FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

MODELING MULTIVARIATE DATA WITH PARAMETER-BASED SUBSPACES

By

AJAY GUPTA

A Dissertation submitted to the
Department of Statistics
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Degree Awarded:
Spring Semester, 2016

Ajay Gupta defended this dissertation on May 18, 2016.
The members of the supervisory committee were:

Adrian Barbu
Professor Directing Thesis

Anke Meyer-Baese
University Representative

Yihuan She
Committee Member

Jinfeng Zhang
Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the dissertation has been approved in accordance with university requirements.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

The following abbreviations are used in this dissertation, and are listed below for clarity.

- **3DMM**: three-dimensional morphable model
- **AAM**: active appearance model
- **CT**: computed tomography
- **FICO**: Fair Isaac Corporation
- **IPCA**: independent principal component analysis
- **LDA**: linear discriminant analysis
- **PCA**: principal component analysis
- **PPCA**: parameterized principal component analysis
- **RMSE**: root mean squared error
- **SVD**: singular value decomposition

LIST OF SYMBOLS

The following symbols are used often in this dissertation, and are listed below to povide an easier reference.

- $\mathbf{0}_{M\times N}$: M row by N column matrix with zeros for all elements
- $\mathbf{1}_{(A)}$: (scalar) indicator function, equal to one if A is true and zero otherwise
- B: number of bin endpoints resulting from division of range of parameter θ into bins
- I_N: identity matrix of size $N \times N$, with ones on diagonal and zeros elsewhere
- K: dimension of manifold ambient space, i.e. dimension of each observation
- $m_{b,k}$: indicator function that pixel (or other element type) k is relevant to bin endpoint b
- $\mathbb{N}$: the set of natural numbers
- n: number of observations in training data
- n_c: number of cycles of optimizing mean vectors, basis vectors, and coefficients in PPCA unless converged earlier
- n_m: number of iterations to use in each gradient descent optimization of PPCA mean vectors
- n_v: number of iterations to use in each gradient descent optimization of PPCA basis vectors
- $\boldsymbol{p}_{b,v}$: vector v out of V total vectors in basis representing variation away from mean for observations relevant to bin endpoint b
- V: dimension of each linear manifold used to estimate larger, non-linear manifold
- $w_{b,i}$: weight that observation i has for bin endpoint b
- $\boldsymbol{x}_i$: observation i, as column vector
- α_m: learning rate used in each gradient descent optimization of PPCA mean vectors
- α_v: learning rate used in each gradient descent optimization of PPCA basis vectors
- $\boldsymbol{\beta}_i$: coefficient vector that is reduced-dimension representation of $\boldsymbol{x}_i$
- θ: parameter of contextual information that parameterized principal component analysis uses to determine how an observation should be modeled
- $\boldsymbol{\mu}_b$: mean vector used to represent observations relevant to bin endpoint b

ABSTRACT

When modeling multivariate data such as vectorized images, one might have an extra parameter of contextual information that could be used to treat some observations as more similar to others. For example, images of faces can vary by yaw rotation, and one would expect a face rotated 65° to the left to have characteristics more similar to a face rotated 55° to the left than to a face rotated 65° to the right. We introduce a novel method, parameterized principal component analysis (PPCA), that can model data with linear variation like principal component analysis (PCA), but can also take advantage of this parameter of contextual information like yaw rotation.

Like PCA, PPCA models an observation using a mean vector and the product of observation-specific coefficients and basis vectors. Unlike PCA, PPCA treats the elements of the mean vector and basis vectors as smooth, piecewise linear functions of the contextual parameter. PPCA is fit by a penalized optimization that penalizes potential models which have overly large differences between corresponding mean or basis vector elements for similar parameter values. The penalty ensures that each observation's projection will share information with observations that have similar parameter values, but not with observations that have dissimilar parameter values.

We tested PPCA on artificial data based on known, smooth functions of an added parameter, as well as on three real datasets with different types of parameters. We compared PPCA to independent principal component analysis (IPCA), which groups observations by their parameter values and projects each group using principal component analysis with no sharing of information for different groups. PPCA recovers the known functions with less error and projects the datasets' test set observations with consistently less reconstruction error than IPCA does. PPCA's performance is particularly strong, relative to IPCA, when there are limited training data.

