Spherelets

Stat 185 Term Paper

Caleb Ren

December 14, 2019

Contents

1 Introduction

Data that exists in a high-dimensional ambient space may instead be considered to lie near a lower dimensional manifold (e.g. a circle in $\mathbb{R}^2$ that lies ambiently in $\mathbb{R}^3$ space). Many techniques are focused on reducing the ambient space to one closer to the intrinsic space such as clustering (Duda, Stork, and Hart 2000), data compression (Hurtado 2012), and predictive modelling (J. A. Lee and Verleysen 2008). Many of these techniques that attempt to approximate manifolds embedded in a lower dimensional space are locally linear, use multiscale dictionaries, and as such struggle with areas with high Gaussian curvature. Additionally, many techniques that approximate local manifolds do not perform well with out-of-sample error as they do not reveal much information on the traits of the lower dimension manifold.

A simple alternative that is able to handle curvature as well as out-of-sample error is to use pieces of (hyper)spheres to locally approximate the unknown subspace (Didong Li and Dunson 2019). This method was first proposed by Li and Dunson, who termed the technique spherical PCA (SPCA) (Li and Dunson 2017), or *spherelets* for short. Whereas principal component analysis (PCA) is an eigenvalue/eigenvector problem from an inherently *linear* dimension reduction problem, spherelets rely on projection mappings to the surfaces of a hypersphere as an underlying mechanism. As such, in simple cases of SPCA, closed form analytic solutions exist and can be seen as a generalization of PCA that is able to incorporate degrees of curvature. Spherical PCA in general does not have a closed-form solution due to the myriad ways in which a particular dataset can be partitioned and fit with sub-manifolds. Various algorithms exist to best partition the original dataset such as cover trees (Beygelzimer, Kakade, and Langford 2006), METIS (Karypis and Kumar 1998), and iterated PCA (Szlam 2009). The type of partitioning done is context-dependent and multiple types of subsetting can be done to cross-validated to reduce test MSE. As examples, we will examine two types of partitioning: naively—by splitting the dataset into equal-sized clusters—and via partioning by iterated PCA (Didong Li and Dunson 2019).

Data that is represented in higher dimension $\mathbb{R}^D$ can be projected down to a manifold in $\mathbb{R}^d$ consisting of a set of spherelets in $\mathbb{R}^d$. Ultimately, the algorithm partitions the dataset into k subsets, each of which is fit with a submanifold $M_k \in \mathbb{R}^d$ with a corresponding projection map $\Psi_k : \mathbb{R}^D \to \mathbb{R}^d$. Given a dataset $\mathbf{X} \in \mathbb{R}^D$, the spherical PCA algorithm returns manifold M and the projection map Ψ, which are the manifold and mapping from the data to the approximate manifold in $\mathbb{R}^d$, respectively.

Manifold approximation for model-building is one useful application of spherelets but the technique can be used for other purposes. Spherical PCA can also be used to denoise manifolds and generate visualizations of data in higher dimensions. Spherelets have been shown to outperform competiting denoising algorithms like Gaussian Blurring Mean Shift (GBMS), Manifold Blurring Mean Shift (MBMS), and Local Tangent Projection (LTP). Visualization techniques that preserve local geometry like Isomap, Locally Linear

Embedding (LLE), and t-distributed Stochastic Neighborhood Embedding (t-SNE) can also be modified to incorporate spherical geometry using the same projection to spherical affine subspace, as in the original spherelets paper (Didong Li and Dunson 2019).

In spherical PCA, a point $\vec{y}_i$ is projected down to a sphere $S_V(c, r)$, where c is the center of the sphere, r is the radius, and V specifies the subspace on which the sphere lies. The optimal estimates for the sphere $S_{\hat{V}}(\hat{c}, \hat{r})$ are given by:

$$\hat{V} = (\vec{v}_1, \ldots, \vec{v}_{d+1})$$
$$\hat{c} = -\frac{\vec{\eta}^*}{2}$$
$$\hat{r} = \frac{1}{N}\sum_{k=1}^{N} \|\vec{z}_i - \hat{c}\|$$

Where $\vec{v}_i$ is the ith eigenvector of the covariance matrix times $\mathbf{\Sigma}$, $\vec{z}_i$ is the the projected data down to the subspace and $\vec{\eta}^*$ is a vector based on the entire projected dataset $\mathbf{Z}$. After an introduction into notation of spherical PCA, I will introduce relevant theory, including the Main Theorem that undergirds spherical PCA in the following section, as well as assumptions made. I will discuss strengths and weaknesses of the algorithm in Section 3, and then provide numerical examples demonstrating areas in which spherelets perform well and where the algorithm struggles in Section 4.

2 Method

2.1 Theory

2.1.1 Covering Number

The Main Theorem underlying spherical PCA asserts that the covering number of spherelets is lower than that of an analogous PCA method using hyperplanes. Given a dictionary of basis functions $\mathcal{B}$, the Main Theorem shows a that the number of individual "pieces" of spheres to approximate a manifold within error threshold $\epsilon > 0$ is lower than that of a purely hyperplane method (Didong Li and Dunson 2019). If M is a d-dimensional, compact, C^3-smooth Riemannian manifold, the covering number $\mathcal{N}_{\mathcal{B}}(\epsilon, M)$ is defined as:

$$N_{\mathcal{B}}(\epsilon, M) := \inf_N \left\{ N : \exists \{C_k, \Psi_k, B_k\}_{k=1}^K \, s.t. \, \|x - \Psi(x)\| \leq \epsilon, \, \forall x \in M \right\}$$

Here, C_k defines the kth partition of ambient space $\mathbb{R}^D$, B_k is a local basis within the dictionary of basis functions $\mathcal{B}$. We have previously defined Ψ to be the global projection,

so Ψ_k is the local projection within the kth partition and $\Psi = \sum_{k=1}^{K} 1_{x \in C_k} \Psi_k$ defines the global projection as a combination of local projections.

This definition of covering number holds that there exists some number N such that we can fit a sphere with error lower than threshold ϵ. In the context of spherical PCA, we have two general choices of $\mathcal{B}$: $\mathcal{S}$—the dictionary of basis functions that use a spherical affine space—and $\mathcal{H}$—the dictionary of basis functions that use hyperplanes. Note that if we allow the construction of spheres with infinite radius, we obtain manifolds with $\kappa = 0$, which are hyperplanes! Thus, $\mathcal{H} \subset \mathcal{S}$. As such, we arrive at the inequality:

$$N_{\mathcal{S}}(\epsilon, M) \leq N_{\mathcal{H}}(\epsilon, M)$$

Spherical PCA asserts that in the worst case with totally linear manifolds, the algorithm will perform as poorly as local PCA that uses hyperplanes instead of sections of spheres as its dictionary of basis functions.

