parallel and that the overall computational complexity of the algorithm grows linearly with dimension. As a consequence, meaningful computations can be completed on reasonably large, high dimensional data sets.

Finally, in Chapter 8, we discussed the relatively new tool from computational topology, persistent homology (PH). PH is a multiscale algorithm that looks for structure in data sets at different scales to observe which features persist in each scale. In many settings noisy, discrete sets of points are realized as a sample of a topological space. Gaining insight into the topological structure of such a sampling is a fundamental consideration of topological data analysis. As the size of the sample grows, the quality of the topological signals produced by PH improves. However, the computational complexity of PH grows dramatically. Thus, we consider an embedding technique to improve topological signal strength of points sampled from a projective algebraic variety without increasing the sample size. By embedding into successive Grassmanians, we observe that the topological signal strength improves in PH computations.

All of the algorithms, results, and techniques in this dissertation have a main focus of understanding the structure of data. By quantizing, reducing the dimensionality, embedding into a new space, or determining extremal points that carry the most information, we can gain knowledge as to the nature of the data.

BIBLIOGRAPHY

[1] Locally linear embedding, swiss roll source code. `http://www.cs.nyu.edu/~roweis/lle/code/swissroll.m`. Accessed: 2013-04-24.

[2] Henry Adams. Jplex with matlab tutorial, 2011.

[3] Andrés Álvarez-Meza, Juliana Valencia-Aguirre, Genaro Daza-Santacoloma, and Germán Castellanos-Domínguez. Global and local choice of the number of nearest neighbors in locally linear embedding. *Pattern Recognition Letters*, 32(16):2171–2177, 2011.

[4] C.M. Bachmann, T.L. Ainsworth, and R.A. Fusina. Exploiting manifold geometry in hyperspectral imagery. *Geoscience and Remote Sensing, IEEE Transactions on*, 43(3):441 – 454, march 2005.

[5] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE*, 22(4):469–483, 1996.

[6] Alexander Barg and Dmitry Yu. Nogin. Bounds on packings of spheres in the grassmann manifold. *IEEE TRANS. INFO. THEORY*, 48:2450–2454, 2002.

[7] Sugato Basu, Ian Davidson, and Kiri Wagstaff. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. Chapman & Hall/CRC, 1 edition, 2008.

[8] Daniel J. Bates, Jonathan D. Hauenstein, Andrew J. Sommese, and Charles W. Wampler II. Bertini: Software for numberical algebraic geometry.

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

[10] Robert Benavente, Maria Vanrell, and Ramon Baldrich. Parametric fuzzy sets for automatic color naming. *Journal of the Optical Society of America A*, 25(10):2582–2593, 2008.

[11] Radu Berinde and Piotr Indyk. Sparse recovery using sparse random matrices. *preprint*, 2008.

[12] Brent Berlin and Paul Kay. *Basic color terms: Their universality and evolution*. Univ of California Press, 1991.

[13] Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997.

[14] Åke Björck and Gene H. Golub. Numerical methods for computing angles between linear subspaces. *Math. Comp.*, 27:579–594, 1973.

[15] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[16] Paul S Bradley and Olvi L Mangasarian. Feature selection via concave minimization and support vector machines. In *Machine Learning Proceedings of the Fifteenth International Conference (ICML98)*, pages 82–90, 1998.

[17] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

[18] Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489–509, 2006.

[19] Emmanuel J Candes and Terence Tao. Decoding by linear programming. *Information Theory, IEEE Transactions on*, 51(12):4203–4215, 2005.

[20] Gunnar Carlsson. Topology and data. *Bull. Amer. Math. Soc. (N.S.)*, 46(2):255–308, 2009.

[21] Jin Chen, Miaogen Shen, Xiaolin Zhu, and Yanhong Tang. Indicator of flower status derived from¡ i¿ in situ¡/i¿ hyperspectral measurement in an alpine meadow on the tibetan plateau. *Ecological Indicators*, 9(4):818–823, 2009.

[22] Yangchi Chen, M.M. Crawford, and J. Ghosh. Applying nonlinear manifold learning to hyperspectral data for land cover classification. In *Geoscience and Remote Sensing Symposium, 2005. IGARSS '05. Proceedings. 2005 IEEE International*, volume 6, pages 4311 – 4314, july 2005.

[23] H. D. Cheng, X. H. Jiang, Y. Sun, and Jing Li Wang. Color image segmentation: Advances and prospects. *Pattern Recognition*, 34:2259–2281, 2001.

[24] Shyi-Chyi Cheng and Chen kuei Yang B. A fast and novel technique for color quantization using reduction of color space dimensionality, 2001.

[25] Fan R. K. Chung. *Spectral graph theory*, volume 92 of *CBMS Regional Conference Series in Mathematics*. Published for the Conference Board of the Mathematical Sciences, Washington, DC, 1997.

[26] Bogdan R Cosofret, Daisei Konno, Aram Faghfouri, Harry S Kindle, Christopher M Gittins, Michael L Finson, Tracy E Janov, Mark J Levreault, Rex K Miyashiro, and William J Marinelli. Imaging sensor constellation for tomographic chemical cloud mapping. *Applied optics*, 48(10):1837–1852, 2009.

[27] Trevor F. Cox and Michael A. A. Cox. *Multidimensional scaling*, volume 59 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, London, 1994. With 1 IBM-PC floppy disk (3.5 inch, HD).

[28] Genaro Daza-Santacoloma, Carlos D Acosta-Medina, and Germán Castellanos-Domínguez. Regularization parameter choice in locally linear embedding. *Neurocomputing*, 73(10):1595–1605, 2010.

[29] V. de Silva and G. Carlsson. Topological estimation using witness complexes. *IEEE Symposium on Point-based Graphic*, pages 157–166, 2004.

[30] Y. Deng and B.S. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(8):800 –810, aug 2001.

[31] David L. Donoho and Carrie Grimes. Hessian eigenmaps: New locally linear embedding techniques for high-dimensional data, 2003.

[32] David Leigh Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306, 2006.

[33] Herbert Edelsbrunner and John Harer. Persistent homology—a survey. In *Surveys on discrete and computational geometry*, volume 453 of *Contemp. Math.*, pages 257–282. Amer. Math. Soc., Providence, RI, 2008.

[34] Herbert Edelsbrunner and John L. Harer. *Computational topology*. American Mathematical Society, Providence, RI, 2010. An introduction.

[35] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete Comput. Geom.*, 28(4):511–533, 2002. Discrete and computational geometry and graph drawing (Columbia, SC, 2001).

[36] Anders Eriksson and Anton Van Den Hengel. Efficient computation of robust low-rank matrix approximations in the presence of missing data using the l¡ inf¿ 1¡/inf¿ norm. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages

771–778. IEEE, 2010.

[37] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[38] Jessica Forrest, David W Inouye, and James D Thomson. Flowering phenology in subalpine meadows: does climate variation influence community co-flowering patterns? *Ecology*, 91(2):431–440, 2010.

[39] Robert Ghrist. Barcodes: The persistent topology of data. *Bulletin of the American Mathematical Society*, 45:61–75, 2008.

[40] Tom Goldstein and Stanley Osher. The split bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.

[41] Leonidas J. Guibas and Steve Y. Oudot. Reconstruction using witness complexes. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1076–1085, New York, 2007. ACM.

[42] Tian Han and D.G. Goodenough. Nonlinear feature extraction of hyperspectral data based on locally linear embedding (lle). In *Geoscience and Remote Sensing Symposium, 2005. IGARSS '05. Proceedings. 2005 IEEE International*, volume 2, pages 1237 – 1240, july 2005.

[43] John A. Hartigan. *Clustering algorithms*. John Wiley & Sons, New York-London-Sydney, 1975. Wiley Series in Probability and Mathematical Statistics.

[44] Paul S. Heckbert and Paul S. Heckbert. Color image quantization for frame buffer display. *Computer Graphics*, 16:297–307, 1982.

[45] M. O. Hill. Diversity and evenness: A unifying notation and its consequences. *Ecology*, 54(2):pp. 427–432, 1973.

[46] Robert V. Hogg and Elliot A. Tanis. *Probability and Statistical Inference*. Prentice Hall, Upper Saddle Rive, NJ, fifth edition, 1999.

[47] Roger A. Horn and Charles R. Johnson. *Matrix analysis*. Cambridge University Press, Cambridge, 1990. Corrected reprint of the 1985 original.

[48] Jinggang Huang and David Mumford. Statistics of natural images and models. pages 541–547.

[49] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[50] I. T. Jolliffe. *Principal component analysis*. Springer Series in Statistics. Springer-Verlag, New York, second edition, 2002.

[51] Tomasz Kaczynski, Konstantin Mischaikow, and Marian Mrozek. *Computational homology*, volume 157 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 2004.

[52] Rasa Karbauskait, Olga Kurasova, and Gintautas Dzemyda. Selection of the number of neighbours of each data point for the locally linear embedding algorithm. *ITC*.

[53] Rasa Karbauskaitė, Gintautas Dzemyda, and Virginijus Marcinkevičius. Dependence of locally linear embedding on the regularization parameter. *Top*, 18(2):354–376, 2010.

[54] Michael Kirby. *Geometric data analysis*. Wiley-Interscience [John Wiley & Sons], New York, 2001. An empirical approach to dimensionality reduction and the study of patterns.

[55] Victor Klee. On the complexity of d-dimensional voronoi diagrams. *Archiv der Mathematik*, 34(1):75–80, 1980.

[56] Teuvo Kohonen. *Self-organizing maps*, volume 30 of *Springer Series in Information Sciences*. Springer-Verlag, Berlin, second edition, 1997.

[57] Olga Kouropteva, Oleg Okun, and Matti Pietikinen. Selection of the optimal parameter value for the locally linear embedding algorithm. In *1 st International Conference on Fuzzy Systems and*, pages 359–363, 2002.

[58] Sandra Lavorel, Karl Grigulis, Sue McIntyre, Nick SG Williams, Denys Garden, Josh Dorrough, Sandra Berman, Fabien Quétier, Aurélie Thébault, and Anne Bonis. Assessing functional diversity in the field–methodology matters! *Functional Ecology*, 22(1):134–147, 2008.

[59] Ann B. Lee, Kim Steenstrup Pedersen, and David Mumford. The nonlinear statistics of high-contrast patches in natural images. *International Journal of Computer Vision*, 54(1-3):83–103, 2003.

[60] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *Communications, IEEE Transactions on*, 28(1):84 – 95, jan 1980.

[61] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pages 849–856. MIT Press, 2001.

[62] M.T. Orchard and C.A. Bouman. Color quantization of images. *Signal Processing, IEEE Transactions on*, 39(12):2677 –2690, dec 1991.

[63] Joseph O'Rourke. *Computational Geometry in C*. Cambridge University Press, New York, NY, USA, 2nd edition, 1998.

[64] Stanley Osher, Martin Burger, Donald Goldfarb, Jinjun Xu, and Wotao Yin. An iterative regularization method for total variation-based image restoration. *Multiscale*

Modeling & Simulation, 4(2):460–489, 2005.

