Sparse Encoding of Observations from a Smooth Manifold via Locally Linear Approximations

by

Nicholas Bertrand

B.S., University of Colorado at Boulder, 2012

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Applied Mathematics

2012

This thesis entitled:
Sparse Encoding of Observations from a Smooth Manifold via Locally Linear Approximations
written by Nicholas Bertrand
has been approved for the Department of Applied Mathematics

_____
Dr. François Meyer

_____
Dr. James Curry

_____
Dr. Shannon Hughes

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Bertrand, Nicholas (M.S., Applied Mathematics)

Sparse Encoding of Observations from a Smooth Manifold via Locally Linear Approximations

Thesis directed by Professor François Meyer

We investigate the problem of finding a parameterization of a smooth, low-dimensional manifold based on noisy observations from a high-dimensional ambient space. The formulation of such parameterizations sees applications in a variety of areas such as data denoising and image segmentation.

We introduce algorithms inspired by the existing k-svd algorithm for training dictionaries for sparse data representation, and the local best-fit flat algorithm for hybrid linear modeling. The output of our algorithm is an assignment of input data points to locally linear models. To demonstrate the applicability of our algorithm, we discuss experiments performed on synthetic datasets.

Acknowledgements

I am extremely grateful to Professor François Meyer for his continued support. His guidance, patience, and advice have been crucial in my development as a researcher.

I would also like to thank the members of my thesis committee for their time.

Finally, many thanks go to my family for their unconditional love and encouragement.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

1.1 Motivation

In many application areas, scientists collect data from a high-dimensional ambient space. In general such data is extremely difficult to analyze and process. However, in many cases, there is an underlying low-dimensional structure which may be exploited to gain unique insight from the data. Consider the simple example of taking observations from a d dimensional shell centered at the origin in $\mathbb{R}^D$ with $d < D$. Although each measurement consists of D values, each of them can be expressed as a function of only d values, $f(x_1, \ldots, x_d)$.

However, when considering real data, the definition of such a parametrization f is not so obvious. For example, consider a set of greyscale images of human faces. Suppose there are $m \times n$ pixels in each image, each holding a value in the range 0-255. Each image may be thought of as a point in $\mathbb{R}^{m \times n}$. However, since each image contains a human face, there are certain combinations of pixels that could never be observed. One may further argue that there are in fact a set of $d \ll nm$ degrees of freedom that can describe each image, for example the size and position of facial features.

In this thesis, we model data as coming from a smooth manifold. This model has been used in a variety of areas such as object tracking [11] and image segmentation [20]. Our goal is to construct a parameterization of the manifold using a set of locally linear approximations. As an example, we now refer back to the shell. Consider a circle in $\mathbb{R}^2$. Given a set of samples, our goal is to approximate the origin object with a set of lines.

Figure 1.1 provides an overview of this process.

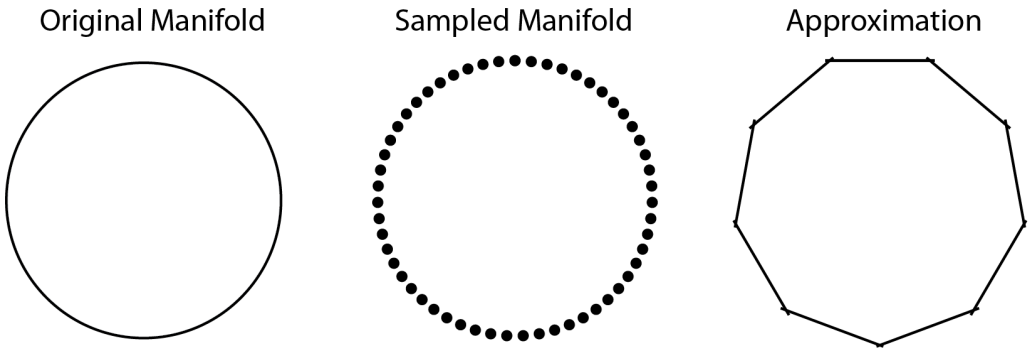

Figure 1.1: Illustration of goal. The original manifold, a circle in this case, is approximated using 9 line segments.

Naturally, we would like our approach to generalize to higher dimensions. In such cases we approximate d-dimensional manifolds embedded in $\mathbb{R}^D$ using a union of affine subspaces.

1.2 Formal Problem Statement

Consider the matrix $\mathbf{X} \in \mathbb{R}^{D \times N}$ whose columns are observations from a manifold of known dimension d embedded in $\mathbb{R}^D$. The goal of this thesis is to develop an algorithm that produces a parameterization of input manifold observations using locally linear approximations. The output should consist of an assignment of the input data to a corresponding linear model. In other words, we wish to compute the matrix $\mathbf{A} \in \mathbb{R}^{D \times kd}$, $\mathbf{X} \in \mathbb{R}^{kd \times N}$, and $\mathbf{B} \in \mathbb{R}^{D \times N}$ such that

$$\mathbf{Y} \approx \mathbf{AX} + \mathbf{B} \tag{1.1}$$

where k is the number of linear models used to approximate data in $\mathbf{Y}$. Each column of $\mathbf{X}$ represents an approximation of a corresponding point in $\mathbf{Y}$ using the d basis vectors of the linear model to which it has been assigned. Hence, columns of $\mathbf{X}$ are d-sparse, and the locations of non-zero entries are common among points being represented by the same

linear model. Each column $\mathbf{b}_i$ in $\mathbf{B}$ corresponds to the center of mass of the model to which point $\mathbf{x}_i$ belongs.

We would like to choose our flats such that global approximation error is minimized. In other words, we would like to solve the following optimization problem:

$$\min_{\mathbf{A},\mathbf{B}} \|\mathbf{Y} - (\mathbf{A}\mathbf{X} + \mathbf{B})\| \tag{1.2}$$

with the geometric constraints discussed above.

1.3 Organization of this Thesis

In Chapter 2, we briefly discuss related works, and provide an overview of two relevant algorithms: K-SVD [1] and Local Best-fit Flats (LBF) [22]. As the algorithms proposed in this thesis are inspired by K-SVD and LBF, a review of these algorithms is beneficial. In Chapter 3, we describe the algorithms introduced in this thesis. Chapter 4 goes over several experiments using the new algorithms. A discussion of algorithm performance and limitations is given in Chapter 5. Concluding remarks and future research directions are discussed in Chapter 6.

Chapter 2

Background

In recent years, much work has been done in attempt to solve problems that are closely related to the one considered in this thesis. Related works include Local Best-fit Flats (LBF), K-SVD, Spectral Curvature Clustering [4], k-planes clustering [16] [17] [21], and General Principle Component Analysis [18]. The first two algorithms, LBF and K-SVD were of particular influence in the development of the algorithms introduced in this thesis. For this reason, we provide a brief introduction to each of these algorithms so the reader will have some intuition as we introduce two new algorithms.

2.1 Local Best-fit Flats

2.1.1 Review of Algorithm

Local Best-fit Flats (LBF) was designed to solve a similar problem to our own. Consider a set of data sampled from a union of affine subspaces (flats). The goal of LBF is to recover a parameterization of the underlying structure. The algorithm is based on geometry, and works by generating a set of c candidate flats, L. From the set L, a subset of "active" flats, $\hat{L}$ is formed. Points are assigned by choosing the nearest flat in $\hat{L}$. In each iteration, a randomly chosen element of $\hat{L}$ is updated by replacing it with an element of L such that the global approximation error is reduced. The An overview of LBF is as follows:

Algorithm 1: Local Best-fit Flats (LBF)

Input : Data points $\mathbf{Y} \in \mathbb{R}^{D\times N}$, d: dimension of subspaces, c: number of candidate flats, k: number of output flats, p: number of update passes

Output: Assignment of points in $\mathbf{X}$ to d-dimensional flats

Choose c points from $\mathbf{X}$ (call this set C) and use an appropriately sized neighborhood around each point to construct a flat. Call this set of flats L.