We also tested the use of spectral clustering to form the groups in an IPCA model. In our experiment, the clustered IPCA model had very similar error to the parameter-based IPCA model, suggesting that spectral clustering might be a viable alternative if one did not know the parameter values for an application.

CHAPTER 1

INTRODUCTION

In recent years, storing and modeling multidimensional data have become very common. Potential datasets include different attributes of potential customers, multiple currencies' exchange rates for each day, and vectorized images. Although these data often lie on non-linear manifolds, a linear manifold or a combination of linear manifolds can often provide a practical and suitably accurate approximation. Particularly for data of very high dimensionality such as vectorized images, a model may need to produce a reduced-dimension representation of the original observations. One particularly effective technique for modeling linear manifolds and incorporating dimensionality reduction is principal component analysis (PCA), which finds a basis $\boldsymbol{P}$ of vectors that can capture the highest-variance directions from the original data [11].

Pitelis et al. (2013) showed how an "atlas" of overlapping linear manifolds that they labeled "charts" could model a non-linear manifold very effectively [13]. Their model was learned by a hill-climbing approach which alternated between assigning observations to charts based on the observations' values and refitting each chart using PCA performed on the relevant subset of observations. The initial charts, which were necessary for the first assignments, could be found by PCA on bootstrap samples. The number of charts was selected by the method based on a user-supplied penalty λ.

Vidal, Ma, and Sastry (2005) introduced Generalized Principal Component Analysis (GPCA), which similarly addressed the idea of dividing a larger manifold into multiple local manifolds. GPCA used polynomials based on Veronese maps to modify and combine elements of the original data vectors. GPCA could still learn the coefficients of the monomial terms by PCA, though, because the relationship between the full polynomial and these coefficients was still linear [15]. The experimental success of GPCA showed that multiple applications of (linear) PCA could be used to learn a complicated manifold, although the local manifolds learned were typically non-linear. The authors noted, though, that piecewise linear models (which could be learned by multiple PCA applications without GPCA's polynomials) are "excellent" in many practical applications at

balancing the need for model expressiveness with the desire for model simplicity [14]. Like the atlas method, GPCA could also select the appropriate number of local manifolds, but using the ranks of Veronese maps evaluated on the observations instead of user-supplied parameters [15].

Techniques such as GPCA and the atlas-based method exist for identifying local manifolds for observations based on the observations' values, and for identifying the number of local manifolds. Situations exist, however, in which data are thought to lie approximately on local manifolds that estimate a larger manifold, but one knows which local manifold corresponds to each observation instead of needing the algorithm to discover this. For example, one may be modeling images of vehicles using a known class ("car," "motorcycle," "SUV," or "truck") for each observation. Linear discriminant analysis (LDA) uses linear manifolds to perform dimensionality reduction in a scenario with such classes. However, LDA chooses its basis vectors based on what can separate classes rather than what can capture the variation of each class [11]. Other techniques such as Class-Information-Incorporated Principal Component Analysis [3], Joint Parameterized Atom Selection [16], Locality Preserving Projections [8], Multi-Manifold Semi-Supervised Learning [6], Multimodal Oriented Discriminant Analysis [5], and Semi-Supervised Dimensionality Reduction [18] learn manifolds in the presence of subspaces or classes. Like LDA, however, these are focused on classification to a local manifold rather than focused on modeling the observations once the classes are known.

One reason that modeling known classes has been effectively unaddressed is that one could treat the modeling problem as many separate problems, each of which could be addressed by existing techniques such as PCA. For notational simplicity, we will refer to the use of a separate PCA model for each group as Independent Principal Component Analysis (IPCA), because none of the classes' models use information from the other classes' observations. If one had a quantitative "parameter" rather than categories to determine the local manifold, one would perform IPCA by forming histogram-like bins and running one PCA model per bin.