[65] N. Papamarkos, A.E. Atsalakis, and C.P. Strouthopoulos. Adaptive color reduction. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 32(1):44 –56, feb 2002.

[66] Marzia Polito and Pietro Perona. Grouping and dimensionality reduction by locally linear embedding. In *Advances in Neural Information Processing Systems 14*, pages 1255–1262. MIT Press, 2001.

[67] Andrew D Richardson, Bobby H Braswell, David Y Hollinger, Julian P Jenkins, and Scott V Ollinger. Near-surface remote sensing of spatial and temporal variation in canopy phenology. *Ecological Applications*, 19(6):1417–1428, 2009.

[68] Andrew D Richardson, Julian P Jenkins, Bobby H Braswell, David Y Hollinger, Scott V Ollinger, and Marie-Louise Smith. Use of digital webcam images to track spring green-up in a deciduous broadleaf forest. *Oecologia*, 152(2):323–334, 2007.

[69] Nicholas Rohrbacker. Sparse multivariate analyses via l-1-regularized optimization problems solved with bregman iterative techniques. Documentation for general BibTeX users, October 2012.

[70] Nicolas Le Roux, Pascal Lamblin, Yoshua Bengio, Marc Joliveau, and Balzs Kgl. Learning the 2-d topology of images, 2008.

[71] Lawrence K. Saul, Sam T. Roweis, and Yoram Singer. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

[72] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 1997.

[73] Andrew J. Sommese, Jan Verschelde, and Charles W. Wampler. Numerical decomposition of the solution sets of polynomial systems into irreducible components. *SIAM J. Numer. Anal.*, 38(6):2022–2046, 2001.

[74] Andrew J. Sommese and Charles W. Wampler, II. *The numerical solution of systems of polynomials*. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2005. Arising in engineering and science.

[75] J. B. Tenenbaum, V. Silva, and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, 2000.

[76] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[77] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics, 1997.

[78] Joost van de Weijer, Cordelia Schmid, Jakob Verbeek, and Diane Larlus. Learning color names for real-world applications. *Trans. Img. Proc.*, 18(7):1512–1523, July 2009.

[79] Robert J. Vanderbei. Linear programming: Foundations and extensions, 1996.

[80] Luiz Velho, Jonas Gomes, Marcos Vinicius, and Rayol Sobreiro. Color image quantization by pairwise clustering. In *in Proc. Tenth Brazilian Symp. Comput. Graph. Image Process*, pages 203–210. IEEE Computer Society, 1997.

[81] R. Vidal, Yi Ma, and S. Sastry. Generalized principal component analysis (gpca). *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(12):1945 –1959, dec. 2005.

[82] K.Q. Weinberger and L.K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004.*

Proceedings of the 2004 IEEE Computer Society Conference on, volume 2, pages II–988 – II–995 Vol.2, june-2 july 2004.

[83] S. Westland and C. Ripamonti. *Computational colour science using MATLAB*. J. Wiley, 2004.

[84] S. Westland, C. Ripamonti, and V. Cheung. *Computational colour science using MATLAB, second edition*. J. Wiley, 2012.

[85] Aimee K Zaas, Minhua Chen, Jay Varkey, Timothy Veldman, Alfred O Hero III, Joseph Lucas, Yongsheng Huang, Ronald Turner, Anthony Gilbert, Robert Lambkin-Williams, et al. Gene expression signatures diagnose influenza and other symptomatic respiratory viral infections in humans. *Cell host & microbe*, 6(3):207–217, 2009.

[86] Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete Comput. Geom*, 33:249–274.

[87] Afra J. Zomorodian, M. J. Ablowitz, S. H. Davis, E. J. Hinch, A. Iserles, J. Ockendon, and P. J. Olver. *Topology for Computing (Cambridge Monographs on Applied and Computational Mathematics)*. Cambridge University Press, New York, NY, USA, 2005.

APPENDIX A

Further Analysis of Images Using BLOSSM

In Section 3.6, we analyzed a sample landscape image using the friendly GUI BLOSSM varying the choice of metric as well as the choice of color space used in the LBG clustering algorithm. Here, we consider an additional two images with properties distinct from the landscape image, such as coloring and objects present in the image. The first is a 166×250 dimensional natural image of 41500 pixels from the freely available PASCAL Object Recognition Database [37]. This image was included as the colors are much different than in the others selected. The second is a 166×200 dimensional (33200 pixels) image of a still-life canvas painted by the author. It was included both for its variety of colors and objects and that it is of an artificial scene.

Both of these images are displayed in Figures A.1 and A.2, respectively. We have initialized the LBG algorithm by selecting starting centers as pixels from the images that we feel best identify the distinct colors within the image. These starting centers are displayed on the right of each GUI. Nine starting centers were selected for the car image and fourteen were selected for the still life image.

We will now analyze the performance across all color spaces and all metrics for both images, using the LBG clustering algorithm. The Figures A.3-A.18 are grouped by color space and the metrics are varied. We have chosen to compute the entropy, denoted as E, for each reconstruction as well as the distortion error, displayed in individual figures.

FIGURE A.1. BLOSSM implemented on an image with 9 centers manually selected by the user to reflect distinct colors within image.

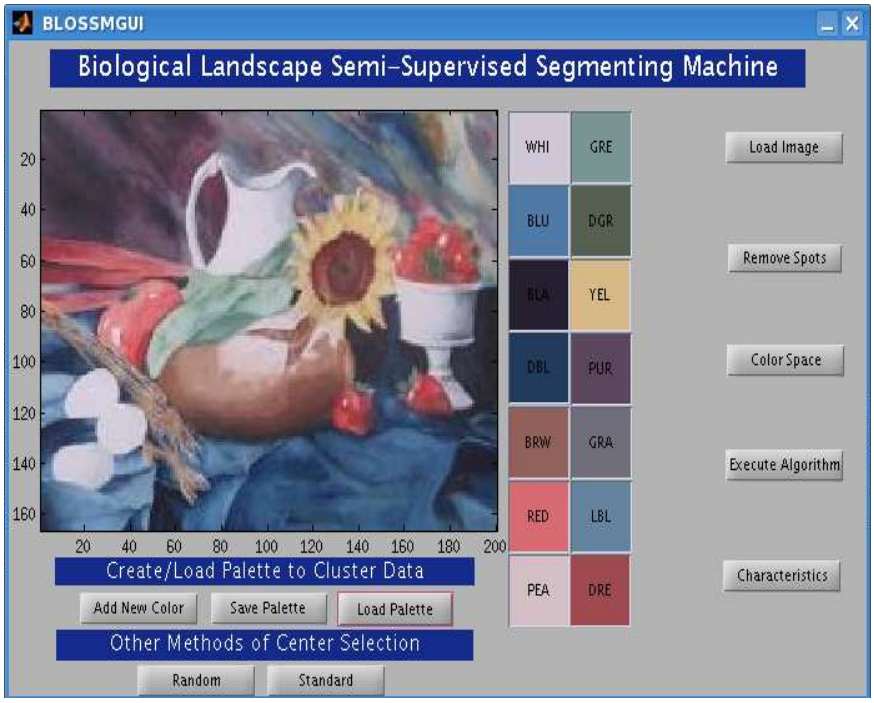


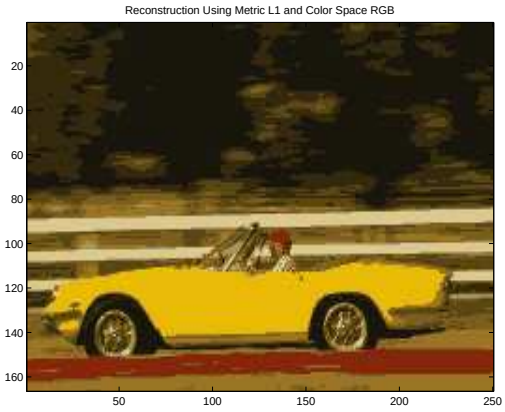
FIGURE A.2. BLOSSM implemented on an image with 14 centers manually selected by the user to reflect distinct colors within image.

Let us first consider Figure A.3. In this image, we focus on the RGB color space and vary all metrics. Notice that each of the ℓ_p norms do a nice job visually. The Mahalanobis distance reconstruction appears to have sharper contrast between objects and separates objects by regions. The spectral angle reconstruction appears quite fuzzy and blurred and visually is the worst reconstruction. Notice, however that the entropy is highest for the spectral angle. In comparing the distortion errors displayed in Figure A.4, we see that spectral angle is much higher than the other measures with the ℓ_1 and ℓ_2 norms producing the smallest error.

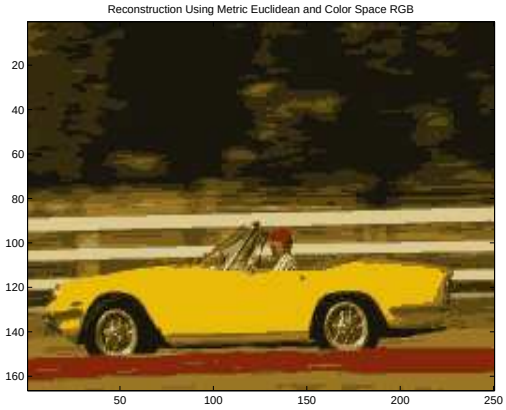
Notice that, as might be expected, the reconstructions for each metric choice using the RGB color space are comparable to the reconstructions using the Quantized RGB color space, Figure A.5. The entropy decreases from the RGB color space in most cases (except for Mahalanobis) as compared to the Quantized RGB space, and the distortion errors appear to be slightly larger, Figure A.6.

Now, we will consider reconstruction images using the Named Color space and varying metrics, Figure A.7. Visually, there appears to be much higher contrast in the background of the image using this color space and the RGB-related spaces. Also, the fence is hardly noticeable in the ℓ_p norm reconstructions. However, the color is quite distorted using Mahalanobis distance. Spectral angle in this color space seems able to reconstruct better than in the RGB spaces. Notice that the entropy is higher in this space than in the RGB spaces. The distortion error is much smaller in this color space, though, Figure A.8. This computation was done in the 11-dimensional Named Color space not the 3-dimensional space that all other color spaces reside in. The distortion error is the largest for Mahalanobis, and at the final number of iterations, it is the smallest using spectral angle.