Choose k flats from L and call this subset $\hat{L}$.

for $i = 1$ **to** p **do**

  Choose a random flat in $\hat{L}$ and replace it with a flat from L such that maximum improvement of the current approximation is achieved.

Assign points in $\mathbf{X}$ to their nearest flat in $\hat{L}$.

An important step in the LBF algorithm is the computation of the "optimal" neighborhood size from which we compute a flat. In order to compute the size of the optimal neighborhood, we consider the following value:

$$\beta(\mathcal{N}) = \frac{\min_{\text{d-flats L}} \sqrt{\sum_{\mathbf{y}\in\mathcal{N}} \|\mathbf{y} - P_L\mathbf{y}\|^2/|\mathcal{N}|}}{\max_{\mathbf{x}\in\mathcal{N}} \|\mathbf{x} - \mathbf{x}_0\|} \tag{2.1}$$

Suppose we wish to construct a flat around the point $\mathbf{x}_0$. We begin by taking a neighborhood consisting of $\mathbf{x}_0$ and its S nearest neighbors. We then use (2.1) to compute the neighborhood's corresponding beta number. The size of the neighborhood is then increased incrementally by T until a local minimum of (2.1) is found.

Theoretical justification for this method is provided in [22], and is not repeated here. However, we offer the following to assist the reader's intuition. In the presence of noise, using more points to recover a flat typically results in a more accurate representation. This result is considered in more detail in most texts in numerical analysis, for example [2]. Keeping this in mind, suppose we compute $\beta(\mathcal{N})$ for the smallest possible neighborhood and incrementally increase the size of the neighborhood as described above. The numerator, which is normalized approximation error, will experience little growth at first, while the denominator, which is the radius of the neighborhood continues to grow. This causes the net effect of $\beta(\mathcal{N})$ decreasing. However, when points from an adjacent flat begin to enter the neighborhood, the approximation error becomes worse and $\beta(\mathcal{N})$ begins to grow. Thus, the

local minimum corresponds to the largest neighborhood containing only points on the same flat on which $\mathbf{x}_0$ resides.

2.1.2 Discussion

In a sense, we wish to generalize LBF by developing a similar algorithm which allows for the underlying structure from which we sample to have curvature. For the moment, suppose we obtain samples from a union of affine subspaces. As long as the initial set of c points used in LBF contains one point in each of the flats, we obtain perfect recovery in most cases[1] . Thus, there is little sensitivity to the location of these points on their corresponding flats. However, should a similar approach be applied to an object with curvature, the location of the c flats is critical. Although this problem could be solved by choosing a very high value for c, we would like to try a different approach. The relationship of LBF to the algorithms proposed in this thesis is discussed further in Chapter 3.

2.2 K-SVD

2.2.1 Review of the Algorithm

K-SVD is an algorithm for generating training over-complete dictionaries. In an over-complete dictionary, the number of elements exceeds the dimensionality of the data which they are intended to represent and are hence not orthogonal. The problem is then to represent a given point using as few elements from the dictionary as possible.

The name comes from the mixture of the k-means clustering algorithm and the singular value decomposition (SVD) of a matrix. K-SVD can be thought of as a generalization of k-means clustering in which points in a cluster are represented by a single point to a version in which points are represented as a linear combination of dictionary atoms.

The basic idea behind K-SVD is to first form an initial dictionary using elements

[1] Note: An additional condition for this to hold it that the neighborhood constructed around each of these points is contained in a single flat.

directly from the dataset. Dictionary atoms are then updated iteratively by using the first singular vector of a matrix related to the residual of the current approximation. An overview of the algorithm is as follows:

Algorithm 2: K-SVD (Details Omitted)

Input : Dataset $\mathbf{Y} \in \mathbb{R}^{n \times N}$
Output: A dictionary $\mathbf{D} \in \mathbb{R}^{n \times m}$, and a sparse encoding $\mathbf{X}$ such that $\mathbf{DX} \approx \mathbf{Y}$

Initialize: Fill $\mathbf{D}$ by selecting random points from $\mathbf{Y}$.
repeat
  Sparse Encoding: Use any sparse encoding scheme such as orthogonal matching pursuit (OMP) to obtain $\mathbf{X}$ for the current dictionary.
  Update Dictionary:
  foreach $\mathbf{d}_k \in \mathbf{D}$ **do**
    Form the set ω which contains the indices of the data points which are used by atom $\mathbf{d}_k$.
    Compute an error quantity associated with atom $\mathbf{d}_k$: $\mathbf{E}_k = \mathbf{Y} - \sum_{i \neq k} \mathbf{d}_i \mathbf{x}_T^{(i)}$.
    Restrict $\mathbf{E}_k$ by removing columns that are not in ω, call it $\mathbf{E}_k^R$.
    Compute the SVD $\mathbf{E}_k^R = \mathbf{U\Sigma V^T}$.
    Set $\mathbf{d}_k$ equal to the first column of $\mathbf{U}$, and set $\mathbf{x}_T^{(i)}$ equal to $\Sigma_{1,1} V_1$.
until stopping criterion

2.2.2 Discussion

The adaptation of KSVD to address our problem is significantly different than the original KSVD algorithm. For this reason we do not provide an in-depth analysis here. We refer interested readers to [3] and [1] for more details. Readers who need a review on SVD itself may find an elementary introduction in [14]. In Chapter 3, we discuss how K-SVD provided us with a starting point to develop an algorithm for our problem.

2.3 Other Related Work

2.4 Contributions

To our knowledge, no one has proposed an algorithm which utilizes locally linear models to approximate a manifold. In this work, we propose two algorithms to solve the optimization problem described in (1.2). In order to demonstrate the effectiveness of each algorithm, we experiment with synthetic datasets. We also discuss issues of sampling, scale, and noise. The primary contribution of this thesis is an algorithm inspired by local best-fit flats which is tailored to work on general smooth manifolds rather than unions of affine subspaces. We effectively eliminate the need to provide the number of "guess clusters" as an input by dynamically exploring the landscape of the input manifold observations.

Chapter 3

Overview of Algorithms

In this chapter, we introduce two new algorithms. Details of the algorithms as well as their connection to the ones discussed in Chapter 2 are discussed.

3.1 Manifold Best-fit Flats

3.1.1 Overview of Algorithm

The first algorithm we introduce is based on LBF. The intention of the algorithm was to dynamically explore the terrain of the input manifold by constructing new candidate flats iteratively after the initial set L is formed. To do so, we attempt to identify regions of the manifold which are poorly represented by the approximation at a given iteration and construct additional flats to correct the model. When choosing where to construct new flats, we rank order error in neighborhoods rather than error in individual points to avoid wasting resources by constructing flats around outliers. Also note that the neighborhood sizes are not necessarily the optimal ones, since it would be extremely computationally expensive to compute the optimal neighborhood around every point in the dataset. The procedure is described below.