Various applications exist in which this extra contextual information would be quantitative. For vehicle images, one could model them differently based on the vehicles' weights, volumes, or prices (MSRPs). Daily percent changes in a stock's closing share price could use the stock's market capitalization, because smaller-capitalization stocks are thought to have more volatile price movements. Lenders with multiple recorded attributes about their borrowers could use the borrowers' rates of interest or FICO credit scores as the parameters.

This contextual parameter carries ordinal and interval information that would be ignored by IPCA. Consider borrower data such as a borrower's number of late payments, with credit scores as the parameter, and bins 300-350, 350-400, and so on until 800-850. The average late payments might decrease in the training examples as the bin increases, except that the 400-450 bin might have a surprisingly low average. IPCA would ignore the other bins and the pattern they form, which could overfit the training examples in the 400-450 bin. Additionally, IPCA would treat customers with credit scores of 355 and 845 as completely different from one with a credit score of 345, because all three are in different bins. It would ignore that the difference between 345 and 355 is much smaller than the difference between 345 and 845, rather than enforcing similarity between how the 345-score and 355-score observations are modeled.

In this dissertation, we propose a new method called parameterized principal component analysis (PPCA) for creating a PCA-like linear model for multivariate data that are associated with a separate parameter with known, observation-specific values. Like IPCA, PPCA makes multiple linear models of mean vectors and bases, which are based on known divisions of the parameter space. Unlike IPCA, PPCA interpolates between the points in the parameter space at which mean vectors and bases were fitted, and penalizes differences in the models for similar parameter values. We describe the PPCA model in Chapter 2, and discuss its implementation in Chapters 3 and 4. In Chapter 5, we apply PPCA to artificial data following smooth functions of the parameter, and to three real datasets: shapes of differently-sized lymph nodes, human facial images with different degrees of added blurriness, and human facial images with different angles of yaw rotation. In all four experiments, PPCA outperformed IPCA, and was particularly beneficial when the number of training examples was limited.

We also performed an experiment to check whether an IPCA model using spectral clustering could substitute adequately for an IPCA model using binning of a parameter, in case one had an application with inaccessible values of the contextual parameter. In our experiment, we projected shape meshes and appearances of faces with varied yaw rotation. The clustering-based IPCA model were comparable to the parameter-based IPCA models. However, a PCA model with no groups was surprisingly effective, and might be considered superior to both IPCA models, depending on the modeling priorities.

CHAPTER 2

PARAMETERIZED PRINCIPAL COMPONENT ANALYSIS

Parameterized principal component analysis (PPCA) applies to an environment in which there are n observations $\boldsymbol{x}_i$, each of dimension K, and there are B bin endpoints arising from the $B-1$ bins that partition the acceptable range of the parameter θ. Each bin endpoint b has a mean vector $\boldsymbol{\mu}_b$ and V basis vectors $\boldsymbol{p}_{b,v}$. Each bin endpoint corresponds to a value of θ, and an observation $\boldsymbol{x}_i$'s parameter value θ_i dictates $\boldsymbol{x}_i$'s bin, with lower endpoint $b_{(l),i}$ and upper endpoint $b_{(u),i}$. Figure 2.1 shows an example using a parameter that varies from $\theta = 1$ to $\theta = 10$. Note that a bin endpoint usually applies to two bins. For the example in Figure 2.1, the bin endpoint at $\theta = 3$ would be an endpoint for the 2-3 bin and for the 3-4 bin.

The parameter θ_i can be translated into weights $w_l(\theta_i)$ and $w_u(\theta_i)$ for bin endpoints $b_{(l),i}$ and $b_{(u),i}$. Equation (2.1) shows this, using $\theta_l(\theta_i)$ and $\theta_u(\theta_i)$ as the parameter values for the bin's lower and upper endpoint, respectively. Figure 3.1 shows an example for an observation with $\theta_i = 4.4$. It has a 60% weight for the bin endpoint at $\theta = 4$ and a 40% weight for the bin endpoint at $\theta = 5$, because 4.4 is 60% of the way from 5 to 4, and 40% of the way from 4 to 5.