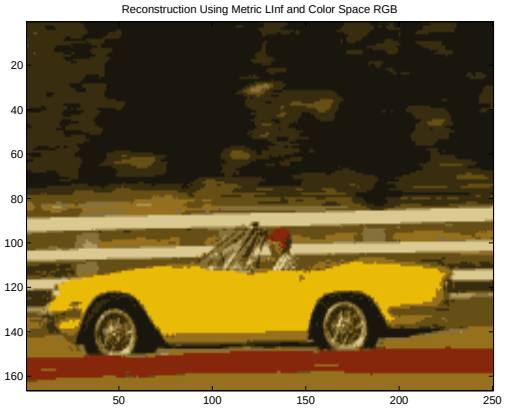
(A) Original



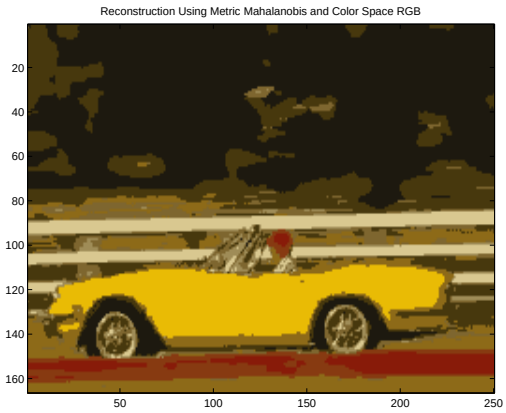
(B) ℓ_1, $E = 1.9632$



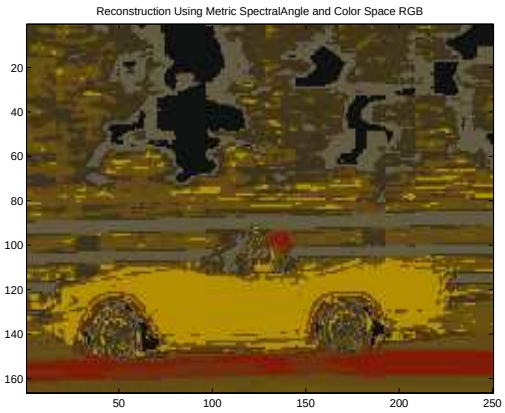
(C) ℓ_2, $E = 1.9616$



(D) ℓ_∞, $E = 1.9106$

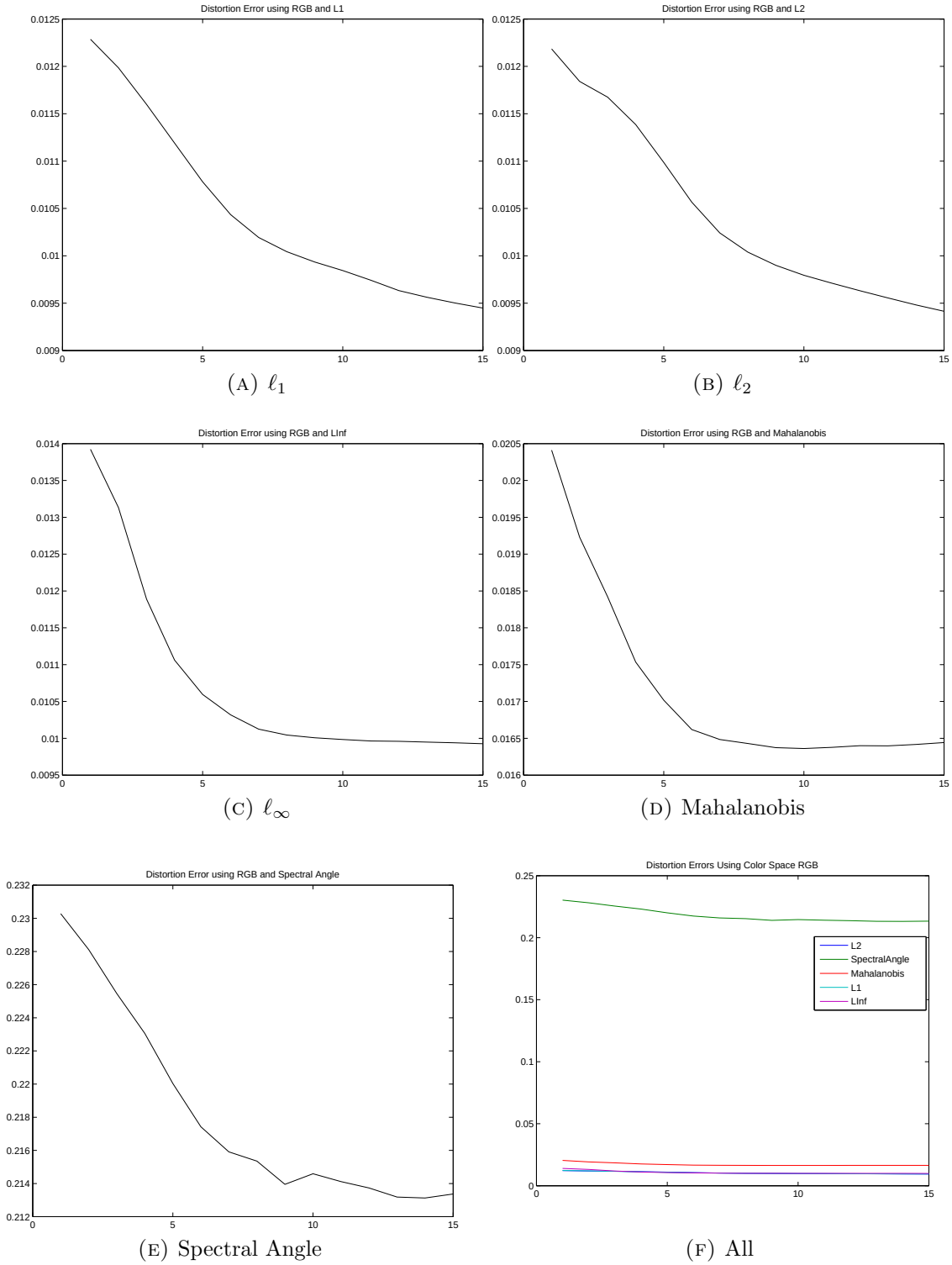


(E) Mahalanobis, $E = 1.8132$



(F) Spectral Angle, $E = 2.099$

FIGURE A.3. Reconstructions using the LBG algorithm with fixed color space RGB and varying the metric used for car image.



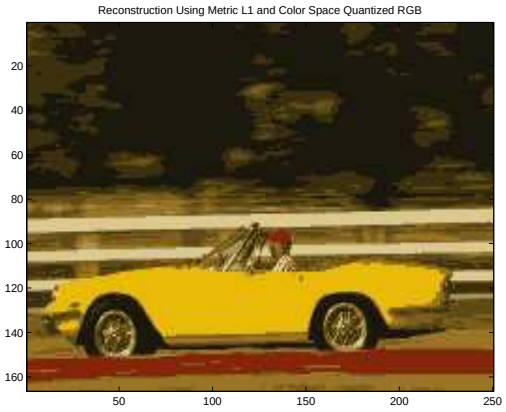
(A) ℓ_1 (B) ℓ_2

(C) ℓ_∞ (D) Mahalanobis

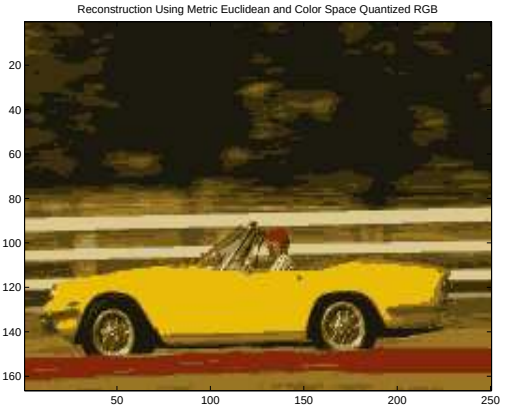
(E) Spectral Angle (F) All

FIGURE A.4. Distortion errors using the LBG algorithm with fixed color space RGB and varying the metric used for car image.

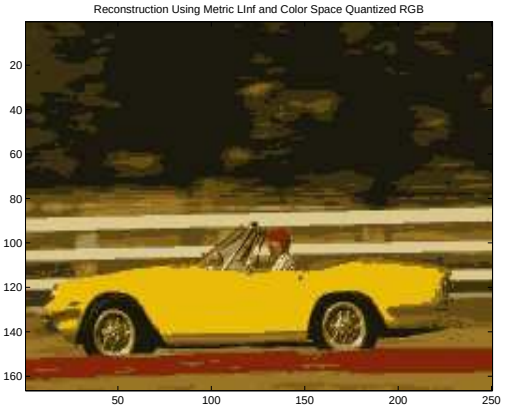
(A) Original



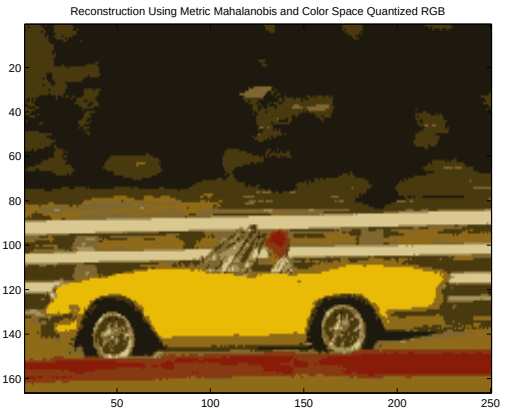
(B) ℓ_1, $E = 1.9056$



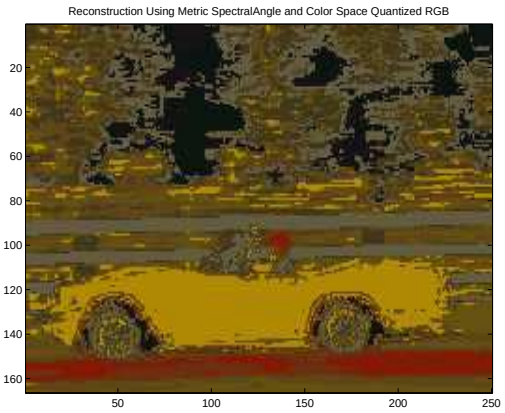
(C) ℓ_2, $E = 1.9129$



(D) ℓ_∞, $E = 1.9106$

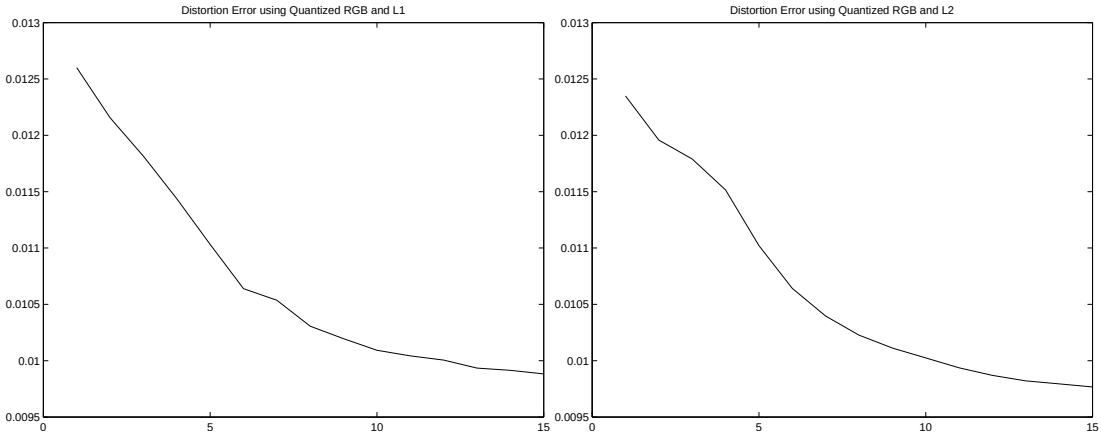


(E) Mahalanobis, $E = 1.8017$



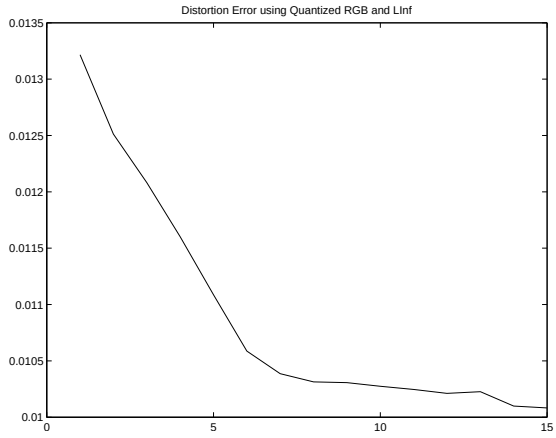
(F) Spectral Angle, $E = 2.0456$

FIGURE A.5. Reconstructions using the LBG algorithm with fixed color space Quantized RGB and varying the metric used for car image.