Algorithm 3: Manifold Local Best-fit Flats (MLBF)

Input : Data points $\mathbf{Y} \in \mathbb{R}^{D \times N}$, d: dimension of subspaces, c: number of candidate flats, k: number of output flats, p: number of update passes

Output: Assignment of points in $\mathbf{X}$ to d-dimensional flats

Choose c points from $\mathbf{X}$ (call this set C) and use an appropriately sized neighborhood around each point to construct a flat. Call this set of flats L.

Choose k flats from L and call this subset $\hat{L}$.

for $i = 1$ **to** p **do**

- Choose a random flat in $\hat{L}$ and replace it with a flat from L such that maximum improvement of the current approximation is achieved.
- For each point in $\mathbf{y}_i \in \mathbf{Y}$, compute the error in a neighborhood of $\mathbf{y}_i$, choosing the neighborhood size to be equal to the neighborhood size associated with the flat representing $\mathbf{y}_i$.
- Rank order the approximation error and generate flat centered around the point corresponding to the highest approximation error.
- Attempt to replace a flat in $\hat{L}$ with the new flat. If there is a replacement such that the global error is reduced, make the optimal replacement and add the new flat to L.

Assign points in $\mathbf{X}$ to their nearest flat in $\hat{L}$.

3.1.2 Scale: Choosing an Optimal Neighborhood

The method for choosing an optimal neighborhood size remains unchanged from the original LBF algorithm. Again, we do not provide theoretical justification for this choice, although theoretical results in [10] suggest that an optimal neighborhood indeed exists. However, the same intuition from LBF applies. In this case, the numerator of (2.1) experiences little growth at the scale where curvature is low. By incrementally increasing the neighborhood size, curvature in the underlying manifold cause the linear approximation to fail, causing an eventual increase in $\beta(\mathcal{N})$. This method is verified experimentally in Section 4.1.5.

3.2 Manifold K-SVD

In this section, we propose Manifold K-SVD (MKSVD), an algorithm inspired by K-SVD. We begin by obtaining an initial clustering of our data, using an algorithm such as k-means. Flats, whose basis vectors are analogous to dictionary atoms in standard K-SVD,

are then computed using all of the points in each cluster. Updates are made by reassigning points at the boundary between clusters in a way such that global approximation error is decreased. In this method, updated flats are the result of reassigning points. This contrasts MLBF where the assignment of points is the result of updating flats. A description of MKSVD is shown below.

Algorithm 4: Manifold K-SVD (MKSVD)

Input : Data points $\mathbf{Y}, k$: number of clusters
Output: Assignment of points in $\mathbf{X}$ to d-dimensional flats

Initialization: Cluster the input data (for example using k-means)
repeat
  Compute the best fit flat for the points in each cluster $\mathbf{C}_i$
  Reassign points:
    Check if the distance to the nearest in-cluster neighbor of $\mathbf{y}_i$ is similar to that of the nearest out-of-cluster neighbor of $\mathbf{y}_i$.
    if $\mathbf{y}_i$ is indeed at the boundary then
      Try moving $\mathbf{y}_i$ to each neighboring cluster, and recompute the best-fit flat and the associated approximation error of the cluster from which $\mathbf{y}_i$ has been removed and the one to which it has been added.
      Move $\mathbf{y}_i$ to the cluster such that this approximation error is minimized.
until stopping criterion

The next chapter explores experiments performed using MLBF and MKSVD.

Chapter 4

Experiments

4.1 Experiments with MLBF

In this section, we analyze the performance of MLBF through a series of experiments. Although MLBF may be used on datasets of arbitrary dimension, the experiments here are performed on curves in $\mathbb{R}^2$ and surfaces in $\mathbb{R}^3$ for simplicity. The experiments were intended to highlight particular aspects of MLBF's performance. In each of the following sections, we describe an experiment including the motivation for the dataset used and the sampling method. For each experiment, we compute the following values to analyze the approximation error in the output of MLBF

$$E_\mu = \frac{\sum_i \|\mathbf{y}_i - \mathbf{P}_i\mathbf{y}_i\|_2}{N} \tag{4.1}$$

$$E_\infty = \max_i \|\mathbf{y}_i - \mathbf{P}_i\mathbf{y}_i\|_2 \tag{4.2}$$

where N is the number of points in dataset $\mathcal{D}$, $\mathbf{y}_i$ is the i-th point in dataset $\mathcal{D}$, and $\mathbf{P}_i\mathbf{y}_i$ is the projection of point i onto its corresponding flat.

We visualize the output of MLBF by assigning a color to points belonging to the same flat. Additionally, a grey line indicating the flat used to approximate a given set of points is drawn.

Since MLBF is initialized randomly, we repeat each experiment one hundred times and report the mean and standard deviation for E_μ and E_∞. In this manner, the typical behavior

of the algorithm may be observed.

The number of flats, k, varies among the experiments. The number of initial guess flats is set to $c = k$, and the maximum number of iterations is $p = 5k$.

4.1.1 Curvature

In the first experiment, our goal was to observe the distribution of flats produced by MLBF. We expected that more flats would cluster in regions of higher curvature where the fidelity of the locally linear approximation is lowest. In order to test this hypothesis, we performed an experiment on the dataset described in Table 4.1.1.

Table 4.1: Description of dataset $\mathcal{D}_1$.

Dataset Name:	$\mathcal{D}_1$
Manifold Description:	Parabola in $\mathbb{R}^2$
Sampling Method:	x equispaced on $[-3, 3]$, spacing $h = 0.05$; $y = x^2$
Sample Count:	$N = 121$

MLBF was executed on $\mathcal{D}_1$ with $k = 2, 4, 6, 8$. The results are shown in Figure 4.1.

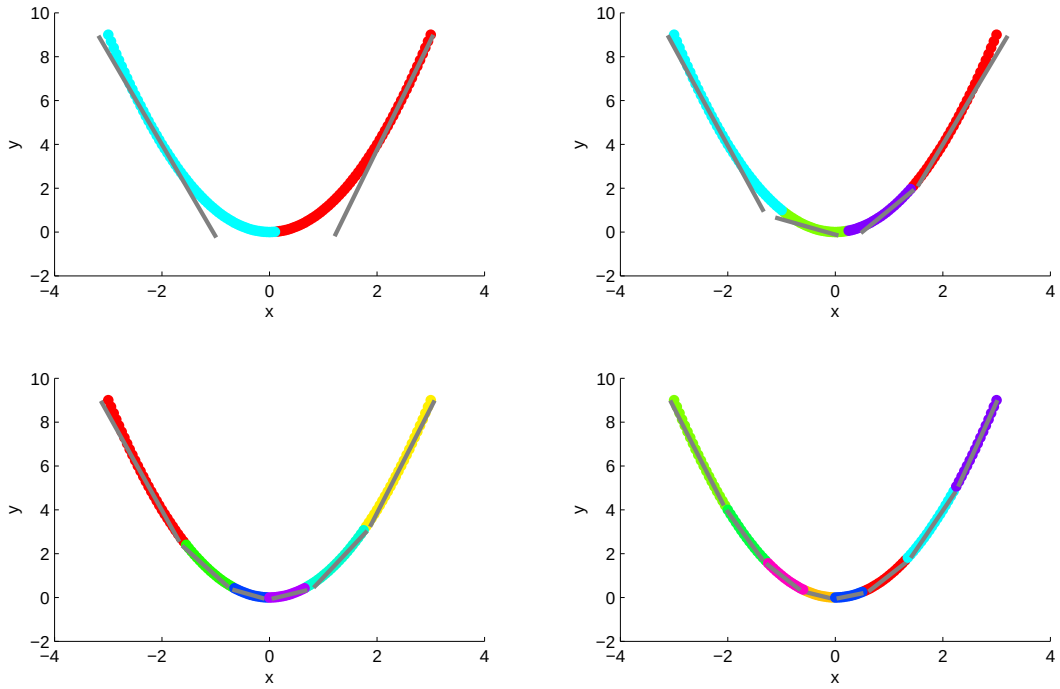

Figure 4.1: Demonstration of convergence rates for MLBF on dataset $\mathcal{D}_1$. Colors represent the assignment of points to different flats. A grey line is drawn to indicate the orientation of each flat.