2.1.2 Main Theorem

We can also achieve a tighter upper bound than the above inequality. Given a manifold M that satisfies the conditions listed above (d-dimensional with $d << D$, C^3-smooth, compact, Riemannian), we define additional notation:

- $K(p, \vec{v})$ is a function that assigns a curvature of the geodesic on M starting at arbitrary point p with direction $\vec{v}$,
- K^* is the maximum curvature defined above with $K^* := \sup_{(p,\vec{v}) \in UTM} K(p, \vec{v}) < \infty$,
- Similarly, if $T(p, \vec{v})$ defines the rate of change of curvature, then $T^* := \sup_{(p,\vec{v}) \in UTM} T(p, \vec{v})$ defines the maximum rate of change in curvature,
- UTM as encountered above defines the unit sphere bundle over M and UT_pM defines the unit sphere bundle centered at point $p \in M$,
- F_ϵ is the set of ϵ-spherical points on M with difference in maximum and minimum curvature under a threshold level parametrized by ϵ:

$$F_\epsilon := \left\{ p \in M : \sup_{\vec{v} \in UT_pM} K(p, \vec{v}) - \inf_{\vec{v} \in UT_pM} K(p, \vec{v}) \leq \left(\frac{2\epsilon}{K^*} \right)^{1/2} \right\}$$

- $B(p, \epsilon)$ is a ball with radius ϵ and center p.

That is to say, the spherical submanifold M_ϵ of M for a given error ϵ is the union of all the balls centered at points in F_ϵ, or

$$M_\epsilon := \bigcup_{p \in F_\epsilon} B\left(p, \left(\frac{6\epsilon}{3+T}\right)^{1/3}\right)$$

We have therefore defined a way to define a spherical submanifold M_ϵ of base manifold M given an error level ϵ. We define V_ϵ as the volume contained within this spherical submanifold M_ϵ.

We have now defined the notation necessary for the main theorem, which contrasts covering numbers for hyperplanes and sections of spheres.

Main Theorem. (Didong Li and Dunson 2019) For any $\epsilon > 0$ and compact C^3 d-dimensional Riemannian manifold M,

$$N_{\mathcal{H}} \lesssim V\left(\frac{2\epsilon}{K^*}\right)^{-d/2}$$
$$N_{\mathcal{S}} \lesssim V_\epsilon\left(\frac{6\epsilon}{3+T^*}\right)^{-d/3} + (V - V_\epsilon)\left(\frac{2\epsilon}{K^*}\right)^{-d/2}$$

A key result from the Main Theorem is that the covering number of spherelets has a lower upper bound when the difference between V and V_ϵ is small. In the case when $V - V_\epsilon \to 0$, the second term in the bound for $N_{\mathcal{S}}$ shrinks to 0 and the first term dominates, which grows at $O(c^{-n/3})$, which outperforms the $O(c^{-n/2})$ that $N_{\mathcal{H}}$ grows at.

In the case when $V_\epsilon \to 0$ (when the degree of curvature is low so the spherical submanifold is unable to cover much of the underlying manifold), spherical PCA converges to hyperplane PCA.

According to Li and Dunson, the bounds to the Main Theorem are tight, implying that spherelets often require many fewer pieces than locally linear dictionaries and approximate M for a given error level ϵ. Spherelets provide an exceptionally good approximation of M when there the curvature across the manifold stays in a relatively narrow range, as these would require many more parameters in locally linear PCA to approximate. As such, spherelets provide a more robust way of fitting datasets that have intrinsic curvature in a lower d-dimensional manifold compared to hyperplane PCA.

2.2 Spherical PCA

2.2.1 Notation

Assume we have the $N \times D$ data matrix $\mathbf{X}$, with N observations and D variable where $x_{i,j}$ represents the ith observation of the jth variable:

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & \dots & x_{1,D} \\ \vdots & \ddots & \vdots \\ x_{i,1} & \dots & x_{i,D} \\ \vdots & \ddots & \vdots \\ x_{N,1} & \dots & x_{N,D} \end{bmatrix}$$

As in linear PCA, a more succinct way to represent this matrix is to write:

$$\mathbf{X} = \begin{bmatrix} \vec{x}_1^T \\ \vdots \\ \vec{x}_i^T \\ \vdots \\ \vec{x}_N^T \end{bmatrix}$$

We will once again treat $\vec{x}_1, \dots, \vec{x}_N$ as i.i.d. samples of a random vector in $\mathbb{R}^D$ from the same underlying distribution F for which $E_{x \sim F}\|x\|^2 < \infty$ to assure that the mean and covariance of the data matrix are well-deefined. We use:

$$\overline{x} = \frac{1}{N} \sum_{i=1}^{N} \vec{x}_i$$

to denote the sample mean of $\vec{x}$. To denote the covariance operator, we use:

$$\mathbf{\Sigma}_x = \frac{1}{N} \sum_{i=1}^{N} (\vec{x}_i - \overline{x})(\vec{x}_i - \overline{x})^T = \frac{1}{N} \left(\sum_{i=1}^{N} \vec{x}_i \vec{x}_i^T \right) - \overline{x}\overline{x}^T = \frac{1}{N} \mathbf{X}^T \mathbf{X} - \overline{x}\overline{x}^T$$

This covariance operator will come into use later on. Additional notation includes:

- 1_N is the column vector of all ones with length N (i.e. $1_N \in \mathbb{R}^N$)
- $\|\vec{z}\|$ denotes the Euclidean norm (i.e. for $\vec{z} \in \mathbb{R}^d$, $\|\vec{z}\| = \left(\sum_{i=1}^{d} z_i^2 \right)^{1/2} = \sqrt{\vec{z}^T \vec{z}}$)
- $\Psi : \mathbb{R}^D \to \mathbb{R}^{d+1}$ is a projection map from space $\mathbb{R}^D$ to $\mathbb{R}^{d+1}$. Note: Ψ maps from $\mathbb{R}^D$ to $\mathbb{R}^{d+1}$ rather than $\mathbb{R}^d$ because the projected points lie on an affine subspace with one fewer degree of freedom than the manifold. One example is fitting a curve in $\mathbf{R}^2$ with sections of 2-dimensional circles; while circles are 2D, the projected points only live along the edge of the circle. As such, we include on extra dimension in our projection map to account for the extra degree of freedom.
- $d^2(\cdot, \cdot)$ is the distance operator between two points. Note that $d^2(x, y)$ is equivalent to $\sqrt{\|x - y\|}$.

2.2.2 Assumptions

There are three main assumptions made by Li and Dunson:

- **Distributional Assumption**. The data matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ consists of a transformation $V^* \Lambda^{*\frac{1}{2}} \mathbf{Z}$ where $\mathbf{Z} \in \mathbb{R}^{N \times D}$ is a data matrix consisting of i.i.d z_i non-degenerate random variables.
- **Moment Assumption**. For the underlying random variables z_i, $E(z_i) = 0$, $E(z_i^2) = 1$, and $E(z_i^6) < \infty$. We can choose an affine transformation V^* such that the underlying random variables satisfy the first two moment assumptions. A similar assumption is considered in Lee et al. that uses bounded fourth moments instead to prove convergence in high dimensional PCA (S. Lee, Zou, and Wright 2010).
- **Spike Population Model Assumption**. If $\lambda_1, \lambda_2, \ldots, \lambda_D$ are the ordered eigenvalues of Λ^* as similar to a PCA setting, then there exists an integer $m > d$ such that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m \geq lambda_{m+1} = \cdots = \lambda_D = 1$.