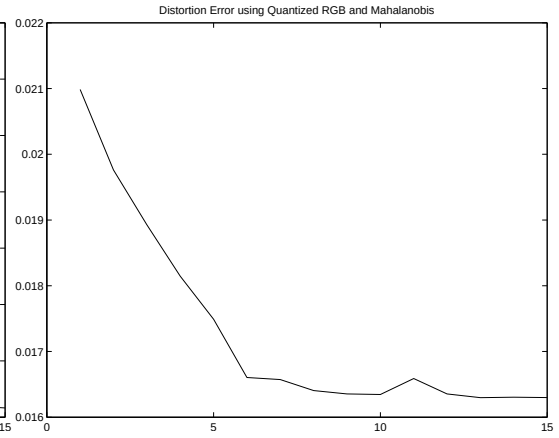


(A) ℓ_1

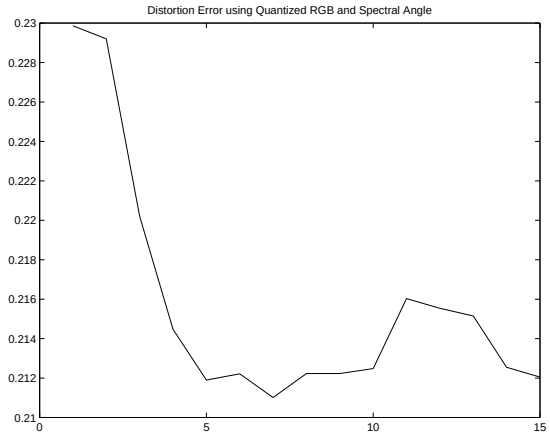
(B) ℓ_2



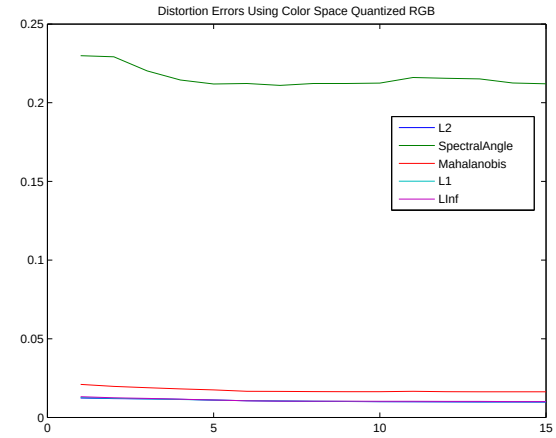
(C) ℓ_∞



(D) Mahalanobis



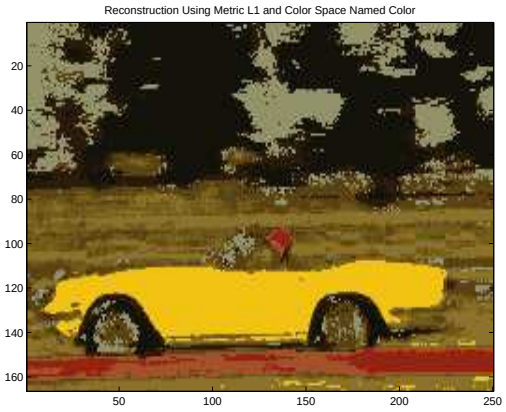
(E) Spectral Angle



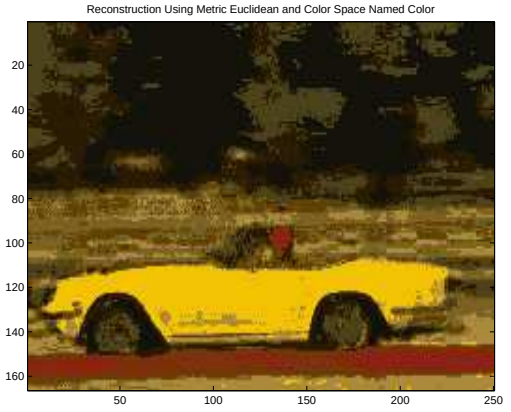
(F) All

FIGURE A.6. Distortion errors using the LBG algorithm with fixed color space Quantized RGB and varying the metric used for car image.

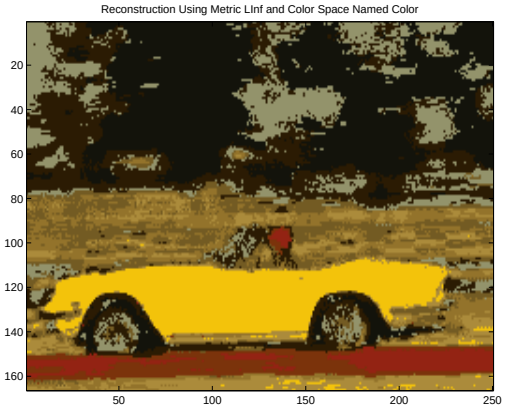
(A) Original



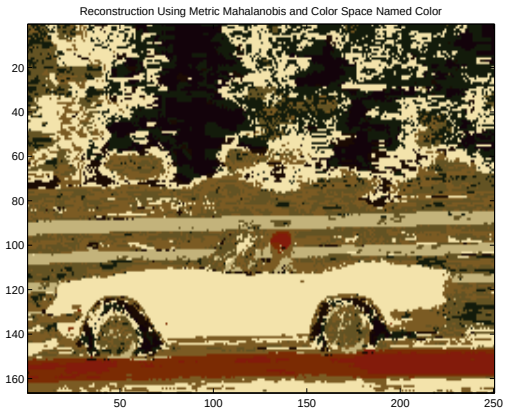
(B) ℓ_1, $E = 2.0509$



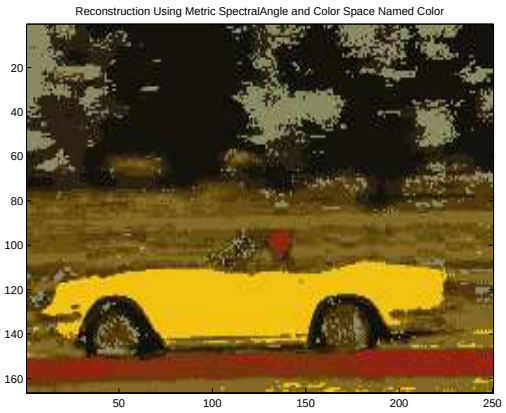
(C) ℓ_2, $E = 2.0313$



(D) ℓ_∞, $E = 2.0339$

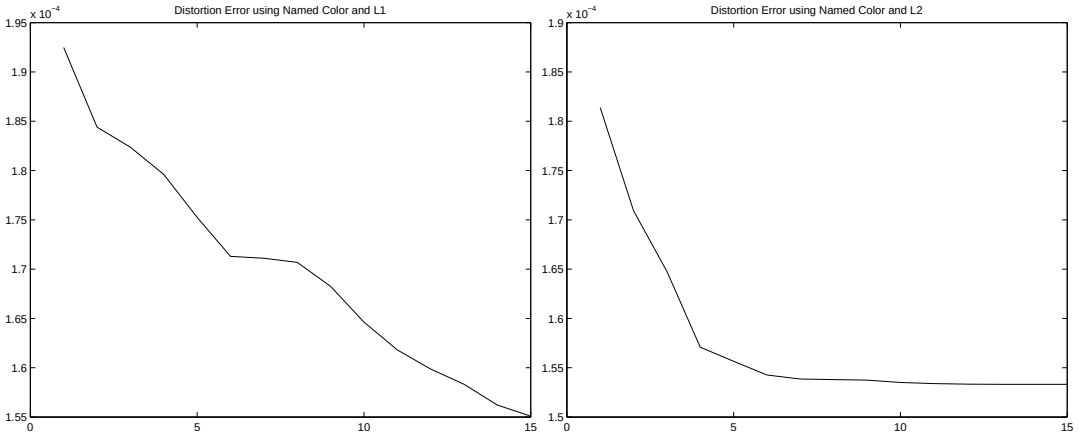


(E) Mahalanobis, $E = 1.9353$



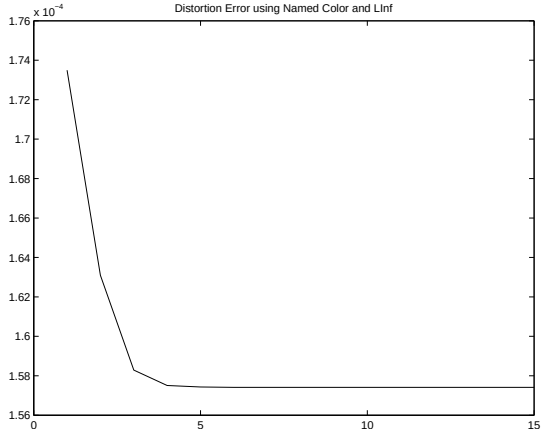
(F) Spectral Angle, $E = 2.0529$

FIGURE A.7. Reconstructions using the LBG algorithm with fixed color space Named Color and varying the metric used for car image.