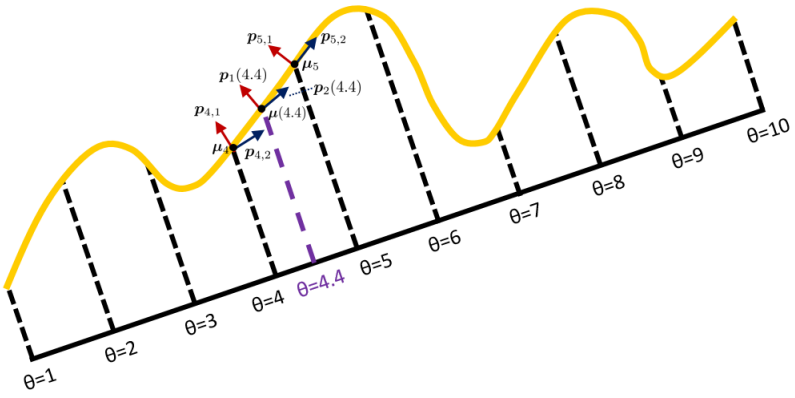

Figure 2.1: Example of Manifold Represented by PPCA

$$w_l(\theta_i) = \frac{\theta_u(\theta_i) - \theta_i}{\theta_u(\theta_i) - \theta_l(\theta_i)}, w_u(\theta_i) = \frac{\theta_i - \theta_l(\theta_i)}{\theta_u(\theta_i) - \theta_l(\theta_i)} \tag{2.1}$$

These weights can produce a mean vector $\boldsymbol{\mu}(\theta_i)$ and a basis $\boldsymbol{P}(\theta_i)$ that are specific to the observation's parameter θ_i, as shown in Equations (2.2) and (2.3).

$$\boldsymbol{\mu}(\theta_i) = w_l(\theta_i)\boldsymbol{\mu}_{b_{(l),i}} + w_u(\theta_i)\boldsymbol{\mu}_{b_{(u),i}} \tag{2.2}$$

$$\boldsymbol{P}(\theta_i) = w_l(\theta_i) \begin{bmatrix} \boldsymbol{p}_{b,1} & \boldsymbol{p}_{b,2} & \cdots & \boldsymbol{p}_{b,V} \end{bmatrix} + w_u(\theta_i) \begin{bmatrix} \boldsymbol{p}_{b+1,1} & \boldsymbol{p}_{b+1,2} & \cdots & \boldsymbol{p}_{b+1,V} \end{bmatrix} \tag{2.3}$$

The model produces a lower-dimensional representation of $\boldsymbol{x}_i$ as the coefficient vector $\boldsymbol{\beta}_i$. This can be translated to a projection of $\boldsymbol{x}_i$ using $\boldsymbol{\mu}(\theta_i) + \boldsymbol{P}(\theta_i)\boldsymbol{\beta}_i$.

2.1 Energy Function

PPCA uses the minimization of an energy function $E(\cdot)$ to achieve a balance between having these projections fit the training examples well and reducing differences between adjacent bin endpoints' corresponding model components.

$$E(\boldsymbol{\mu}, \boldsymbol{p}, \boldsymbol{\beta}, \lambda_{r,m}, \lambda_{r,v}, \lambda_o) = E_{\text{data}}(\boldsymbol{\mu}, \boldsymbol{p}, \boldsymbol{\beta}) + E_{\text{rough.}}(\boldsymbol{\mu}, \boldsymbol{p}, \lambda_{r,m}, \lambda_{r,v}) + E_{\text{ortho.}}(\boldsymbol{p}, \lambda_o) \tag{2.4}$$

Equation (2.4) uses the vectors $\boldsymbol{\mu}$, $\boldsymbol{p}$, and $\boldsymbol{\beta}$, which are stacked from vectors introduced earlier, as detailed in Equation (2.5). The functions $\boldsymbol{\mu}(\theta_i)$ and $\boldsymbol{P}(\theta_i)$ can be derived from the vectors $\boldsymbol{\mu}$ and $\boldsymbol{p}$, and the coefficient vectors $\boldsymbol{\beta}_i$ can be extracted from $\boldsymbol{\beta}$.