The consequence of these assumptions is that we have a theoretical guarantee that the estimate projection $\hat{\Psi}$ converges in probability to Ψ^*. In other words:

$$\lim_{n \to \infty} P(\sup_{x \in M} \|\hat{\Psi}(x) - \Psi^*(x)\| > \epsilon) = 0, \, \forall \epsilon < 0$$

2.2.3 Estimation

Given a set of data $\vec{x}_1, \ldots, \vec{x}_N \in \mathbb{R}^D$ organized into data matrix $\mathbf{X}$, spherical PCA estimates the best approximating sphere $S_V(c, r)$, where c is the center, r is the radius, and $V \in \mathbb{R}^{(d+1) \times (d+1)}$ is the $(d+1)$th dimensional affine subspace the sphere lives on.

For observation $\vec{x}_i$, the closest point $\vec{y}_i$ lying on the sphere $S_V(c, r)$ is the point that minimizes Euclidean distance $d^2(\vec{x}_i, \vec{y}_i)$ between $\vec{x}$ and $\vec{y}$.

Continuing the convention that Ψ is an orthogonal projection from R^D to the affine subspace given by c + V, then Ψ is given by:

$$\hat{\Psi}(\vec{x}_i) = \hat{c} + \frac{\hat{r}}{\|\hat{V}\hat{V}^T(\vec{x}_i - \hat{c})\|} \hat{V}\hat{V}^T(\vec{x}_i - \hat{c})$$

Note that $\vec{x} - \Psi_{V,c}(\vec{x}) \perp \Psi_{V,c}(\vec{x}) - \vec{y}$ since Ψ is an orthogonal projection.

The optimal subspace V is given by $\hat{V} = (\vec{v}_1, \ldots, \vec{v}_{d+1})$, where $\vec{v}_i, i \in \{1, \ldots, d+1\}$ is the ith eigenvector ranked in descending order of $(\mathbf{X} - 1_N \bar{\mathbf{X}})^T (\mathbf{X} - 1_N \bar{\mathbf{X}})$. This is a result taken from linear PCA, where we eigendecomposed the covariance matrix Σ to obtain the decreasing eigenvalues and eigenvectors to obtain the principal component

scores and loadings. In fact, the matrix given by $(\mathbf{X} - 1_N\bar{\mathbf{X}})^T(\mathbf{X} - 1_N\bar{\mathbf{X}})$ is equivalent to:

$$\begin{aligned}(\mathbf{X} - 1_N\bar{\mathbf{X}})^T(\mathbf{X} - 1_N\bar{\mathbf{X}}) &= \mathbf{X}^T\mathbf{X} - \bar{\mathbf{X}}^T 1_N^T\mathbf{X} - \mathbf{X}^T 1_N\bar{\mathbf{X}} + \bar{\mathbf{X}}^T 1_N^T 1_N\bar{\mathbf{X}} \\ &= \mathbf{X}^T\mathbf{X} - N\overline{x}\overline{x}^T \\ &= N \cdot \Sigma_X\end{aligned}$$

In other words, the first step of spherical PCA is to eigendecompose the scaled covariance matrix Σ_X! This is where the "PCA" aspect of spherical PCA arises.

If $\vec{z}_i = \bar{\mathbf{X}} + \hat{V}\hat{V}^T(\vec{x}_i - \bar{\mathbf{X}})$ are a change of basis to affine subspace V, then it can be shown that the minimizing pair $(\vec{\eta}^*, \vec{\xi}^*)$ of loss function $g(\vec{\eta}, \vec{\xi}) = \sum_{k=1}^N (\vec{z}_i^T\vec{z}_i + \vec{\eta}^T\vec{x}_i + \vec{\xi})^2$ is:

$$\begin{aligned}\vec{\eta} &= -H^{-1}\omega \\ \vec{\xi} &= -\frac{1}{N}\sum_{k=1}^{N}(\vec{z}_i^T\vec{z}_i + \vec{\eta}^T\vec{z}_i)\end{aligned}$$

where H and ω are defined as:

$$\begin{aligned}H &= \sum_{i=1}^{N}(\vec{z}_i - \overline{z})(\vec{z}_i - \vec{z})^T \\ &= N\Sigma_z \\ \vec{\omega} &= \sum_{i=1}^{N}\left(\|\vec{z}_i^T\vec{z}_i\| - \frac{1}{N}\sum_{j=1}^{N}\|\vec{z}_j^T\vec{z}_j\|\right)(\vec{z}_i - \overline{z})\end{aligned}$$

The H matrix can be seen as a centered matrix that is the covariance matrix of new coordinates $\mathbf{Z}$ multiplied by number of observations N. $\vec{\omega}$ is a vector where ω_i is the centered ith z-coordinate scaled by a weight, where the weight is a centered magnitude of the ith z-coordinate.

The optimal parametrization $(\hat{V}, \hat{c}, \hat{r})$ of the projection of $\mathbf{X} \in \mathbb{R}^{N\times D}$ onto the sphere $S_V(c, r)$ is:

$$\begin{aligned}\hat{V} &= (\vec{v}_1, \ldots, \vec{v}_{d+1}) \\ \hat{c} &= -\frac{\vec{\eta}^*}{2} \\ \hat{r} &= \frac{1}{N}\sum_{k=1}^{N}\|\vec{z}_i - \hat{c}\|\end{aligned}$$

The projection map $\hat{\Psi}$ of data matrix $\mathbf{X}$ onto sphere $S_{\hat{V}}(\hat{c}, \hat{r})$ is the projection map onto affine subspace $\hat{c} + \hat{V}$, given by:

$$\hat{\Psi}(\vec{x}_i) = \hat{c} + \frac{\hat{r}}{\|\hat{V}\hat{V}^T(\vec{x}_i - \hat{c})\|}\hat{V}\hat{V}^T(\vec{x}_i - \hat{c})$$

2.3 Algorithm

We have now defined spherical PCA (SPCA) to project the data $\mathbf{X}$ down to single sphere S_V. In general, this single sphere will typically not be a sufficient approximation for the inherent manifold M. Instead, we partition the space $\mathbb{R}^D$ into k disjoint subsets $C_1, \ldots, C_k$.

For the kth disjoint subset, we can define a data matrix $\mathbf{X}_k = \{X_i : X_i \in C_k\}$ that is a partition of the original data that lies within C_k. After applying SPCA to $\mathbf{X}_k$, we obtain spherical volume, center, and radius $(\hat{V}_k, \hat{c}_k, \hat{r}_k)$ alongside projection map Φ_k as a map from $x \in C_k$ to $y \in S_{\hat{V}_k}(\hat{c}_k, \hat{r}_k)$. A spherelets estimation $\hat{M}$ of the manifold M can be obtained by setting $\hat{M} = \bigcup_{k=1}^{K} \hat{M}_k$, where $\hat{M}_k$ is the local SPCA in the kth region and $\hat{M}_k = S_{\hat{V}_k}(\hat{c}_k, \hat{r}_k) \cap C_k$.

We thus arrive at the spherical PCA algorithm. The basic strokes of the spherical PCA process are:

1. Subdivide the space (and dataset) into k partitions: $\{C_k\}_{k=1}^K$ and $\{M_k\}_{k=1}^K$,
2. Calculate a submanifold $\hat{M}_k$ and projection map $\hat{\Psi}_k$ for each submanifold,
3. Find the union of $\{M_k\}_{k=1}^K$ and Ψ_k to obtain estimates $\hat{M}$ and Ψ.