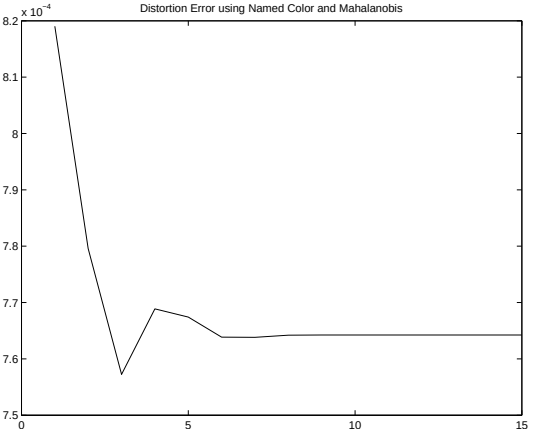


(A) ℓ_1

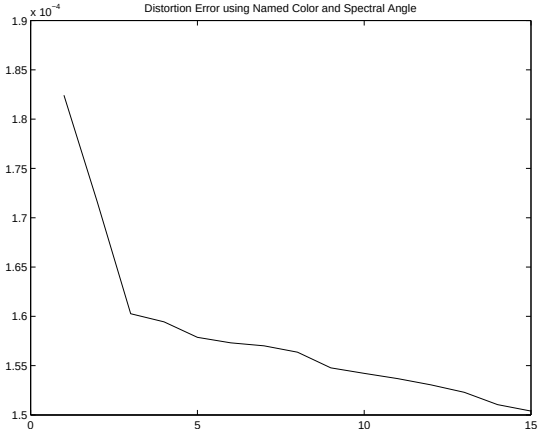
(B) ℓ_2



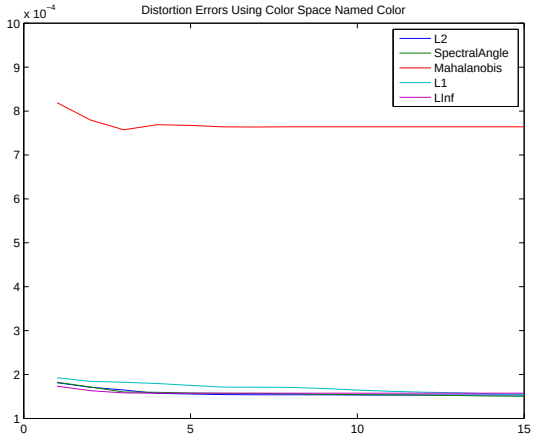
(C) ℓ_∞



(D) Mahalanobis



(E) Spectral Angle



(F) All

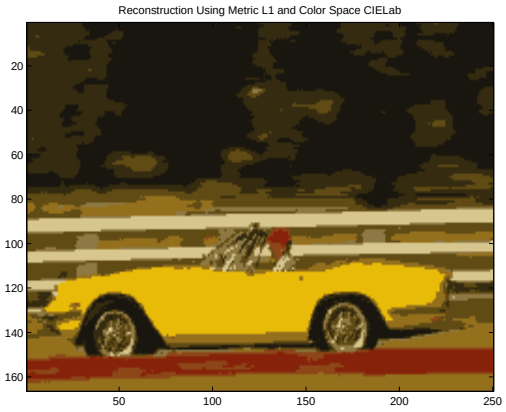
FIGURE A.8. Distortion errors using the LBG algorithm with fixed color space Named Color and varying the metric used for car image.

We will finally consider this image in the CIELab color space varying all metrics, Figure A.9. Recall that CIELab was designed to be more perceptually uniform as measured with the Euclidean distance. Visually it seems that the ℓ_2 norm does produce the best reconstruction. However, all of the cars appear to be flattened (except spectral angle) as compared to the other color spaces. Again, spectral angle produced an image that appears to be smudged and has more of a greenish appearance than is present in the original image. The distortion errors seem comparable to the RGB color spaces in that spectral angle has the most error while the ℓ_p norms have the smallest error, Figure A.8. However, the error is much lower for CIELab. This is a fair comparison with the RGB color spaces as they are both measured in $\mathbb{R}^3$.

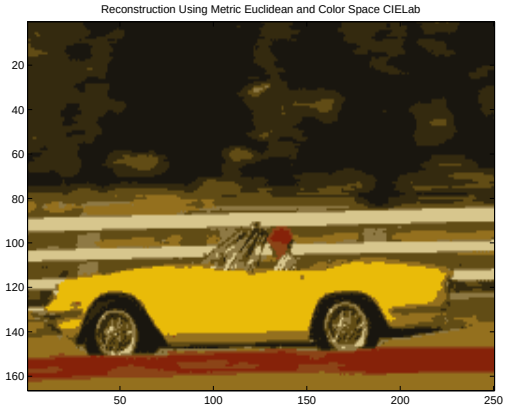
It seems that across all color spaces the ℓ_p norms give the best visual reconstructions for this image.

Now, let us consider the still life image, Figure A.11. We will first focus on the RGB color space across metrics. Again, the ℓ_p norms seem to visually give the best reconstructions. There is a bit more pixellation using the ℓ_∞ norm, the shading on the bread seems more gradated in the ℓ_2 norm, but the red is a bit more vivid in the ℓ_1 norm. The Mahalanobis distance seems to smooth out the colors for each object with far less shading. However, the individual colors are more vivid and distinct. The spectral angle seems to blur objects together and gives the worst visual reconstruction. There appears to be some yellow pixels in the bread using this metric, which is not the case for the others. Notice, that the entropy is much higher for this image than the car image. This is due to the fact that there are more centers representing the data, and the image has a more uniform coloring. Finally, notice

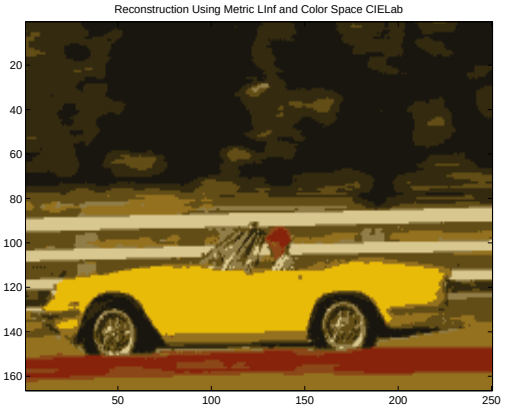
(A) Original



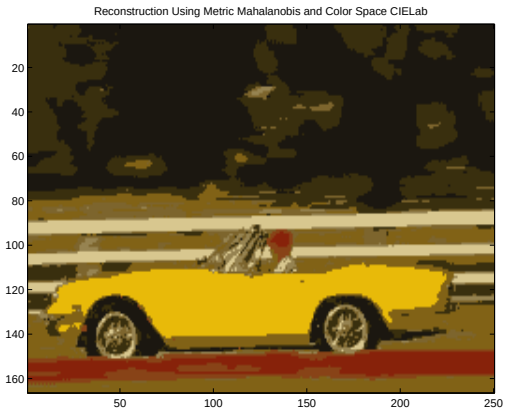
(B) ℓ_1, $E = 1.9381$



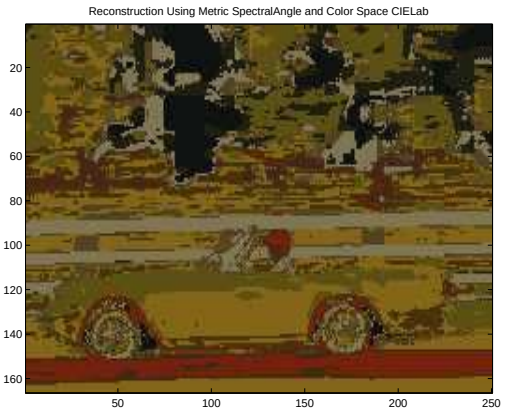
(C) ℓ_2, $E = 1.9399$



(D) ℓ_∞, $E = 1.944$

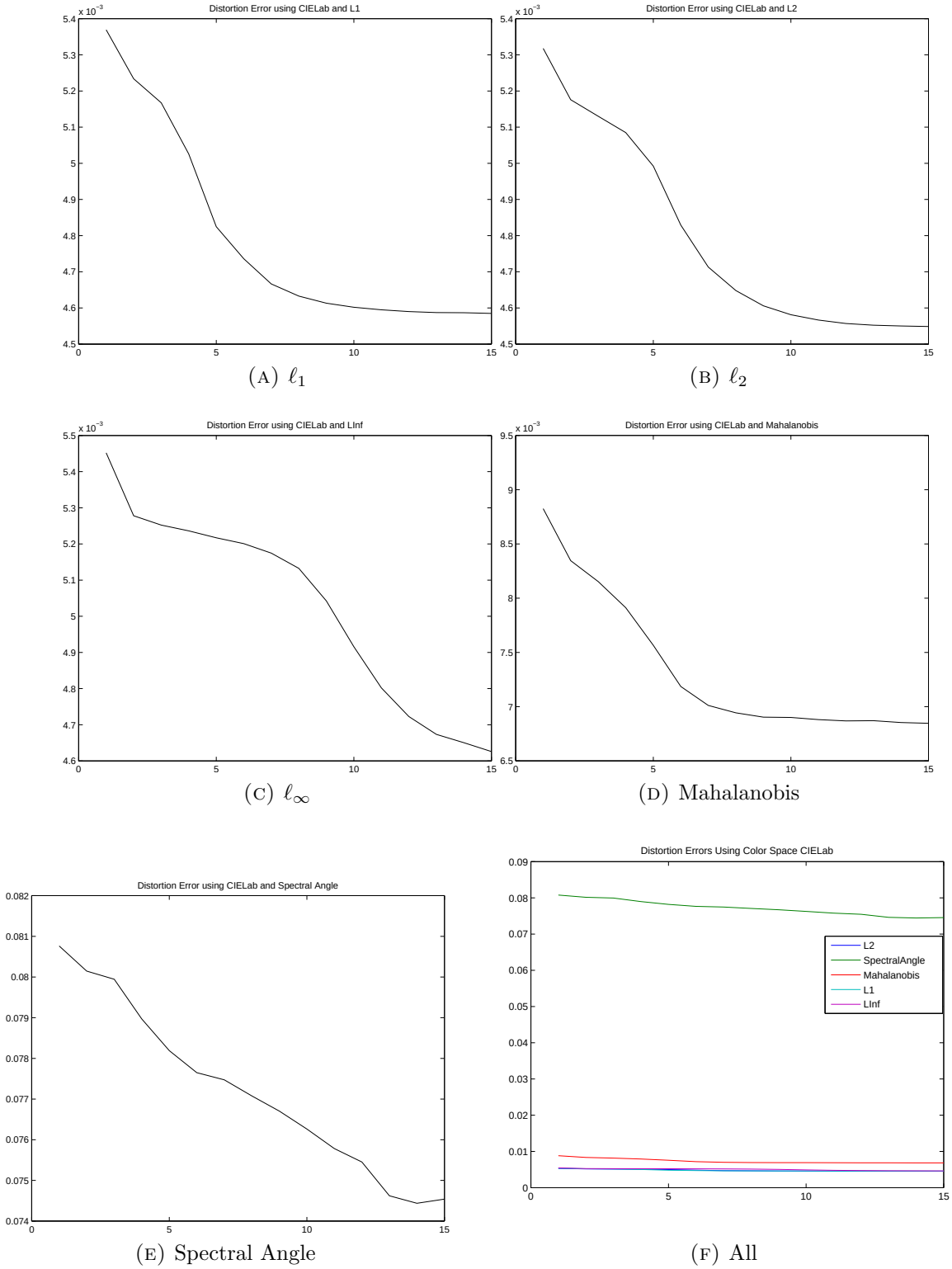


(E) Mahalanobis, $E = 1.8737$



(F) Spectral Angle, $E = 2.0725$

FIGURE A.9. Reconstructions using the LBG algorithm with fixed color space CIELab and varying the metric used for car image.