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \\ \vdots \\ \boldsymbol{\mu}_B \end{bmatrix}, \boldsymbol{p} = \begin{bmatrix} \boldsymbol{p}_{1,1} \\ \boldsymbol{p}_{1,2} \\ \vdots \\ \boldsymbol{p}_{1,V} \\ \boldsymbol{p}_{2,1} \\ \vdots \\ \boldsymbol{p}_{B,V-1} \\ \boldsymbol{p}_{B,V} \end{bmatrix}, \boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \\ \vdots \\ \boldsymbol{\beta}_n \end{bmatrix} \tag{2.5}$$

The first term within the energy function in Equation (2.4) is $E_{\text{data}}(\cdot)$, which is detailed in Equation (2.6). $E_{\text{data}}(\cdot)$ is the mean ℓ_2 norm of the training examples' approximation errors, using the model's current representation of all training examples' $\boldsymbol{\mu}(\theta_i)$, $\boldsymbol{P}(\theta_i)$, and $\boldsymbol{\beta}_i$.

$$E_{\text{data}}(\boldsymbol{\mu}, \boldsymbol{p}, \boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{\mu}(\theta_i) - \boldsymbol{P}(\theta_i)\boldsymbol{\beta}_i\|^2 \tag{2.6}$$

The second term from $E(\cdot)$, $E_{\text{rough.}}(\cdot)$, is shown in Equation (2.7). It uses the penalty coefficients $\lambda_{r,m}$ and $\lambda_{r,v}$ to ensure smooth functions for the mean vectors and basis vectors, respectively. Differences between corresponding elements for vectors relevant to two endpoints of the same bin are penalized.

$$E_{\text{rough.}}(\boldsymbol{\mu}, \boldsymbol{p}, \lambda_{r,m}, \lambda_{r,v}) = \frac{\lambda_{r,m}}{B-1} \sum_{b=1}^{B-1} \|\boldsymbol{\mu}_b - \boldsymbol{\mu}_{b+1}\|^2 + \frac{\lambda_{r,v}}{B-1} \sum_{b=1}^{B-1} \sum_{v=1}^{V} \|\boldsymbol{p}_{b,v} - \boldsymbol{p}_{b+1,v}\|^2 \tag{2.7}$$

Large values of $\lambda_{r,m}$ and $\lambda_{r,v}$ will enforce more smoothness in the representation, at the expense of the projection error on the training set.

PPCA's two roughness penalty coefficients $\lambda_{r,m}$ and $\lambda_{r,v}$ force the model for an observation to incorporate information from observations with similar observations: those in its bin and those in the adjacent bin(s). The amount of the information sharing depends on the differences in parameter values, even for observations in the same bin. The weighted pooling of information enforces a prior belief that observations with more similar values of a parameter should be modeled in a more similar manner. It enforces smoothness, but not monotonicity. Ordinal trends can still be captured, but only locally. This gives PPCA the ability to approximate more complicated smooth functions, though, such as sinusoidal curves. Like the prior beliefs in Bayesian models, PPCA's prior belief is more useful in the presence of limited training data, because the pooling of information can reduce overfitting.

The energy function also includes the Lagrange multiplier $E_{\text{ortho.}}$, which is illustrated in Equation (2.8). In Equation (2.8), the functions $\mathbf{1}_{(v=w)}$ are indicators for the condition $v = w$.

$$E_{\text{ortho.}}(\boldsymbol{p}, \lambda_o) = \lambda_o \sum_{b=1}^{B} \sum_{v=1}^{V} \sum_{w=v}^{V} \left(\langle \boldsymbol{p}_{b,v}, \boldsymbol{p}_{b,w} \rangle - \mathbf{1}_{(v=w)} \right)^2 \tag{2.8}$$

$E_{\text{ortho.}}(\cdot)$ encourages orthonormality in each bin endpoint's basis. It penalizes differences from zero for dot products of pairs of vectors from the same bin, promoting orthogonality of each basis. It also penalizes differences from one for the squared ℓ_2 norm of each vector.