The specific algorithm is as follows:

Algorithm 1 Spherelets

Input: Data matrix $\mathbf{X}$; intrinsic dimension d; partition $\{C_k\}_{k=1}^K$

Output: Local estimated manifolds $\hat{M}_k$ and projection map $\hat{\Psi}_k, k \in \{1, \ldots, K\}$; global estimated manifold $\hat{M}$ of intrinsic manifold M and projection map Ψ.

1: **for** (k = 1 : K) **do**
2: Define $\mathbf{X}_{[k]} = \mathbf{X} \cap C_k$
3: Calculate $\hat{V}_k, \hat{c}_k, \hat{r}_k$
4: Calculate $\hat{\Psi}_k(x) = \hat{c}_k + \frac{\hat{r}_k}{\|\hat{V}_k\hat{V}_k^T(x-\hat{c}_k)\|}(x - \hat{c}_k)$
5: Calculate $\hat{M}_k = S_{\hat{V}_k}(\hat{c}_k, \hat{r}_k) \cap C_k$
6: **end for**
7: Calculate $\Psi(x) = \sum_{k=1}^K \mathbf{1}_{\{x \in C_k\}}\hat{\Psi}_k(x)$, and $\hat{M} = \bigcup_{k=1}^K \hat{M}_k$.

3 Discussion

Previously within this paper, I have incorporated parts of the geometric intuition and connections to linear PCA in other sections but will rehash it briefly in this section as well. Within the Section 2.1, I also discuss the Main Theorem and its theoretical guarantees for the relative performance of spherelets in terms of covering number.

Within this section, I will discuss the various strengths and weaknesses that spherelets provide relative to local linear PCA using hyperplanes. I will also incorporate extensions and applications of spherelets to other aspects of dimension reduction and data visualization.

3.1 Strengths

The two main draws for spherical PCA are its flexibility and its ability to offer out of sample testing without reinitializaiton of the entire algorithm. However, as noted by Li and Dunson, other potential advantages to spherelets can be in lateral applications of the algorithm to other linear-based approximate methods like Isomap and LLE. Spherelets also can be extended to denoise data and stacks up well against competing, well-attested algorithms in the literature.

- **High curvature.** As mentioned prior, the motivating desire behind spherical PCA is to approximate manifolds in lower dimensions that other fully linear methods are not able to capture. Spherical PCA projects a subset of data down to a sphere living on an affine subspace of $\mathbb{R}^D$, allowing
- **Analytic solution...** ... the depends on choice of partitioning algorithm. Because spherical PCA relies on a projection to affine subspace for a given partition C_k in $\mathbb{R}^D$, there is a closed-form unique solution for a partition of the original dataset. However, the choice of partition algorithm can be done in myriad ways. In the examples below, I sometimes use a naive partitioning algorithm that splits the data into k consecutive subsets, while the function written by Minerva Mukhopadhyay (Mukhopadhyay 2019) instead uses iterated PCA to recursively divide the dataset and run spherical PCA until a desired level of error is reached and no further gains in error can be achieved. Other algorithms mentioned before include METIS and cluster trees; in fact, any sort of clustering algorithm can be used, but care must be taken to avoid overfitting and computational expensiveness when $D >> d$.
- **Modeling and visualization.** Like linear PCA, spherelets can be a powerful tool both to visualize intrinsically curved data. Examples given in the following section include datasets with curved manifolds that might perform poorly when modeled with hyperplanes, such as the Euler spiral and the 3D cylinder. Spherelets can also be used to build a parametrized model, since the algorithm returns the

centers, radii, and subspace basis vectors in the form $S_V(c, r)$ of the spherelet that is used.

- **Out of sample testing.** Spherical PCA goes beyond what most locally linear methods halt at—spherelets define a manifold M. As such, spherical PCA lends itself easily to out-of-sample testing, since information is learned about the subspace directly. Given the results of the spherical PCA $\{S_{\hat{V}_k}(\hat{c}_k, \hat{r}_k)\}_{k=1}^K$, we can project new data down and measure the test MSE. One concern with this approach is that the curse of dimensionality comes into play, since d is one of our tuning parameters in the spherelets algorithm, so a robust measure of error must be picked that does not grow exponentially with d.
- **Spherical analog to linear methods.** As we will explore later, spherical PCA can be seen as a starting point to develop spherical settings for originally linear methods. Just as linear PCA is a precursor to other techniques, spherical PCA can also be seen as a spherical precursor to more complex techniques, where aspects of spherical PCA can be taken to better fit curved data.

3.2 Weaknesses

While spherical PCA is capable of fitting manifolds that have higher degrees of curvature than linear PCA, spherelets still struggle in some similar areas akin to linear PCA. Main disadvantages of spherelets include sensitivity to non-uniform dimensions and curvature along the manifold and computational inefficiency when paired with a poor choice of partitioning algorithm and choice of intrinsic manifold dimensionality d.

- **Manifolds with non-uniform curvature.** The Main Theorem offers a perspective to why the spherical submanifold method is not particularly effective with manifolds with large changes in curvature. As V_ϵ approaches V, we have either 1) relaxed the accuracy threshold ϵ which is undesirable for the sake of attaining accurate and precise results, or 2) the covering number approaches that of local linear PCA using a dictionary of basis functions $\mathcal{H}$ that's based on hyperplanes instead of spheres. The first circumstance is a trivially poor example and is true of any algorithm. However, in the second circumstance, spherelets would not perform drastically different as opposed to local linear PCA.
- **Differences in scale across dimensions.** As in linear PCA, spherelets are also sensitive to scale. A dataset with variables that differ drastically in range and variance would challenge spherelets the same way that linear PCA would be challenged. This is because in spherical PCA, the projection map Ψ still relies on an eigendecomposition and transformation to affine subspace spanned by V as a core principal. If one particular dimension displays a much higher degree of variance that outweighs the remaining dimensions, then the overall manifold may look up looking more and more *ellipse-like*. The Main Theorem states that

the ϵ-spherical set of points F_ϵ is affected by the difference between minimum and maximum curvature in the manifold, so more likely than not the spherelets algorithm would use an excessive amount of spheres to approximate the ellipsoid manifold.

With that being said, a potential avenue for exploration would be to generalize the concept of spherelets to "*ellipselets*" that would be able to incorporate different bounded axes within ellipsoids as a dictionary of basis functions. This would offer more flexibility to detect manifolds with greater extremes in curvature.

- **User-supplied intrinsic dimension** d**.** A major downside to the spherelets algorithm is that it requires an input dimension d or for the runner of the algorithm. The loss function is not guaranteed to be convex in this scenario, as with increased dimensionality in the form of d-hyperspheres, there is always the increased potential to overfit. Some degree of content knowledge is required on part of the user to either 1) specify if only one intrinsic dimension makes physical/realistic sense or 2) tune the d parameter, which is often computationally expensive with complicated partitioning algorithms.
- **Computationally expensive.** While the projection step of the algorithm is analytic, the partitioning of the data for an "optimal" solution is an NP-hard problem which may not be guaranteed to converge. Additionally, the curse of dimensionality appears in the Main Theorem, where poor approximations of the manifold result in $O(c^{-n/2})$ computational complexity as opposed to $O(c^{-n/3})$. The specified algorithm proposed by Li and Dunson also features computationally-intensive bottlnecks that do not scale well, such as matrix inversions (calculation of the H^{-1} matrix) and traversing across matrices (calculating ω and $\mathbf{Z}$).