Note that flats tend to cluster more densely around the origin where curvature is greatest, as expected. A summary of approximation error is found in Table 4.2.

Table 4.2: Error analysis for MLBF on dataset $\mathcal{D}_1$.

k	$\mu(E_\mu)$	$\sigma(E_\mu)$	$\mu(E_\infty)$	$\sigma(E_\infty)$
2	0.358	0.0632	1.2131	0.1917
4	0.0888	0.0092	0.3945	0.0695
6	0.0365	0.0027	0.1702	0.0273
8	0.0193	0.001	0.0907	0.0149

Naturally, as we add more flats into our model, the approximation error decreases.

4.1.2 Sampling

One issue we hoped to address in the design of MLBF was sensitivity of the algorithm to non-uniform sampling. To test this aspect of the algorithm's performance, we experimented with dataset $\mathcal{D}_2$ as described in Table 4.1.2.

Table 4.3: Description of dataset $\mathcal{D}_2$.

Dataset Name:	$\mathcal{D}_2$
Manifold Description:	Curve in $\mathbb{R}^2$
Sampling Method:	x equispaced on $[-2, 2]$, spacing $h = 0.03$; $y = \frac{1}{1+16x^2}$
Sample Count:	$N = 134$

The key property of this dataset is that the sampling rate at the spike in the center is much lower than the rest of the curve. The output of MLBF on $\mathbf{D}_2$ with $k = 7$ is depicted in Figure 4.2.

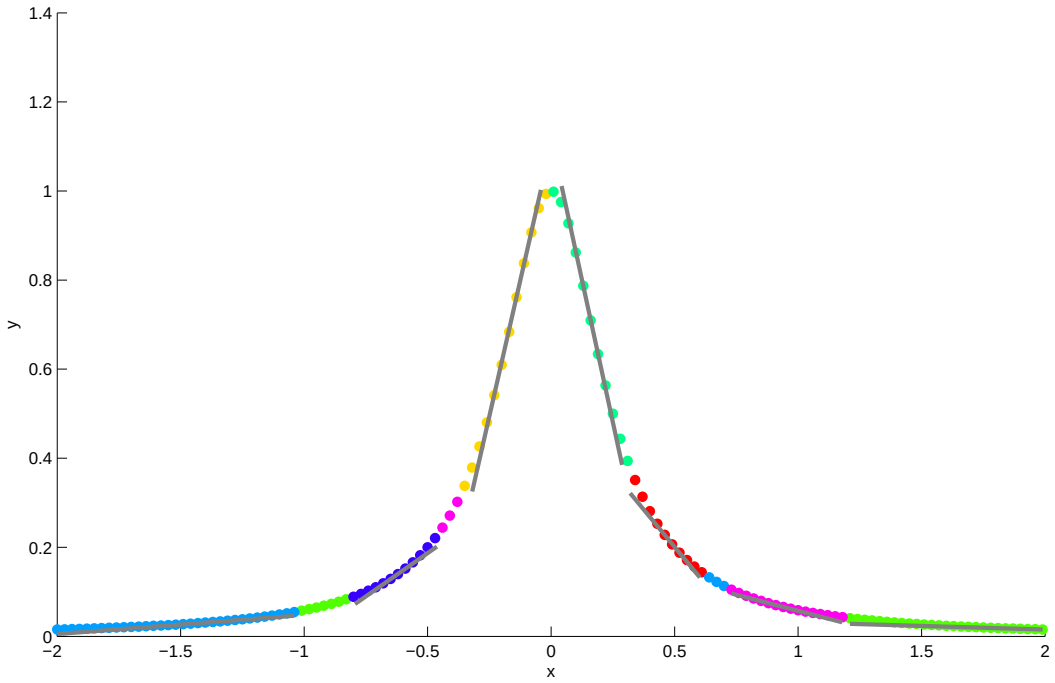

Figure 4.2: Output of MLBF on $\mathcal{D}_2$ with $k = 7$.

Indeed, MLBF was able to construct appropriate flats even in regions with fewer samples. However, an unexpected result presented itself. The intention of MLBF was to approximate a given point with an affine subspace that is similar to the tangent space around that point. However, this notion of locality is not enforced in the current version of the algorithm. This may be seen by observing that several points on the left side of the curve are being represented by a flat computed on the right side. Although the tangent space for the light green points on the left is more similar to that of the the adjacent flats, the points happen to lie closer to the flat formed using a cluster of points from the right side.

4.1.3 Noise

Next, we sought to test the performance of MLBF in the presence of noise. To do so, we modified dataset $\mathcal{D}_1$ by adding noise from a normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 0.1$ to the y component. We refer to this dataset as $\mathcal{D}_3$. The output of MLBF with $k = 7$ is shown in Figure 4.3.

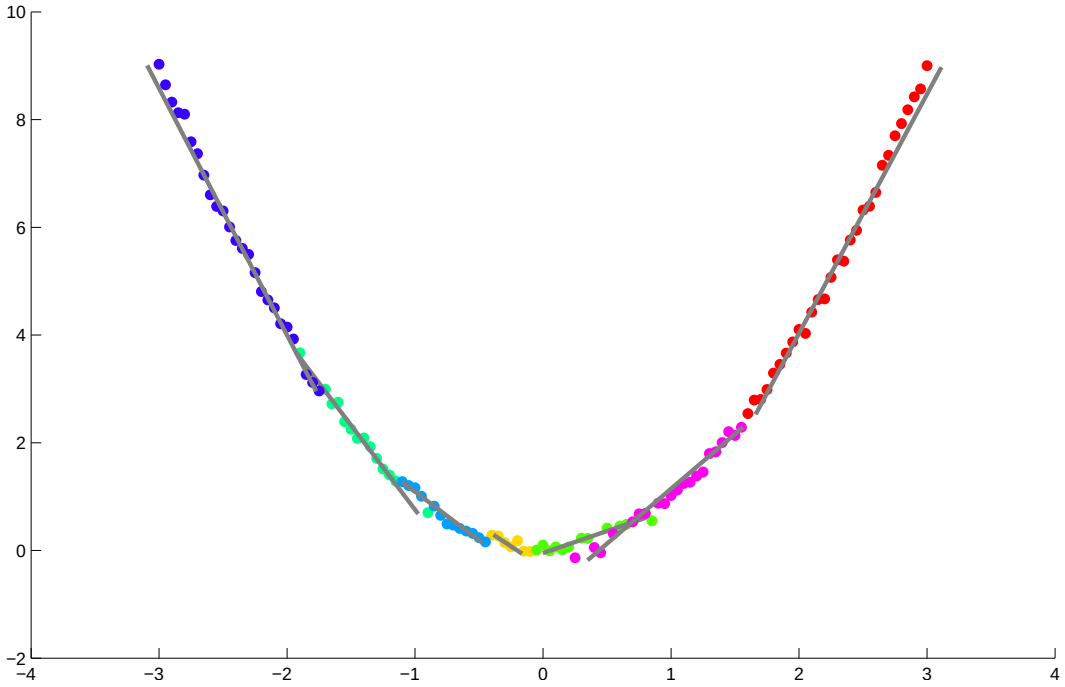

Figure 4.3: Output of MLBF on dataset $\mathcal{D}_3$.