CHAPTER 3

LEARNING A PARAMETERIZED PRINCIPAL COMPONENT ANALYSIS MODEL

Because the energy function is composed of quadratic terms, we assume it to be locally convex. We find a local minimum in the energy function using partial derivatives of the energy function with respect to the stacked mean vector $\boldsymbol{\mu}$, the stacked basis vector $\boldsymbol{p}$, and each observation's coefficient vector $\boldsymbol{\beta}_i$. We either perform gradient descent or set the derivative to the zero vector and solved for the model component being optimized.

PPCA needs to choose optimal vectors $\boldsymbol{\mu}$, $\boldsymbol{p}$, and $\boldsymbol{\beta}$, and a derivative-based method for one of the three requires knowing or estimating the other two. In PPCA, we optimize one at a time, holding the other two as constant vectors based on their most recent estimates. After initialization, we run several cycles of optimizing the mean vectors, followed by the basis vectors, and then the coefficient vectors. We choose a pre-determined number of cycles n_c, and terminate the algorithm early if the algorithm is deemed to have converged, based on the energy. We store one previous iteration's estimates of the model components $\boldsymbol{\mu}$, $\boldsymbol{p}$, and $\boldsymbol{\beta}$, so these estimates can be treated as final if the energy increases afterward.

3.1 Learning Mean Vectors

A closed-form solution for $\hat{\boldsymbol{\mu}}$, the PPCA estimate of $\boldsymbol{\mu}$, is displayed in Equation (3.1). This uses the observation-specific matrix $\boldsymbol{W}_i$ from Equation (3.2), which is made up of bin endpoint weights $w_{b,i}$. The weight $w_{b,i}$ is equal to $w_l(\theta_i)$ if b is the lower endpoint for observation i, $w_u(\theta_i)$ if b is upper endpoint for observation i, and zero otherwise. Equation (3.1) also uses the weight-product matrix $\boldsymbol{C}_{(M),i}$ from Equation (3.3) and the matrix $\boldsymbol{R}_{(M)}$ from Equation (3.4). $\boldsymbol{R}_{(M)}$ has only three diagonals of non-zero elements, all of which are -1, 1, or 2.

$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\left(\frac{1}{n}\sum_{i=1}^{n}\left[\boldsymbol{C}_{(M),i}\right] + \frac{\lambda_{r,m}}{B-1}\boldsymbol{R}_{(M)}\right)^{-1}\sum_{i=1}^{n}\left(\boldsymbol{W}_i^T\left[\boldsymbol{x}_i - \boldsymbol{P}(\theta_i)\boldsymbol{\beta}_i\right]\right) \tag{3.1}$$

$$\boldsymbol{W}_i = \begin{bmatrix} w_{1,i}I_K & w_{2,i}I_K & \cdots & w_{B,i}I_K \end{bmatrix} \tag{3.2}$$

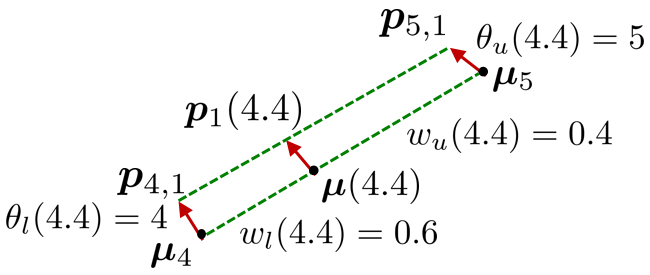

Figure 3.1: Example of Combination from Bin Endpoint Weights

$$\boldsymbol{C}_{(M),i} = \begin{bmatrix} w_{1,i}^2 I_K & w_{1,i}w_{2,i} I_K & \cdots & w_{1,i}w_{B,i} I_K \\ w_{2,i}w_{1,i} I_K & w_{2,i}^2 I_K & \cdots & w_{2,i}w_{B,i} I_K \\ \vdots & \vdots & \ddots & \vdots \\ w_{B,i}w_{1,i} I_K & w_{B,i}w_{2,i} I_K & \cdots & w_{B,i}^2 I_K \end{bmatrix} \tag{3.3}$$