3.3 Further Applications

Many dimension reduction techniques rely on linear methods due to the well-behaved nature of linear objects. Some of these lend themselves to representation in a spherical subspace. Just as PCA can be seen as a starting point for other related techniques like CCA or NMF, spherelets can also be extended to myriad techniques. A couple options that I will cover briefly are manifold denoising and data visualization.

- **Manifold Denoising**. One class of denoising algorithms that exists serves to denoise data which is assumed to near a intrinsically lower-dimensional manifold. Gaussian Blurring Mean Shift (GBMS), Local Tangent Projection (LTP), and the in-between algorithm Manifold Blurring Mean Shift (MBMS) use linear methods to approximate a manifold obtained through standard PCA. Li and Dunson changed the underlying standard PCA assumption in MBMS to that of spherical PCA, which they deemed Spherical Manifold Blurring Mean Shift (SMBMS) or alternatively, Local Spherical Projection (LSP) (Didong Li and Dunson 2019).

This method was shown to do better both with toy models as well as with real data (USPS handwritten digits dataset).

- **Data Visualization**. Based on the experimentation of Li and Dunson, the gold standard for dimension reduction for the purpose of data visualization has been to use t-distributed Stochastic Neighbors (tSNE). In a another paper titled "Geodesic Distance Estimation with Spherelets", Li and Dunson extend the concept of tSNE to use geodesic distance along the surfaces of derived spheres instead of Euclidean distance (Li and Dunson 2019).

4 Examples

To generate numerical examples, I used the `SPCA` and `SS_calc` functions written by co-author Minerva Mukhopadhyay (Mukhopadhyay 2019). The `SPCA` function takes in a matrix of N observations $\vec{x}_i \in \mathbb{R}^D, i \in 1, \ldots, N$ and returns the error given by spherical and local PCA (`SS` and `SS_new`), as well as the projected values `Y_D`.

4.1 Euler Spiral

The Euler spiral is a curve in $\mathbb{R}^1$ that has a curvature that changes linearly with arc length. In other words, $\kappa(s) = s$. The Euler spiral can be parametrized as follows:

$$\begin{bmatrix} x(s) \\ y(s) \end{bmatrix} = \begin{bmatrix} \int_0^s \cos(t^2)dt \\ \int_0^s \sin(t^2)dt \end{bmatrix} \quad s \in [0, 4]$$

We use the Euler spiral as a demonstration to see how spherical PCA is able to handle regions with curvature. The first set of graphs shows what spherical PCA returns after partitioning the Euler spiral in a naive manner (splitting the data so $|X_1| = \cdots = |X_k|$, each set is the same size). The second set of plots uses the more robust iterated PCA method by Mukhopodhyay. Progressive plots show deeper levels of recursion up to a tolerance level δ.

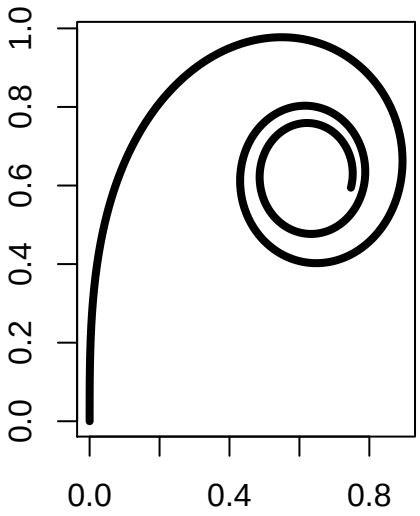

Figure 1: Euler spiral.

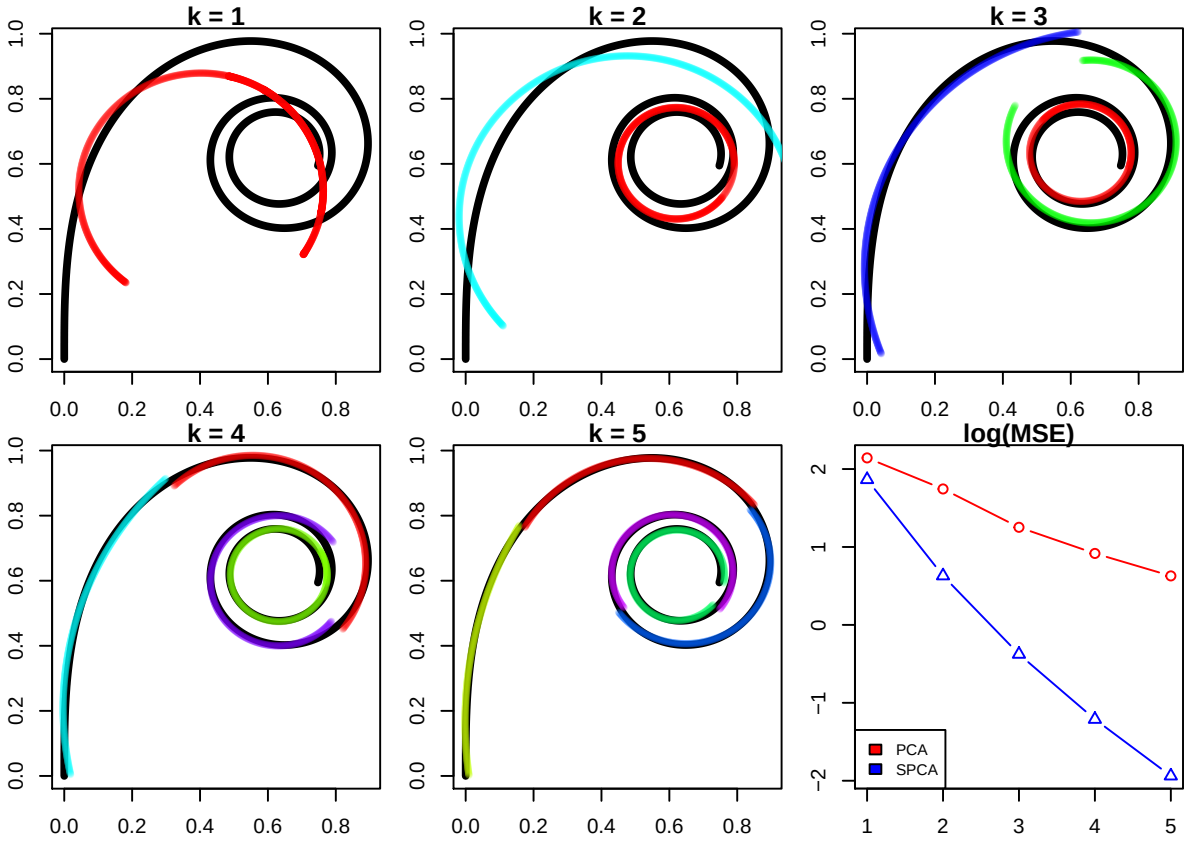

Figure 2: Spherical PCA performed on an Euler spiral with $k = 1, \ldots, 5$. Partitioning done naively.