(A) ℓ_1 (B) ℓ_2

(C) ℓ_∞ (D) Mahalanobis

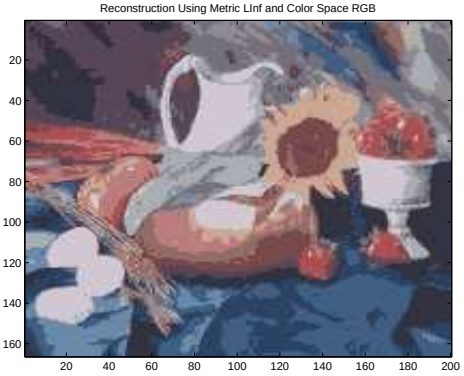
(E) Spectral Angle (F) All

FIGURE A.10. Distortion errors using the LBG algorithm with fixed color space CIELab and varying the metric used for car image.

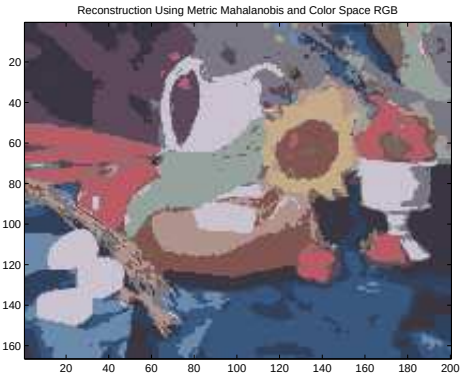
(A) Original

(B) ℓ_1, $E = 2.589$

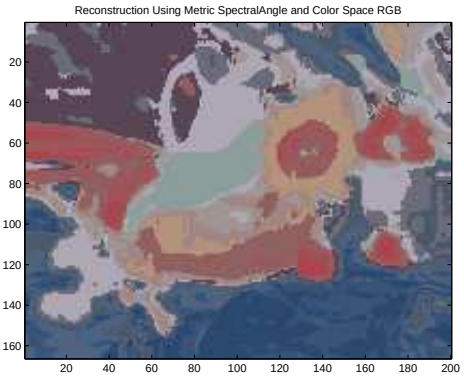
(C) ℓ_2, $E = 2.5993$



(D) ℓ_∞, $E = 2.5728$



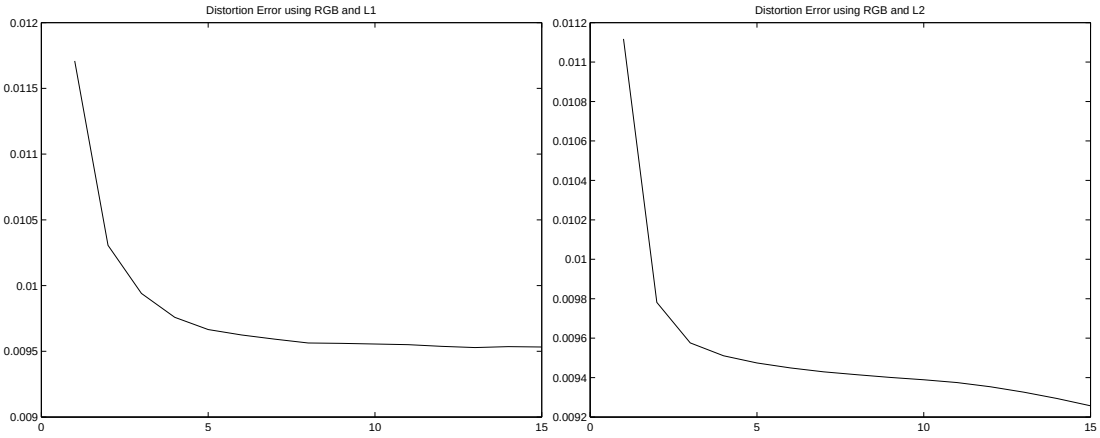
(E) Mahalanobis, $E = 2.5384$



(F) Spectral Angle, $E = 2.5647$

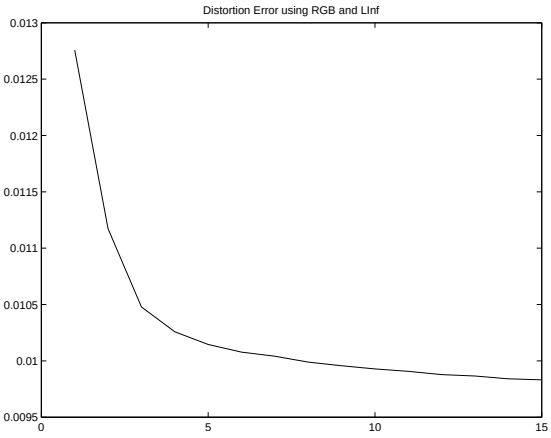
FIGURE A.11. Reconstructions using the LBG algorithm with fixed color space RGB and varying the metric used for still life image.

that, as might be expected, the distortion error for the Mahalanobis distance and spectral angle is greater than for the ℓ_p norms, Figure A.12.

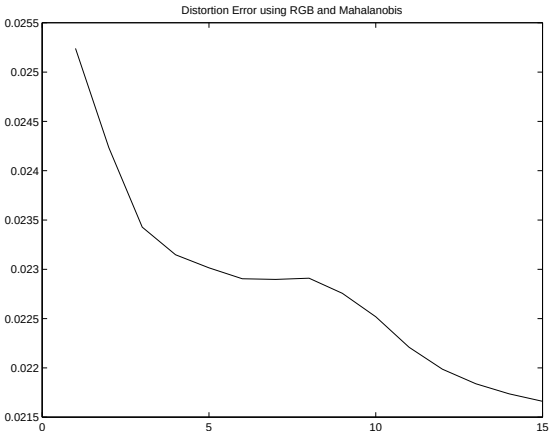


(A) ℓ_1

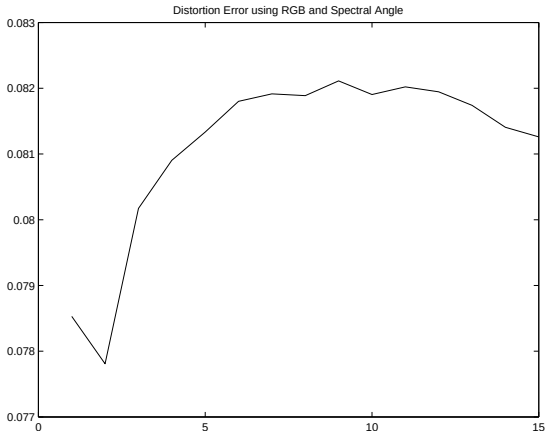
(B) ℓ_2



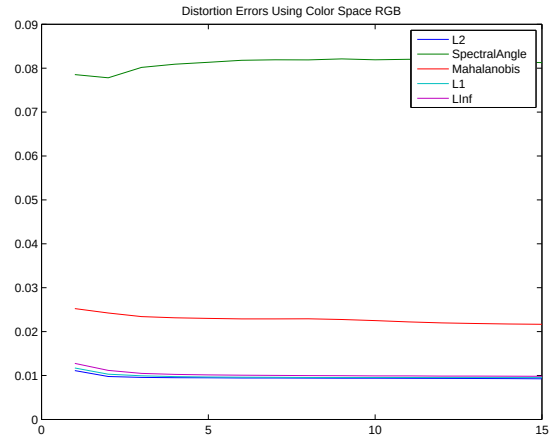
(C) ℓ_∞



(D) Mahalanobis



(E) Spectral Angle



(F) All

FIGURE A.12. Distortion errors using the LBG algorithm with fixed color space RGB and varying the metric used for still life image.

In looking at the still life image in the Quantized RGB space across all metrics, we see very similar reconstructions as with RGB, Figure A.13. Notice, that the entropy is higher in this space (except for spectral angle and ℓ_2) than in the RGB space. Also notice that the distortion error is higher in the Quantized RGB space than the RGB space across all metrics, Figure A.14.

The Named Color space appears to remove most of the shading in the objects in the still life image as well as add more pixellation to the reconstructions, Figure A.15. Notice all highlights from the tomatoes and strawberries have been removed and there is often bleeding of colors such as the brown and yellow in the sunflower. The entropy is lower than in the RGB color spaces. Visually, it seems that the ℓ_∞ norm gives the worst reconstruction. However, the distortion error is largest using the Mahalanobis distance, Figure A.16.

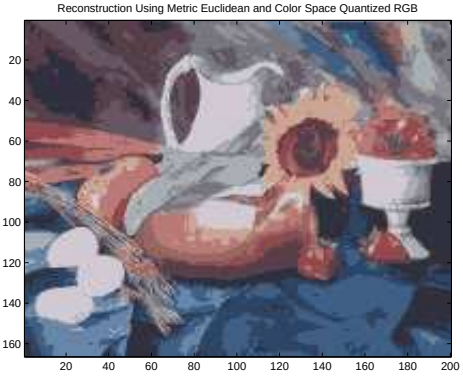
The CIELab color model seems to give fairly uniform reconstructions across all metrics, except spectral angle which appears to be blurred, Figure A.17. These images are not as sharp as using the RGB color models, and there does not appear to be as much shading. However, the distortion errors are much less than as measured in the RGB spaces, Figure A.18.

Again, across all color spaces the ℓ_p, norms appear to give the best visual reconstructions.

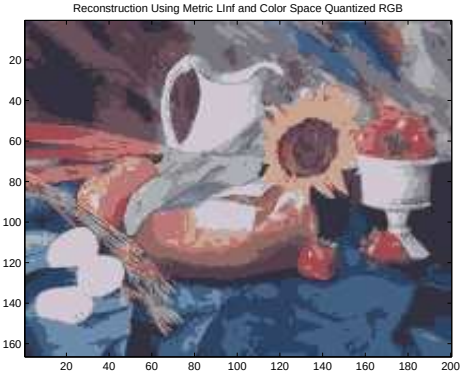
As noted in Section 3.6, we observe that varying the color space and metric greatly affects the clustering reconstruction. Therefore, depending on the application, an appropriate choice may be selected based upon the desired output of the clustering and the type of image to be analyzed.