$$\boldsymbol{R}_{(M)} = \begin{bmatrix} I_K & -I_K & \boldsymbol{0}_{K\times K} & \cdots & \boldsymbol{0}_{K\times K} & \boldsymbol{0}_{K\times K} & \boldsymbol{0}_{K\times K} \\ -I_K & 2I_K & -I_K & \cdots & \boldsymbol{0}_{K\times K} & \boldsymbol{0}_{K\times K} & \boldsymbol{0}_{K\times K} \\ \boldsymbol{0}_{K\times K} & -I_K & 2I_K & \cdots & \boldsymbol{0}_{K\times K} & \boldsymbol{0}_{K\times K} & \boldsymbol{0}_{K\times K} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \boldsymbol{0}_{K\times K} & \boldsymbol{0}_{K\times K} & \boldsymbol{0}_{K\times K} & \cdots & 2I_K & -I_K & \boldsymbol{0}_{K\times K} \\ \boldsymbol{0}_{K\times K} & \boldsymbol{0}_{K\times K} & \boldsymbol{0}_{K\times K} & \cdots & -I_K & 2I_K & -I_K \\ \boldsymbol{0}_{K\times K} & \boldsymbol{0}_{K\times K} & \boldsymbol{0}_{K\times K} & \cdots & \boldsymbol{0}_{K\times K} & -I_K & I_K \end{bmatrix} \tag{3.4}$$

The use of a matrix inverse or linear system solution for $\hat{\boldsymbol{\mu}}$ is either impractically slow or inaccurate for high-dimensional data such as vectorized images. For these data, we optimized the mean vectors using a gradient descent algorithm and the energy derivative from Equation (3.5).

$$\frac{\partial E}{\partial \boldsymbol{\mu}} = -\frac{2}{n}\sum_{i=1}^{n}\left[\boldsymbol{C}_{(M),i}\left(\boldsymbol{y}_i - \boldsymbol{\mu} - \boldsymbol{B}_i\boldsymbol{p}\right)\right] + \frac{2\lambda_{r,m}}{B-1}\boldsymbol{R}_{(M)}\boldsymbol{\mu} \tag{3.5}$$

Equation (3.5) uses the observation-specific coefficient matrices $\boldsymbol{B}_i$, which are defined using Equations (3.6) and (3.7). It also uses the stacked vectors $\boldsymbol{y}_i$, which stack B identical copies of an observation $\boldsymbol{x}_i$.

$$\boldsymbol{B}_i = \begin{bmatrix} \boldsymbol{B}_{(B),i} & \boldsymbol{0}_{K\times KV} & \cdots & \boldsymbol{0}_{K\times KV} \\ \boldsymbol{0}_{K\times KV} & \boldsymbol{B}_{(B),i} & \cdots & \boldsymbol{0}_{K\times KV} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}_{K\times KV} & \boldsymbol{0}_{K\times KV} & \cdots & \boldsymbol{B}_{(B),i} \end{bmatrix}_{BK\times BKV} \tag{3.6}$$

$$\boldsymbol{B}_{(B),i} = \begin{bmatrix} \beta_{i,1}I_K & \beta_{i,2}I_K & \cdots & \beta_{i,V}I_K \end{bmatrix}_{K\times KV} \tag{3.7}$$

3.1.1 Derivation of Energy Derivative with Respect to Stacked Mean Vector

To get the derivative from Equation (3.5) and the estimated mean vector from Equation (3.1), we first rearrange the energy function to allow more convenient derivatives. Using stacked vectors such as the stacked observations $\boldsymbol{y}_i$ from Equation (3.5), we can translate several vectors and matrices from Equations (2.6), (2.7), and (2.8). We can replace an observation $\boldsymbol{x}_i$ with $\boldsymbol{W}_i\boldsymbol{y}_i$, its mean $\boldsymbol{\mu}(\theta_i)$ with $\boldsymbol{W}_i\boldsymbol{\mu}$, and its coefficient-basis product $\boldsymbol{P}(\theta_i)\boldsymbol{\beta}_i$ with $\boldsymbol{W}_i\boldsymbol{B}_i\boldsymbol{p}$. This produces the alternative form of E_{data} in Equation (3.8).