The resulting error computations are shown in Table 4.4.

Table 4.4: Error analysis for MLBF on dataset $\mathcal{D}_3$.

k	$\mu(E_\mu)$	$\sigma(E_\mu)$	$\mu(E_\infty)$	$\sigma(E_\infty)$
7	0.0339	0.0027	0.1691	0.0427

In the presence of noise, the approximation error on $\mathcal{D}_3$ with $k = 7$ is similar to that of $\mathcal{D}_1$ with $k = 6$. However, the locality issue again presents itself. The green cluster on the left has points that extend into the adjacent blue clusters. Similar behavior appears on the right side between the green and pink clusters. The issue of locality is discussed further in Chapter 5.

4.1.4 MLBF in Higher Dimension

Next, we demonstrate the application of MLBF on datasets in $\mathbb{R}^3$. The first dataset used is described in Table 4.5.

Table 4.5: Description of dataset $\mathcal{D}_4$.

Dataset Name:	$\mathcal{D}_4$
Manifold Description:	Surface in $\mathbb{R}^3$, two parallel plates connected by a curved sheet
Sampling Method:	Plates - x, y equispaced on [0,5]; z constant
	Sheet - t equispaced; x uniform on [0,5]; $y = \cos \pi t$, $x = \sin \pi t$
Sample Count:	$N = 1200$
Flat Count:	$k = 2, 4, 6, 8$

We expect that two large flats will be used to approximate the top and bottom plates, and that the remaining flats will cluster along the curved sheet. This behavior is confirmed in Figure 4.4 which displays the output of MLBF on dataset $\mathcal{D}_4$. An error analysis is shown in Table 4.6.

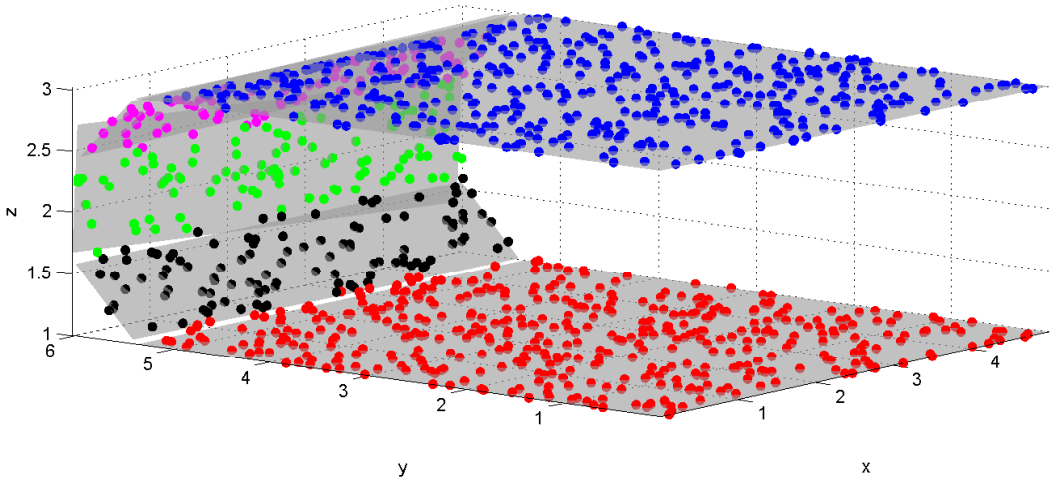

Figure 4.4: Demonstration of MLBF applied to dataset $\mathcal{D}_4$.

Table 4.6: Error analysis for MLBF on dataset $\mathcal{D}_4$.

k	$\mu(E_\mu)$	$\sigma(E_\mu)$	$\mu(E_\infty)$	$\sigma(E_\infty)$
5	0.0091	0.0010	0.1203	0.0215

Next, we applied MLBF to the 2-dimensional analog of dataset $\mathcal{D}_1$, described in Table 4.7.

Table 4.7: Description of dataset $\mathcal{D}_5$.

Dataset Name:	$\mathcal{D}_5$
Manifold Description:	Paraboloid in in $\mathbb{R}^3$
Sampling Method:	Plates - x, y equispaced on [-1,1] with $h = 0.1$; $z = x^2 + y^2$
Sample Count:	$N = 442$
Flat Count:	$k = 3, 5, 7, 9$

The output of MLBF on dataset $\mathcal{D}_5$ is shown for the $k = 5$ case in Figure 4.5. An error analysis may be found in Table 4.8.

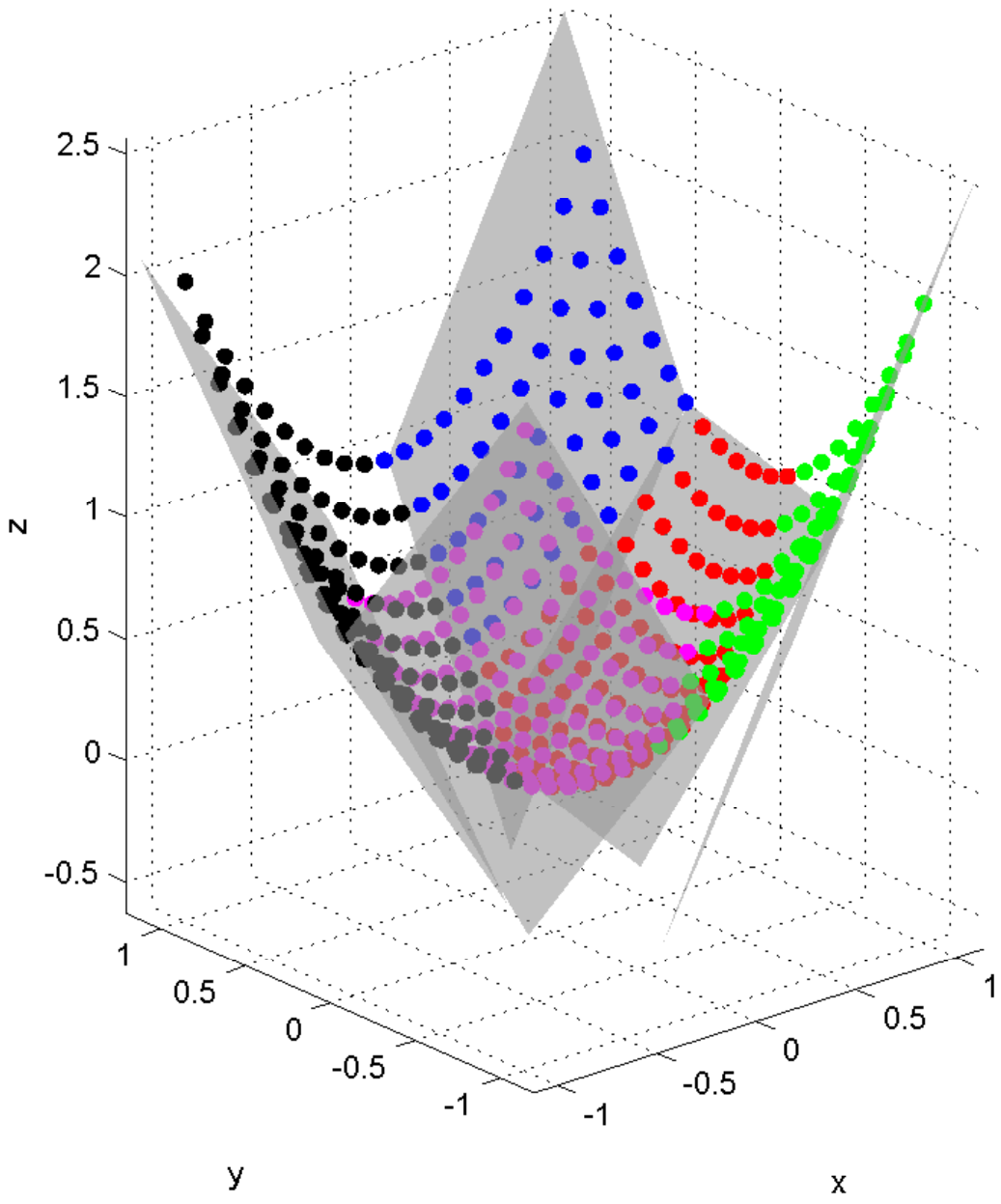

Figure 4.5: Demonstration of MLBF applied to dataset $\mathcal{D}_5$ with $k = 5$.