(A) Original

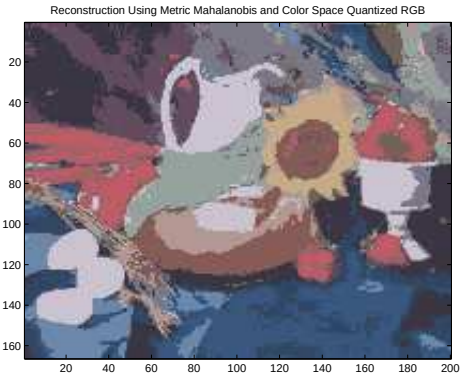
(B) ℓ_1, $E = 2.5921$



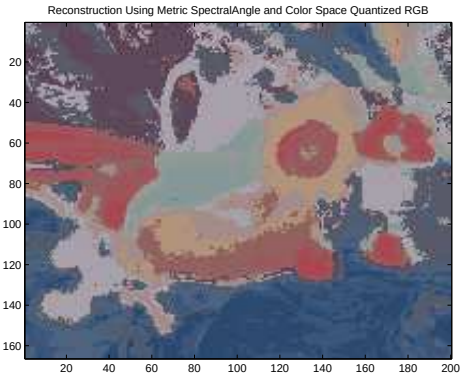
(C) ℓ_2, $E = 2.5912$



(D) ℓ_∞, $E = 2.5783$

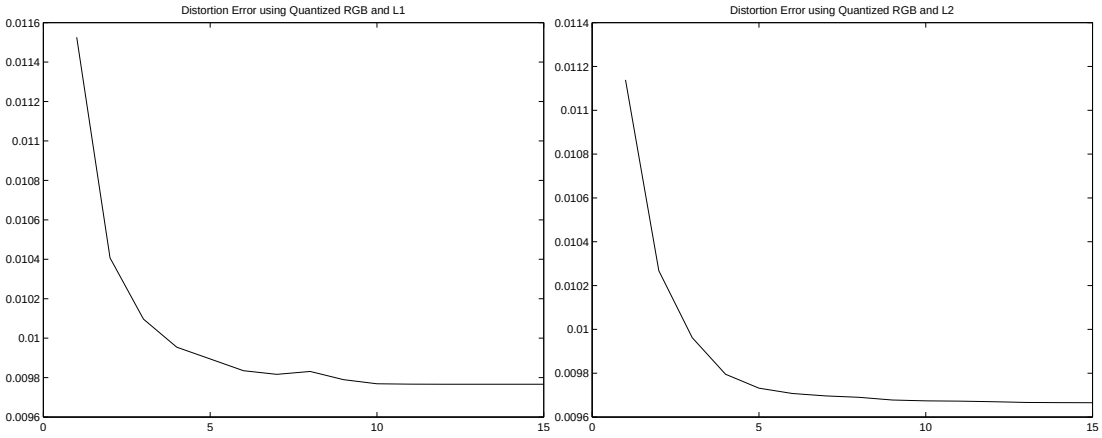


(E) Mahalanobis, $E = 2.5412$



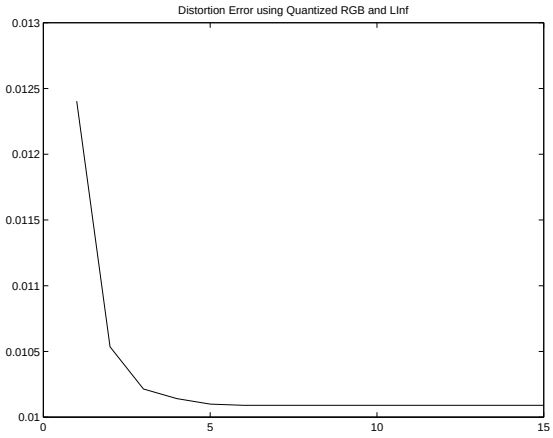
(F) Spectral Angle, $E = 2.5576$

FIGURE A.13. Reconstructions using the LBG algorithm with fixed color space Quantized RGB and varying the metric used for still life image.

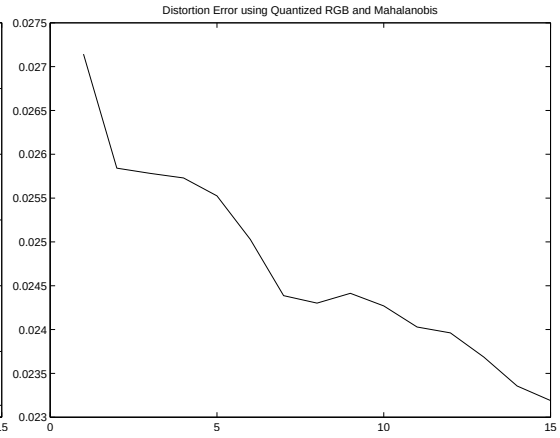


(A) ℓ_1

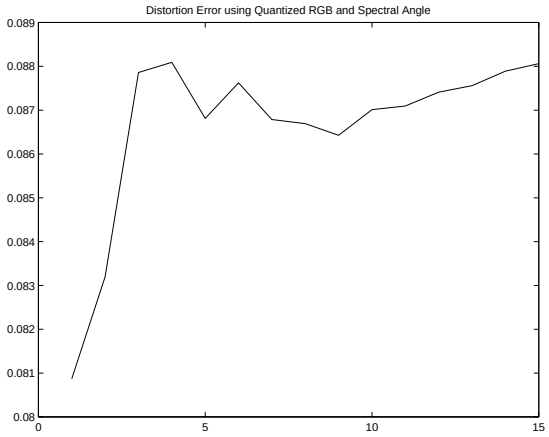
(B) ℓ_2



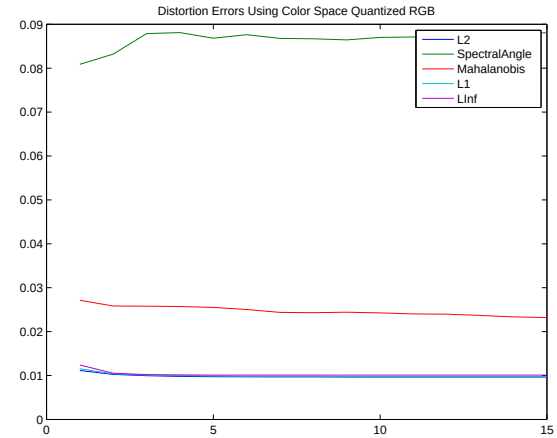
(C) ℓ_∞



(D) Mahalanobis



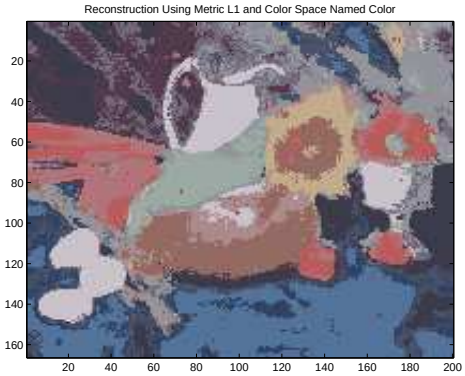
(E) Spectral Angle



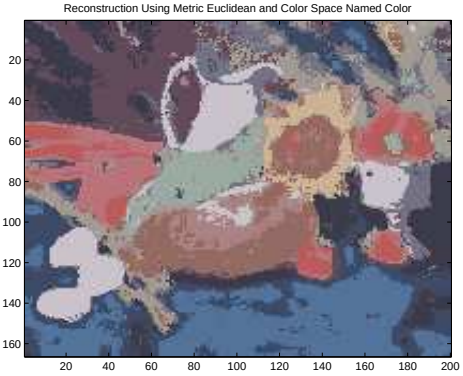
(F) All

FIGURE A.14. Distortion errors using the LBG algorithm with fixed color space Quantized RGB and varying the metric used for still life image.