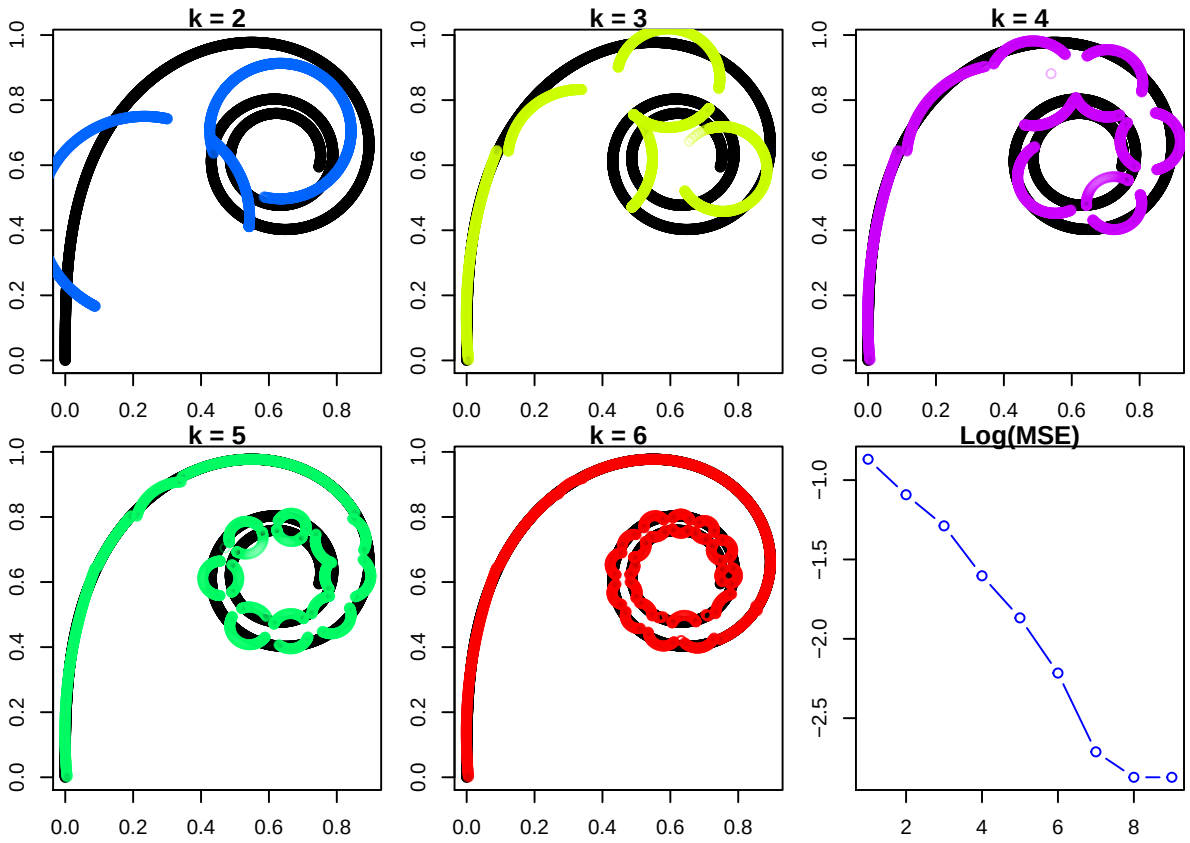

Figure 3: Spherical PCA partitioned using iterated PCA according to Mukhopadhyay 2019. Note that using iterated PCA results in better and better fits as the number of partitions increases, but the spherelets algorithm fits many more circles than necessary in the tight spiral.

We see that the difference in partitioning algorithm made a drastic difference in how the spherical PCA ended up dividing the Euler spiral. For low values of k, the circles were distributed across the length of the Euler spiral to cover as much of the "tail" and the "whorl" as much as possible. However, with higher levels of k, it became clear that the algorithm was attempting to cover the tight curl by lining up circles side by side down the arc length as opposed to placing overlapping circles. The naive partitioning performed in a more intuitive manner by aligning as much of the curve with the sphere as possible.

4.2 Helix

A helix has constant curvature κ and is a simple curve in $\mathbb{R}^3$:

$$\begin{bmatrix} x(s) \\ y(s) \\ z(s) \end{bmatrix} = \begin{bmatrix} \cos(s) \\ sin(s) \\ s \end{bmatrix}, \; s \in [0, 5\pi]$$

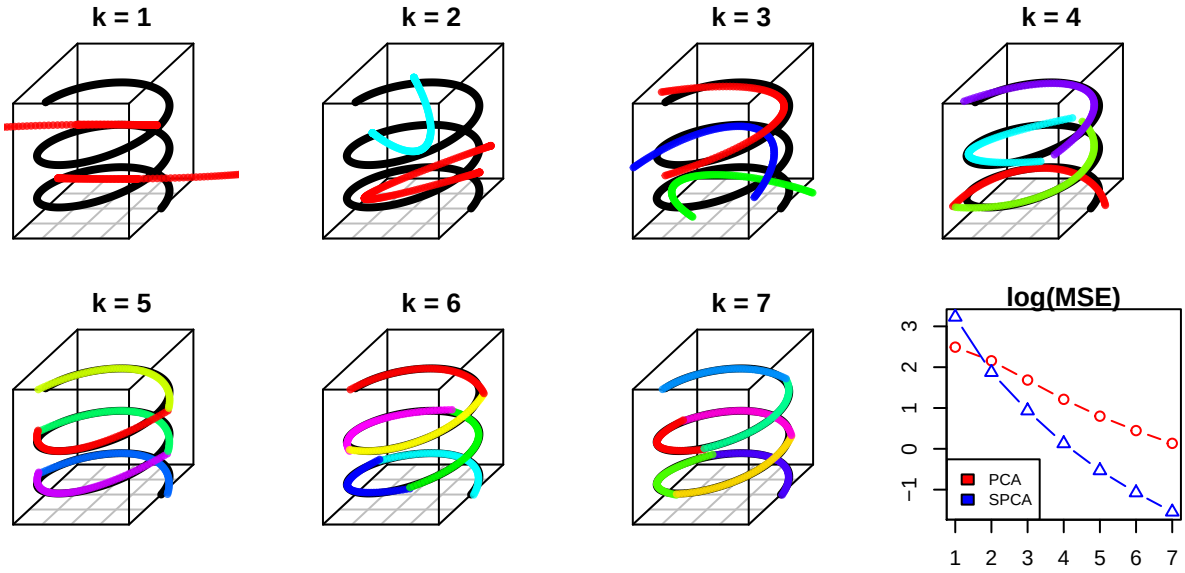

Figure 4: Spherical PCA performed on a helix with $k = 1, \ldots, 7$. Partitioning done naively. MSE plot included.

The native partition does poorly at very low values of k but performs much better with increased values of k, since circles can perfectly approximate a setion of the helix due to their constant curvature.