Table 4.8: Error analysis for MLBF on dataset $\mathcal{D}_5$.

k	$\mu(E_\mu)$	$\sigma(E_\mu)$	$\mu(E_\infty)$	$\sigma(E_\infty)$
3	0.2095	0.0189	0.9256	0.2163
5	0.1059	0.0078	0.3918	0.0643
7	0.0697	0.0037	0.2841	0.0516
9	0.0505	0.0022	0.1998	0.0265

The decrease in error is significant as k ranges from 3 to 7, however we notice diminishing returns after $k = 7$.

4.1.5 Scale

Next, we experimentally verify that the method proposed for choosing the optimal neighborhood size in MLBF works as expected. We expect that in regions of low (high) curvature, a larger (smaller) neighborhood may be used to compute the tangent plane around a given point. To show that the proposed method indeed does this, we performed an experiment on dataset $\mathcal{D}_4$ where the neighborhood size is computed for every point[1] . The result is shown in Figure 4.6.

[1] In the actual execution of the MLBF algorithm, we do not in fact compute the optimal neighborhood size around each point. The computation is done for every point in this experiment to test the viability of the method

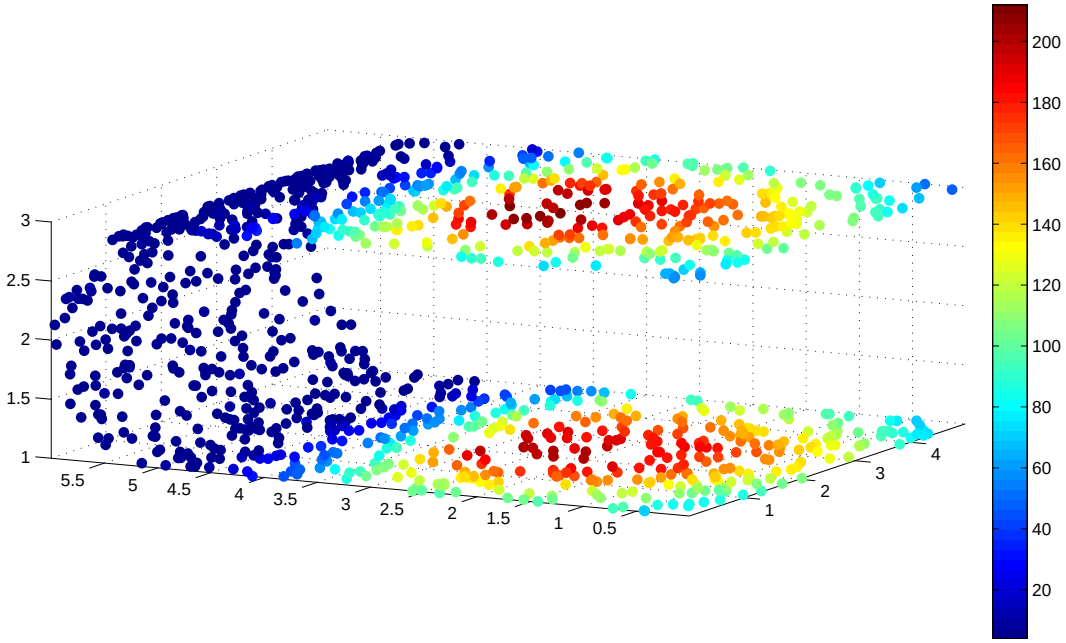

Figure 4.6: Number of points in the optimal neighborhood centered at each point.

First, notice that the region with curvature will utilize far fewer points than the flat region. Furthermore, it is interesting to note that the middle of the flat region uses more points than the edges. This is because neighborhoods centered around interior points with a given radius contain more points than ones centered around points along the boundary. At the edges, fewer points may be included before points from the opposing plate enter the neighborhood. This experimentally verifies that the method of determining the optimal neighborhood works as expected.

4.2 Experiments with MKSVD

In this section, we apply MKSVD to dataset $\mathcal{D}_4$. We will examine two sets of initial conditions. The first set of initial conditions is produced using k-means clustering. The initial clustering and output of MKSVD is shown in Figure 4.7.

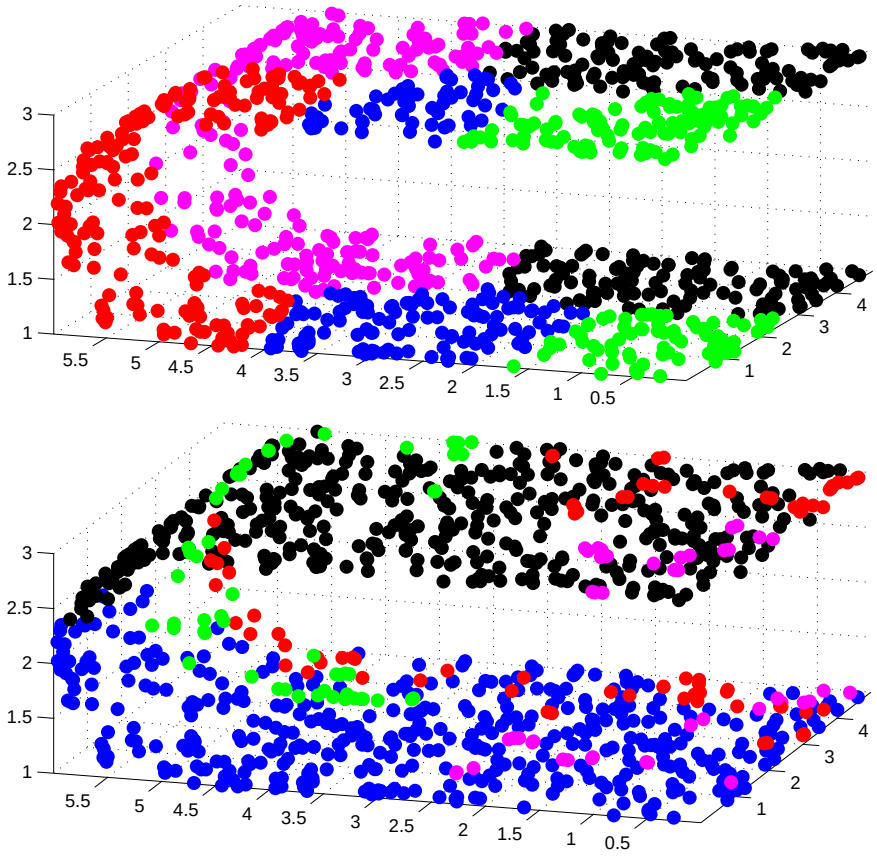

Figure 4.7: Initial clustering (top) and output (bottom) for MKSVD applied to dataset $\mathcal{D}_4$.