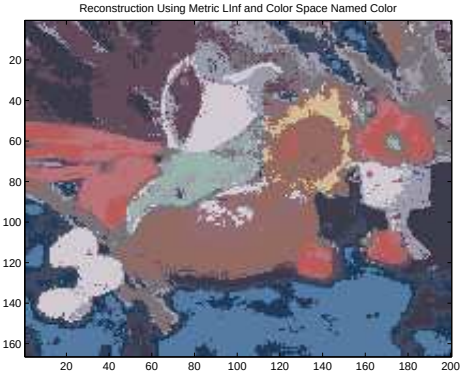
(A) Original



(B) ℓ_1, $E = 2.555$

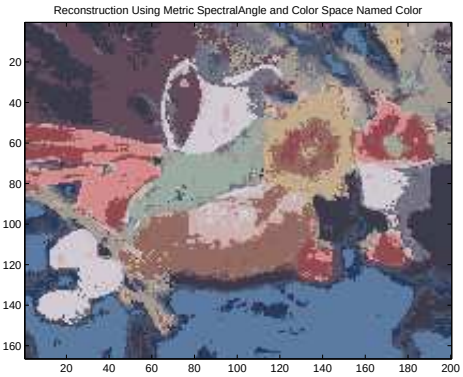


(C) ℓ_2, $E = 2.5436$



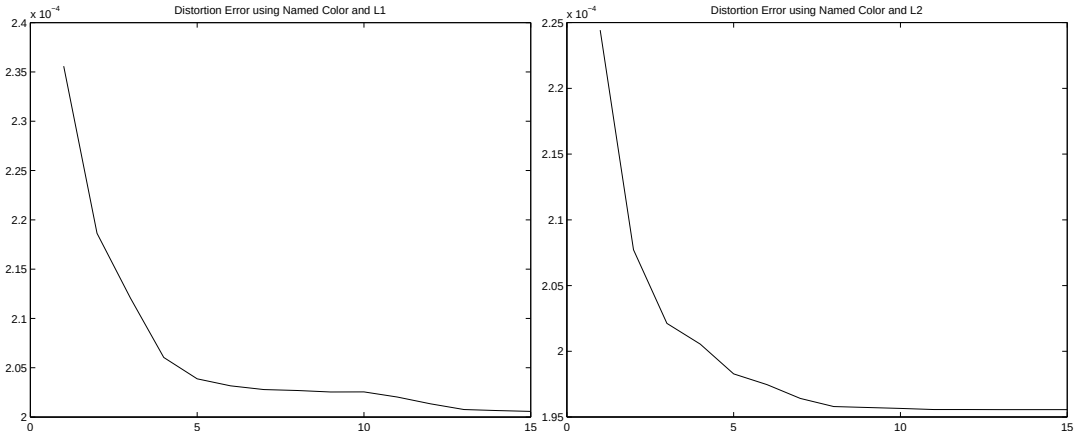
(D) v_∞, $E = 2.5157$

(E) Mahalanobis, $E = 2.4582$

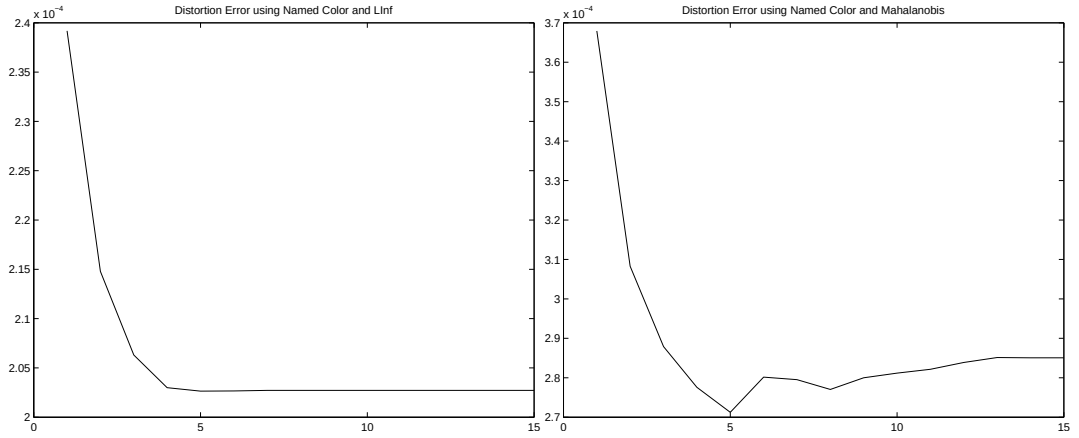


(F) Spectral Angle, $E = 2.5286$

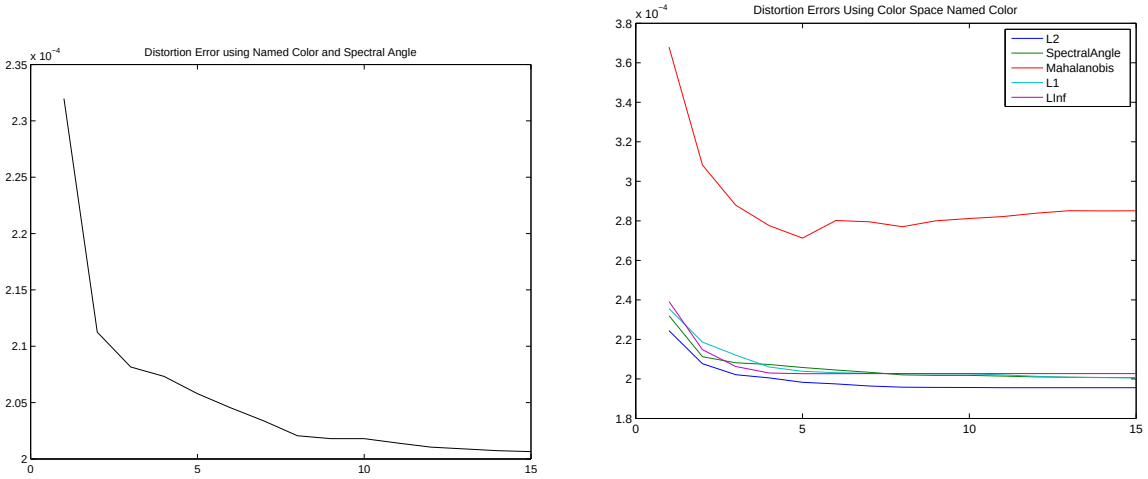
FIGURE A.15. Reconstructions using the LBG algorithm with fixed color space Named Color and varying the metric used for still life image.



(A) ℓ_1 (B) ℓ_2



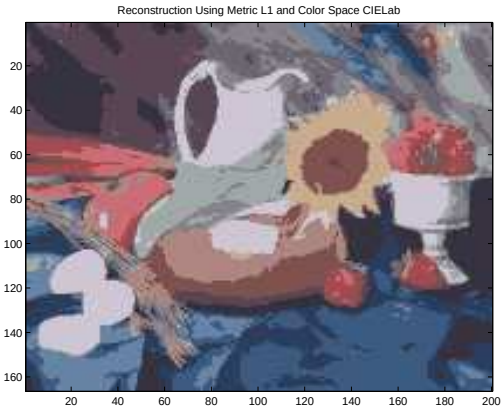
(C) ℓ_∞ (D) Mahalanobis



(E) Spectral Angle (F) All

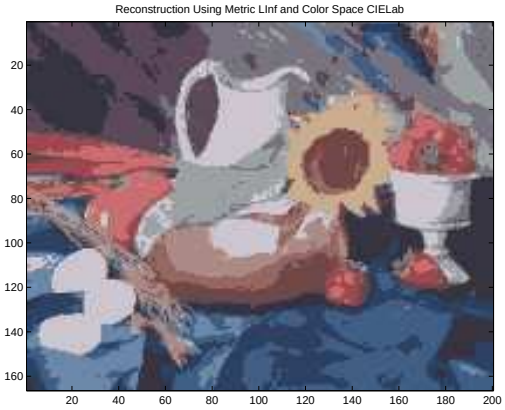
FIGURE A.16. Distortion errors using the LBG algorithm with fixed color space Named Color and varying the metric used for still life image.

(A) Original

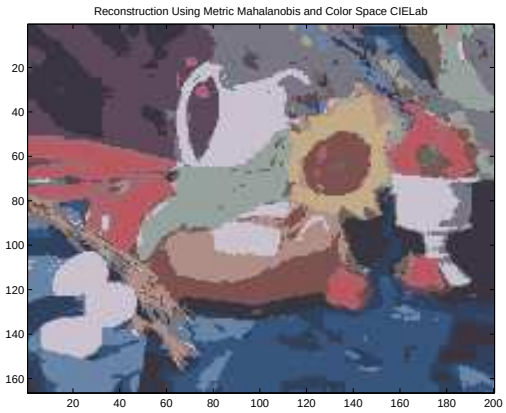


(B) ℓ_1, $E = 2.5657$

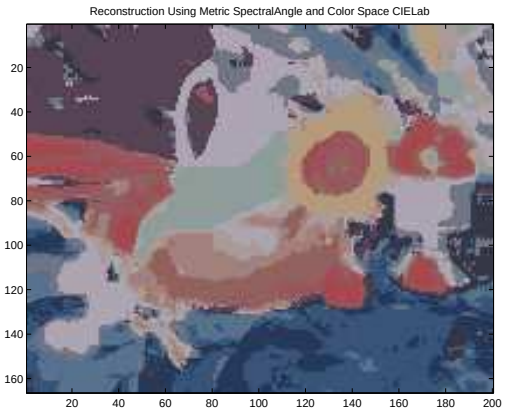
(C) ℓ_2, $E = 2.541$



(D) ℓ_∞, $E = 2.5397$

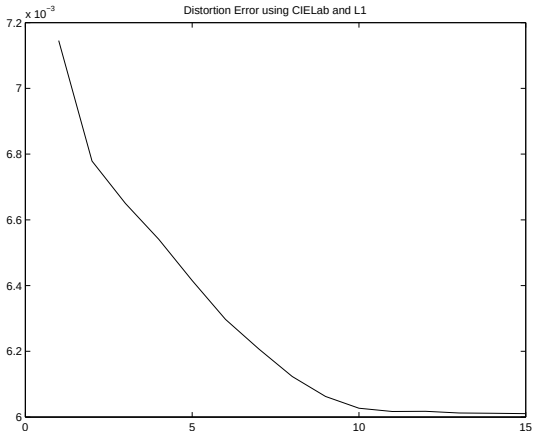


(E) Mahalanobis, $E = 2.5506$

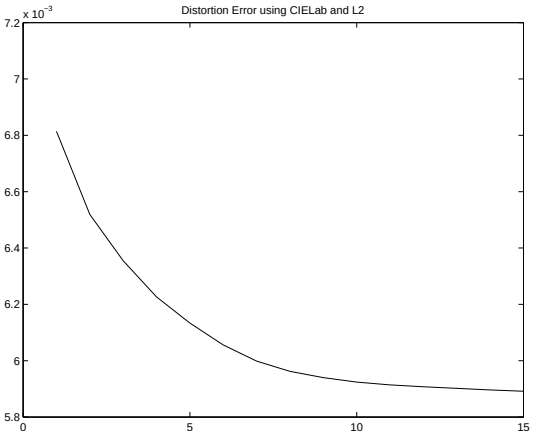


(F) Spectral Angle, $E = 2.5373$

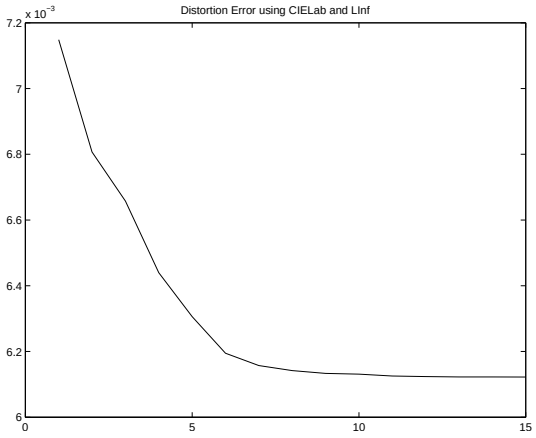
FIGURE A.17. Reconstructions using the LBG algorithm with fixed color space CIELab and varying the metric used for still life image.



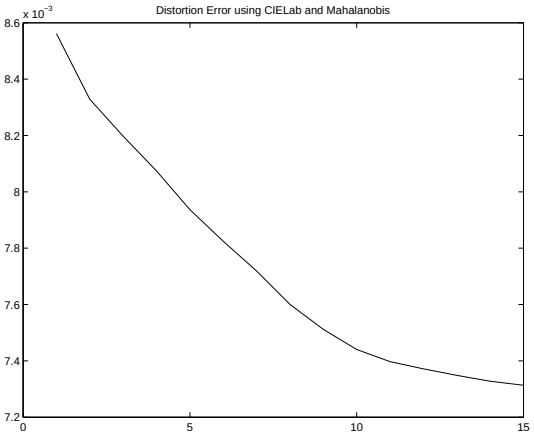
(A) ℓ_1



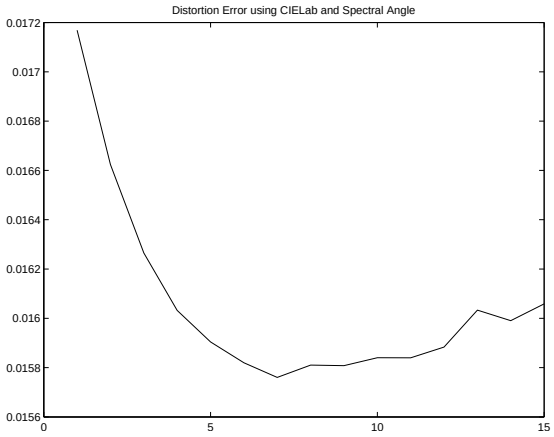
(B) ℓ_2



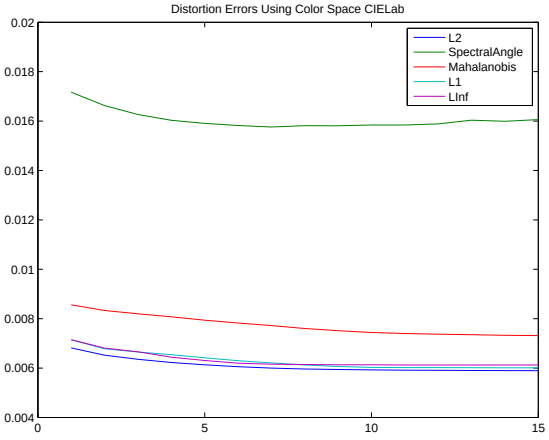
(C) ℓ_∞



(D) Mahalanobis



(E) Spectral Angle



(F) All

FIGURE A.18. Distortion errors using the LBG algorithm with fixed color space CIELab and varying the metric used for still life image.