4.3 Cylinder

An unbounded cylinder is an interesting toy model for spherical PCA since in two dimensions the curvature along the surface of the cylinder is constant κ, but along the axis of the cylinder, curvature is infinite. We know that spherelets tend to do poorly when the range in curvature along the manifold is high. A cylinder is parametrized below:

$$\begin{bmatrix} x(s) \\ y(s) \\ z(s) \end{bmatrix} = \begin{bmatrix} \cos(s) \\ \sin(s) \\ 3s \end{bmatrix} \quad s \in [0, 2\pi]$$

I sampled 2500 points along the surface of the cylinder to generate the dataset:

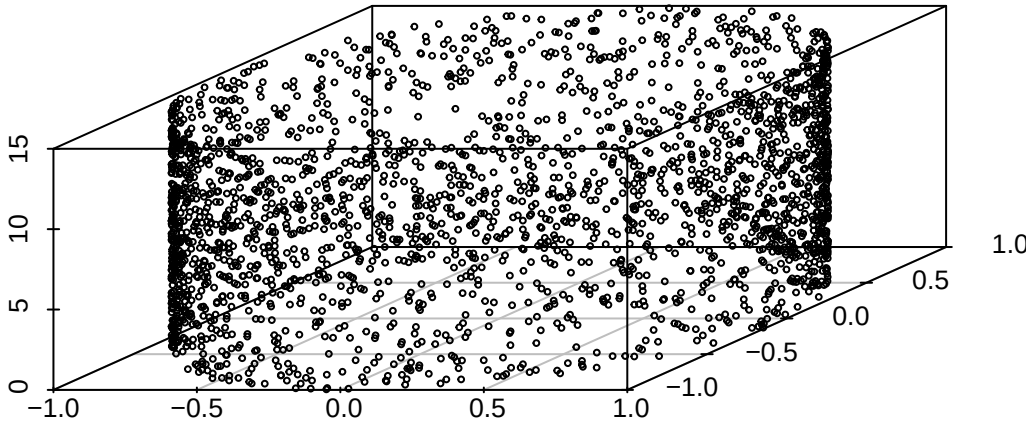

Figure 5: A cylinder in $\mathbb{R}^3$.

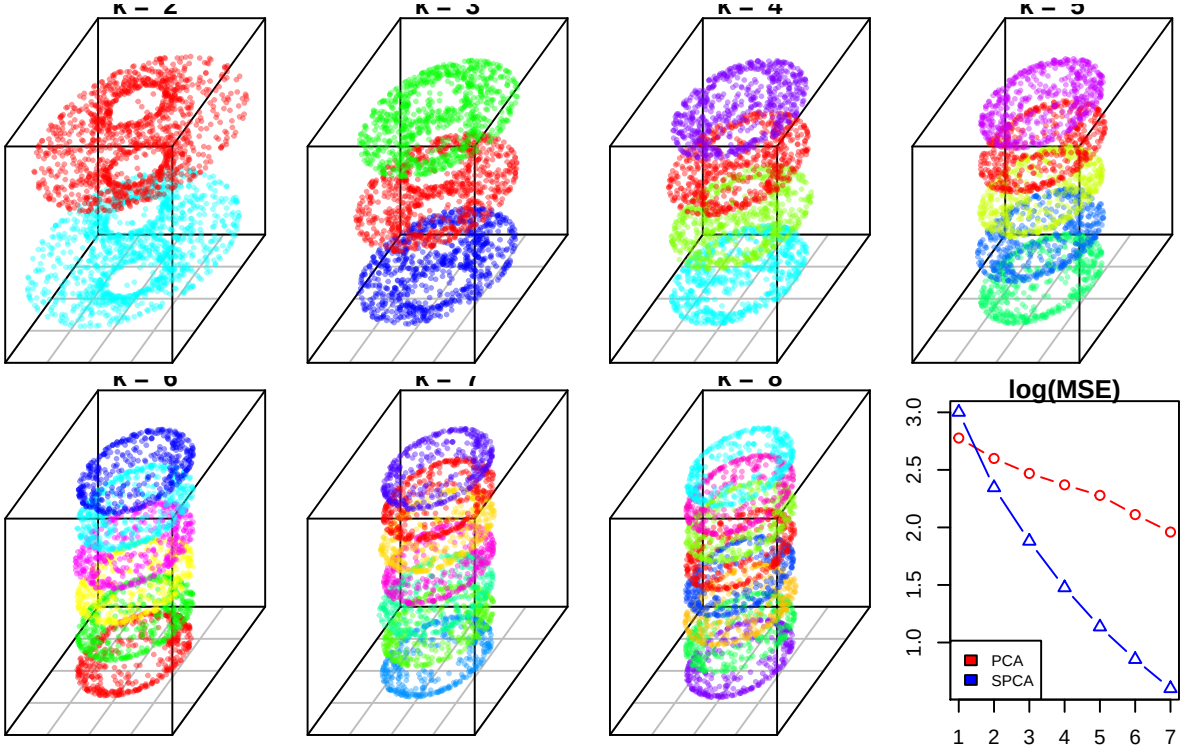

Figure 6: Spherical PCA performed on a cylinder with $k = 1, \ldots, 7$. Partitioning done naively. MSE plot included.

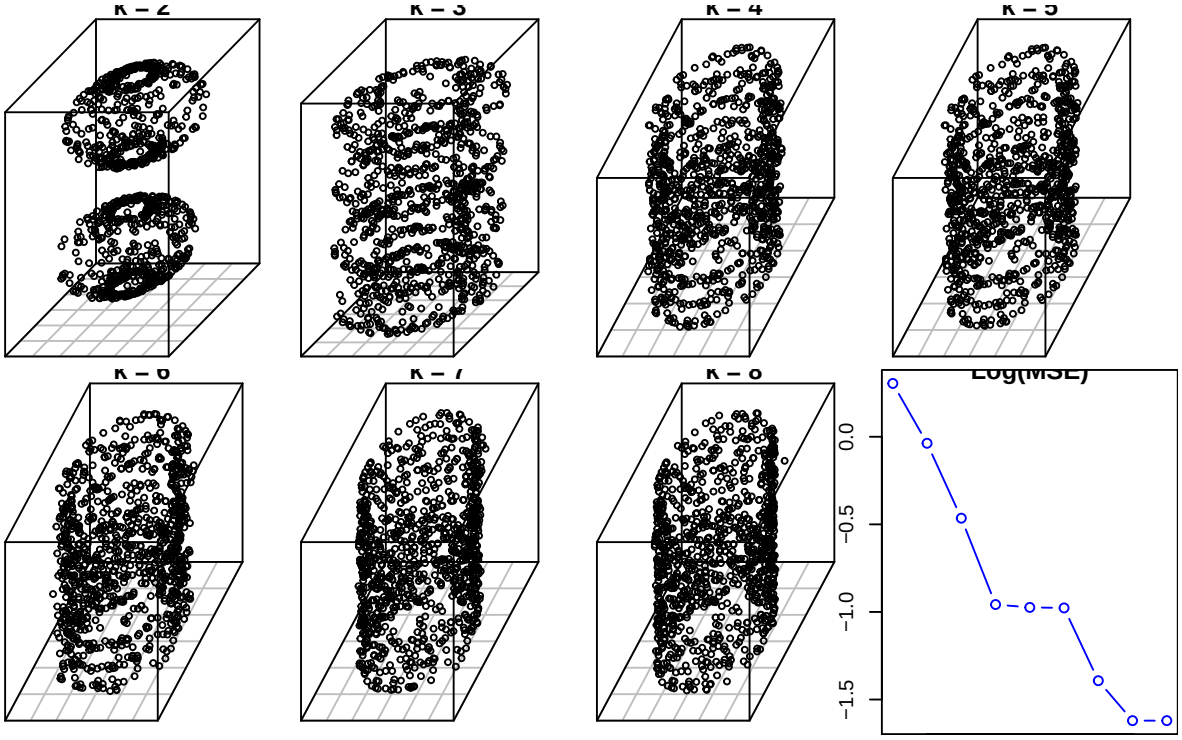

Figure 7: Spherical PCA on a cylinder based on iterated PCA. Notice that for the $k = 2$ case, the iterated PCA version is able to much more reasonably approximate the sphere, unlike the naive partioning algorithm, demonstrating that the choice in partitioning algorithm can have profound effects on final outcome for low k.