This experiment demonstrates extreme sensitivity to initial conditions. Since k-means did not yield anything close to the desired result, the output turned out to be a scattering of the input data. To observe the performance of MKSVD with a more ideal initial clustering of the input data, we performed the experiment depicted in Figure 4.8.

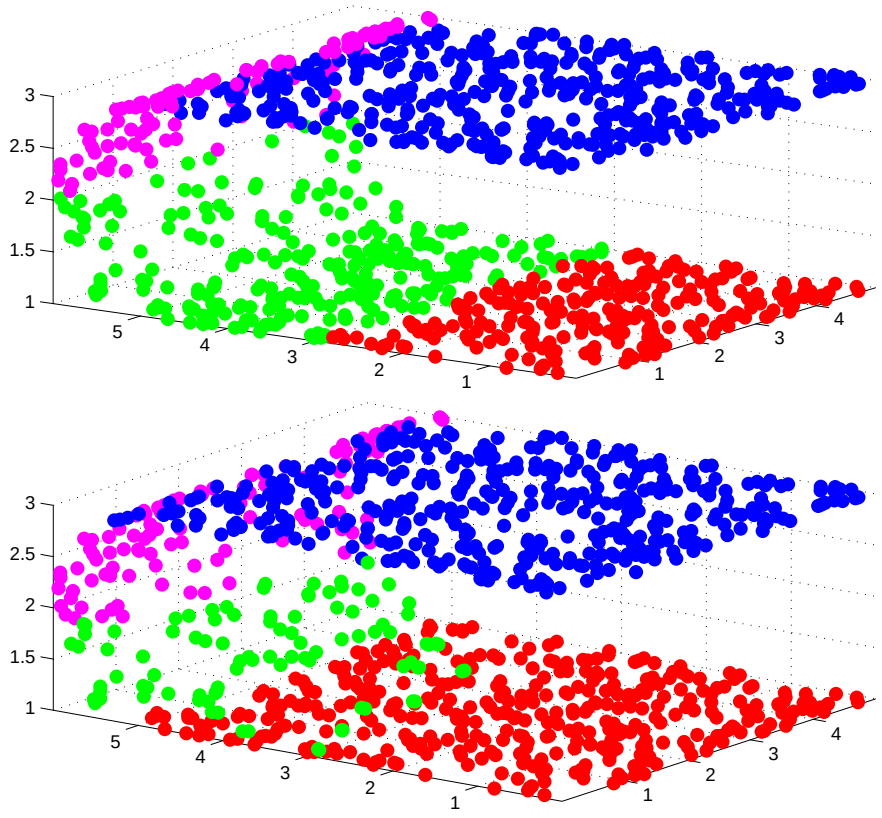

Figure 4.8: Initial clustering (top) and output (bottom) for MKSVD applied to dataset $\mathcal{D}_4$.

The output in this case is much closer to the desired output. However, the boundary between clusters is unnatural. For example, notice that the green cluster on the right hand side of Figure 4.8 extends into the red cluster. Because of these two limitations of MKSVD, we do not perform further experiments. Ideas for future directions for MKSVD are discussed in Section 5.

Chapter 5

Discussion

In this chapter we discuss several aspects of the two algorithms proposed. MKSVD is discussed briefly in Section 5.1, and the remainder of the chapter covers MLBF.

5.1 MKSVD

As we saw in the experiments from Section 4.2, two limitations of MKSVD prevented it from being a viable solution to our problem. In this section, we briefly discuss the reasons for these limitations and propose potential workarounds.

The first limitation of MKSVD is its sensitivity to initial conditions. As we saw in the experiments performed, there are cases where the subspaces constructed using the initial clustering of points are extremely far from the ideal ones. For example, we would expect that the parallel plates from dataset $\mathcal{D}_4$ would each constitute a cluster. However, the initial conditions produced using k-means clustering groups points from the top and bottom plates into the same cluster. The resulting subspace was thus a poor approximation for points from either side of the object. Subsequent reassignment of points results in a scattering of the data, and eventually a local minimum of global approximation error is reached. As we saw, the issue subsides when a more reasonable initial clustering of the data is given. If one wished to pursue MKSVD further, we propose using the output of MLBF as an initial condition for MKSVD.

Next is the issue of unnatural boundaries between clusters. To remedy this, we propose

applying a penalty during the reassignment of points which discourages reassignments which will increase the length of the boundary. Obviously this requires a method for computing the length of the boundary between clusters. We do not investigate this problem here.

This concludes our discussion of MKSVD. The remaining part of this chapter focus on MLBF.

5.2 MLBF

5.2.1 Parameters

In this section, we discuss the various input parameters of MLBF. The number of samples, N, is a function of the dimensionality and curvature of the underlying manifold, as well as noise. A classical result from numerical analysis tells us that using more samples to solve an over-determined system results in less error due to noise. However, in many applications, data acquisition may be very expensive, so it is desirable to use as few samples as possible. In other applications, there may be an overabundance of data, in which case one must use only a subset of the available data. The problem of choosing the optimal number of samples, as well as methodology to ensure that samples are taken in such a way that the structure of the underlying manifold is captured is very complex, and is not studied in detail here. Additional information in this area may be found in [15].

The number of flats, k, used to approximate a dataset is a function of curvature, scale, and the desired fidelity of the resulting approximation. We saw in our experiments that using more flats resulted in better approximation error. However, we also observed that at some point, increasing the number of flats offers diminishing returns in terms of the quality of approximation. Furthermore, using more flats than necessary results in an over-complicated model which lacks the benefits of a more simple one. This suggests that there exists a loose notion of the "optimal" number of flats to approximate a given manifold. In the version of the algorithm proposed here, k is an input parameter that must be chosen manually. A

method to automate this process is proposed in Section 5.2.4.

The parameters S and T are used when we determine the optimal neighborhood size for the construction of a flat. As described in Chapter 3, the algorithm begins by trying a neighborhood of size S, and increases the number of points in the neighborhood by T each iteration until the optimal neighborhood is found. The dimensionality and curvature of the underlying manifold, as well as scale are important factors in the choice of these parameters. In our experiments, we use $S = d + 1$, because this is the minimum number of points required to recover a subspace of dimension d in the absence of noise. The choice of T is highly dependent on scale. If a manifold is sampled very densely, it may be computationally efficient to choose a larger value of T.

5.2.2 Unions of Manifolds

In many applications, it makes sense to model data as coming from a union of manifolds, each of different dimensionality. For example, suppose we wish to model the sound produced by musical instruments in a controlled setting. The work described in [7] and [8] suggests that a manifold model might be appropriate. The region of the manifold representing a brass instrument, whose sound may be changed in many ways, such as pressing keys or modifying the length of tuning slides, would likely have many more degrees of freedom than that of a tuning fork, whose sound would depend primarily on the force with which the object was struck.

If we had a map available which would indicate the dimensionality of the manifold in different regions, we could partition the dataset according to dimension, run MLBF on the individually on subsets of equal dimension, and combine the resulting model. Techniques for obtaining such a map were not investigated in this work. We refer interest readers to [13], [6], [19] as a starting point.

5.2.3 Applications

In this section, we discuss various applications for the algorithms presented in this thesis. One straightforward application is data denoising. This is easily done by projecting a measured point on to its corresponding flat. Similarly, the model produced by MLBF may be used for interpolation. In this way, we can recover the value of pieces of the manifold for which we do not have samples.