While SPCA outperforms regular PCA when it comes to MSE for cylinders, we see that spherical PCA still struggles a little with low values of partitioning k. We can notice a slight difference visually in how the two partitioning algorithms handle a cylinder. However, it seems that in both scenarios, spherical PCA tends to stack identical spheres atop each other along the axis of the cylinder in order to cover as much of the cylinder in as few spheres as possible, which creates a "bumpy" cylinder. An ellipsoid may be better suited to handle this particular task.

4.4 Handwritten Digits

The MNIST dataset is a famous one consisting of handwritten images. There are a total of 42,000 observations in the dataset. Each digit is coded as a vector in $\mathbb{R}^7 84$ to account for the 28×28 pixel image. In this case, lying on a lower-dimensional manifold would be attempting to find d-dimensional hyperspheres that the observations lie near. Partitioning the dataset using iteracted PCA would be computationally expensive given the size of the dataset and is difficult to define in 784-dimensional space. For the sake of simplicity and to illustrate the effect of dimensionality, the partitioning step is skipped

in this implementation of spherical SPCA. Instead, different values of d are tuned to show how spherical PCA might be applied.

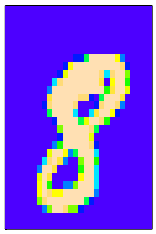

Figure 8: The first handwritten 8 digit taken from the MNIST dataset.

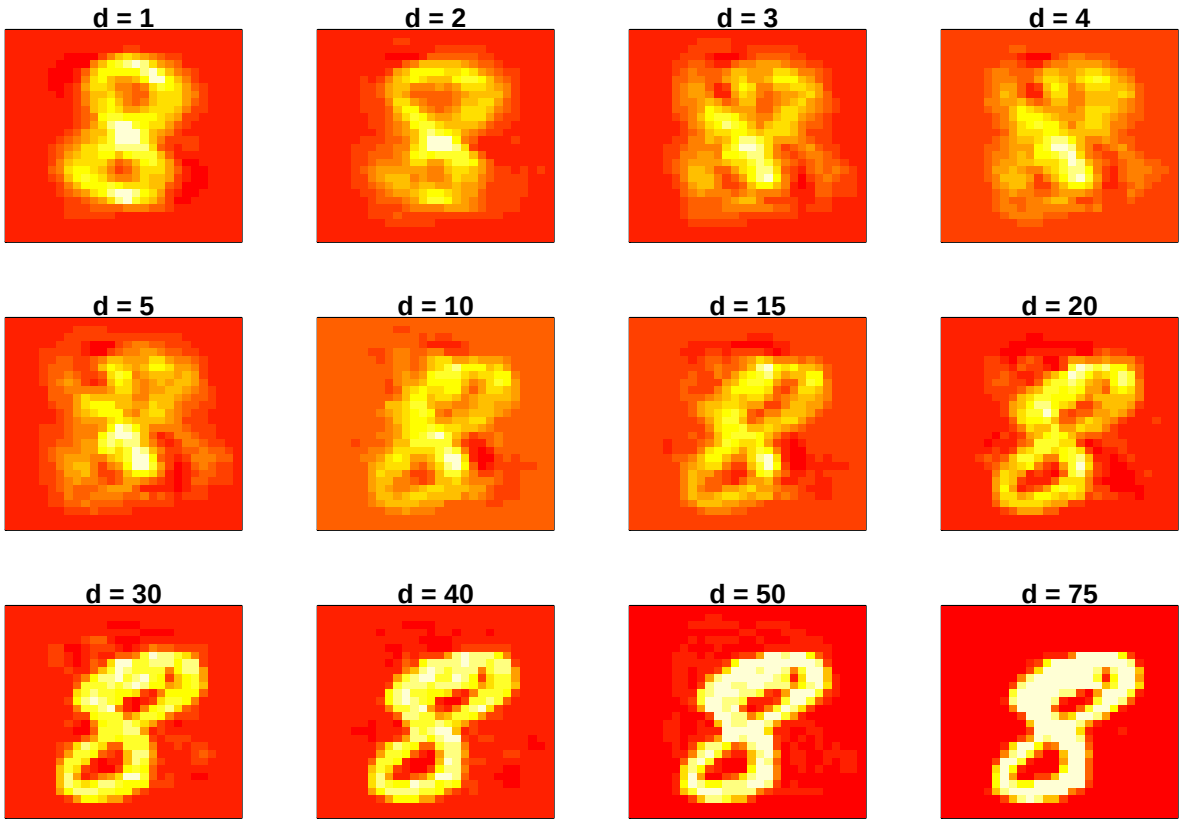

Figure 9: The projected first handwritten 8 digit based on varying dimensions d. The definition of the digit gets steadiliy and steadily less clear from the first image $d = 1$, since the first image is most likely the closest to a simple average of all the observed digits. Once d crosses the $d = 15$ mark, it seems to regain clarity, though at this point d approaches sample size and we are nearing overfitting. A balance must be struck between optimal d to not be overly general but also not to be overly fit so as to lose generality. Therein lies the conundrum of spherical PCA!

5 References

Beygelzimer, Alina, Sham Kakade, and John Langford. 2006. "Cover Trees for Nearest Neighbor." ACM.

Didong Li, Minerva Mukhopadhyay, and David B Dunson. 2019. "Efficient Manifold Approximation with Spherelets." *arXiv*, February.

Duda, Richard O, David G Stork, and Peter E Hart. 2000. *Pattern Classification*. 2nd ed. Wiley.

Hurtado, Jorge E. 2012. *Structural Reliability*. Vol. 17. Springer-Verlag.

Karypis, George, and Vipin Kumar. 1998. "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs." *SIAM Journal on Scientific Computing* 20 (1): 359–92. doi:10.1137/s1064827595287997.

Lee, John A, and Michel Verleysen. 2008. *Nonlinear Dimensionality Reduction*. Springer Science.

Lee, Seunggeun, Fei Zou, and Fred A Wright. 2010. "Convergence and Prediction of Principal Component Scores in High-Dimensional Settings." *The Annals of Statistics* 38 (6): 3605–29. doi:10.1214/10-aos821.

Li, Didong, and David B Dunson. 2017. "Efficient Manifold and Subspace Approximations with Spherelets." *arXiv*, June. ResearchGate.

———. 2019. "Geodesic Distance Estimation with Spherelets." *arXiv*, June.

Mukhopadhyay, Minerva. 2019. "Efficient Manifold Learning Using Spherelets." Github. April 9.

Szlam, Arthur. 2009. "Asymptotic regularity of subdivisions of Euclidean domains by iterated PCA and iterated 2-means." *Applied and Computational Harmonic Analysis*, March. Academic Press. https://www.sciencedirect.com/science/article/pii/S1063520309000190.