The model produced by MLBF also allows for data compression. The immediate benefit is that each raw measurement which consists of D values may be represented as a linear combination of basis vectors, requiring only d coefficients. Moreover, since each flat represents a region of the manifold, we have the benefit of encoding entire regions using only d basis vectors and one additional vector for the affine offset.

5.2.4 Limitations and Future Work

We have seen several limitations in our discussion of MLBF. In this section, we summarize these limitations, and suggest avenues for future research.

The first issue we will address is the lack of locality in the approximation produced by MLBF[1] . The problem arises from the fact that no boundaries are constructed for each flat. Points are assigned by minimizing their distance to a flat of infinite extent. To correct this, one might consider the following approach: first assign points as we did before. Then, in a second pass, compute distance of each point to the nearest several flats. Rather than simply choosing the nearest flat, a penalty may be introduced which discourages the assignment of a point to a flat in which the majority of points are far away from the point being assigned.

Next, the current version of the algorithm requires manual input of the parameter k. In order to automate the selection of k, one may consider running multiple iterations of MLBF, each time increasing k. Information such as an upper bound on approximation error

[1] See Section 4.1.2 for an example of this.

given by the user and relative increase in approximation quality between iterations might be used to find a suitable balance between approximation quality and model simplicity.

MLBF is also prone to becoming stuck in local minima. Consider the case where there are no points around which to construct new flats. One could construct an example in which a gain in global approximation error is possible only by exchanging two or more flats from $\hat{L}$ simultaneously. Since MLBF only allows one flat in $\hat{L}$ to be exchanged at a time, the algorithm will get stuck, even though a more optimal configuration of flats exists. A straightforward approach to address this problem is to run MLBF several times and use the best solution. Alternatively, one could investigate methods for combining the results of separate executions of MLBF to construct a solution that is superior to any individual solution.

Another limitation of MLBF is the lack of any theoretical proof or guarantee on convergence rates. It is easy to see that the global approximation error between iterations decreases[2] . However, it does not do so strictly. Furthermore, MLBF has no guarantee on the proximity of the resulting output to the optimal arrangement of flats.

Lastly, we have not offered any methods for optimizing MLBF for large datasets. An obvious technique for dealing with large datasets is to use only a subset of the data available for the execution of MLBF. One could then reincorporate the remaining data into the model at the end to evaluate the quality of approximation. More sophisticated techniques might be able to intelligently choose the subset of the data which MLBF uses. Furthermore, one could investigate randomizing portions of the algorithm. For example, rather than computing the distance of every single point to the nearest flat to determine the approximation error for a given iteration, we could perform the computation for only a randomly chosen subset of points.

[2] This is proved easily by noting that an exchange of flats occurs only if global approximation error results.

Chapter 6

Conclusion and Future Directions

Locally linear approximations provide an effective method to produce parameterizations of manifold-valued data. To our knowledge, no one has proposed an algorithm that attempts to solve this problem. We have developed two algorithms which seek to generate such parameterizations. In particular, Manifold Local Best-fit Flats (MLBF) is a good first step towards a complete solution. Through a series of experiments, we have shown the strengths of MLBF and discovered several avenues for future work.

There is an essential trade-off between approximation quality and model complexity. While using more affine subspaces to approximate a curved manifold improves the approximation quality, experiments show that beyond a certain point, this improvement becomes insignificant.

There are a variety of applications in which MLBF may be utilized. For example, we demonstrated experimentally that MLBF may be used to cluster datasets for which traditional algorithms such as k-means fail. Other applications include data denosing, interpolation, and compression.

Several limitations of the current state of our work suggest several directions for future work. The current implementation of MLBF does not enforce locality in the representation of data. In other words, the flat used to represent a given point may not be similar to the tangent space around that point. If this constraint is desired, modifications must be made to the algorithm. Furthermore, no theoretical proofs or guarantees are provided in this work.

Although MKSVD did not prove successful with its current implementation, it may be worthwhile to pursue this method further. We suspect that in some cases, a better approximation may be obtained by using a best-fit subspace using all points in a cluster, rather than using an approximation of the tangent space around a single point. One may consider a combination of MKSVD and MLBF to develop this idea.

Bibliography

[1] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. Signal Processing, IEEE Transactions on, 54(11):4311 –4322, nov. 2006.

[2] K. Atkinson. An Introduction to Numerical Analysis. Wiley, New Jersey, 1989.

[3] Alfred Bruckstien, David L. Donoho, and Michael Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. SIAM Review, 51(1):34–81, 2009.

[4] G. Chen and G. Lerman. Spectral curvature clustering (scc). International Journal of Computer Vision, 83(3):317–330, 2009.

[5] G. David and S. Semmes. Singular integrals and rectifiable sets in rn. Asterisque, 193:1–145, 1991.

[6] Matthias Hein and Jean-Yves Audibert. Intrinsic dimensionality estimation of submanifolds in rd. Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, 2005.

[7] A. Jansen and P. Niyogi. A geometric perspective on speech sounds. 2005.

[8] A. Jansen and P. Niyogi. Intrinsic fourier analysis on the manifold of speech sounds. 2005.

[9] P. Jones. Rectifiable sets and the traveling salesman problem. Invent Math, 102(1):1–15, 1990.

[10] Daniel N. Kaslovsky. Geometric Sparsity in High Dimension. PhD thesis, University of Colorado at Boulder, 2012.

[11] Z.H. Khan and I.Y.-H. Gu. Tracking visual and infrared objects using joint riemannian manifold appearance and affine shape modeling. In Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, pages 1847 –1854, nov. 2011.

[12] G. Lerman. Quantifying curvelike structures of measures by using l2 jones quantities. Communications on Pure and Applied Mathematics, 56(9):1294–1365, 2003.

[13] Anna V. Little, Y. Jung, and M. Maggioni. Multiscale estimation of instrinsic dimensionality of data sets. Proc. A.A.A.I, 2009.

[14] Peter Olver and Cheri Shakiban. Applied Linear Algebra. Prentice Hall, New Jersey, 2005.

[15] Emil Saucan, Eli Appleboim, and Yehoshua Zeevi. Sampling and reconstruction of surfaces and higher dimensional manifolds. Journal of Mathematical Imaging and Visualization, 30:105–123, 2008.

[16] M Tipping and Bishop C. Mixtures of probabilisitic principle component analysers. Neural Computation, 11(2):443–482, 1999.

[17] P Tseng. Nearest q-flat to m points. Journal of Optimization Theory and Applications, 105:249–252, 2000.

[18] R. Vidal, Ma. Y, and S. Sastry. Generalized principal component analysis (gpca). IEEE Transactions on Pattern Analysis and Machine Learning, 27(12), 2005.

[19] Xiaohui Wang and J.S. Marron. A scale-based approach to finding effective dimensionality in manifold learning. Electronic Journal of Statistics, 2:127–148, 2008.

[20] Qilong Zhang, Richard Souvenir, and Robert Pless. Segmentation informed by manifold learning. In EMMCVPR, number 0297 in 3757, pages 398–413. LNCS, 2005.

[21] T. Zhang and Szlam A. Median k-flats for hybrid linear modeling with many outliers. Computer Vision Workshops, IEEE 12th International Conference on Computer Vision, pages 234–241.

[22] Teng Zhang, A. Szlam, Yi Wang, and G. Lerman. Randomized hybrid linear modeling by local best-fit flats. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 1927 –1934, june 2010.