$$E_{\text{data}}(\boldsymbol{\mu}, \boldsymbol{p}, \boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} \| \boldsymbol{W}_i \left(\boldsymbol{y}_i - \boldsymbol{\mu} - \boldsymbol{B}_i\boldsymbol{p} \right) \|^2 \tag{3.8}$$

For modifying $E_{\text{rough.}}$ to use stacked vectors, we need to define bin-comparison matrices $\boldsymbol{R}_{(M),b}$ and $\boldsymbol{R}_{(V),b}$ as shown in Equations (3.9) and (3.10). This allows us to use the stacked vectors $\boldsymbol{\mu}$ and $\boldsymbol{p}$, as shown in Equation (3.11).

$$\boldsymbol{R}_{(M),b} = \left[\begin{array}{cccc} \mathbf{1}_{(b=1)} I_K & \left(\mathbf{1}_{(b=2)} - \mathbf{1}_{(b=1)}\right) I_K & \cdots & \left(\mathbf{1}_{(b=B-1)} - \mathbf{1}_{(b=B-2)}\right) I_K & -\mathbf{1}_{(b=B-1)} I_K \end{array} \right] \tag{3.9}$$

$$\boldsymbol{R}_{(V),b} = \left[\begin{array}{cccc} \mathbf{1}_{(b=1)} I_{KV} & \left(\mathbf{1}_{(b=2)} - \mathbf{1}_{(b=1)}\right) I_{KV} & \cdots & \left(\mathbf{1}_{(b=B-1)} - \mathbf{1}_{(b=B-2)}\right) I_{KV} & -\mathbf{1}_{(b=B-1)} I_{KV} \end{array} \right] \tag{3.10}$$

$$E_{\text{rough.}}(\boldsymbol{\mu}, \boldsymbol{p}, \lambda_{r,m}, \lambda_{r,v}) = \frac{\lambda_{r,m}}{B-1}\sum_{b=1}^{B-1} \|\boldsymbol{R}_{(M),b}\boldsymbol{\mu}\|^2 + \frac{\lambda_{r,v}}{B-1}\sum_{b=1}^{B-1}\sum_{v=1}^{V} \|\boldsymbol{R}_{(V),b}\boldsymbol{p}\|^2 \tag{3.11}$$

In the summation for $E_{\text{ortho.}}(\cdot)$, one can expand $\left(\langle \boldsymbol{p}_{b,v}, \boldsymbol{p}_{b,w}\rangle - \mathbf{1}_{(v=w)}\right)^2$, producing the version in Equation (3.12).

$$E_{\text{ortho.}}(\boldsymbol{p}, \lambda_o) = \lambda_o \sum_{b=1}^{B}\sum_{v=1}^{V}\sum_{w=v}^{V} \langle \boldsymbol{p}_{b,v}, \boldsymbol{p}_{b,w}\rangle^2 - 2\lambda_o \sum_{b=1}^{B}\sum_{v=1}^{V}\sum_{w=v}^{V} \mathbf{1}_{(v=w)}\langle \boldsymbol{p}_{b,v}, \boldsymbol{p}_{b,w}\rangle + \lambda_o \sum_{b=1}^{B}\sum_{v=1}^{V}\sum_{w=v}^{V} \mathbf{1}^2_{(v=w)} \tag{3.12}$$

The second term from Equation (3.12) simplifies as shown in Equation (3.13).

$$-2\lambda_o \sum_{b=1}^{B}\sum_{v=1}^{V}\sum_{w=v}^{V} \mathbf{1}_{(v=w)}\langle \boldsymbol{p}_{b,v}, \boldsymbol{p}_{b,w}\rangle = -2\lambda_o \sum_{b=1}^{B}\sum_{v=1}^{V} \|\boldsymbol{p}_{b,v}\|^2 = -2\lambda_o \|\boldsymbol{p}\|^2 \tag{3.13}$$

The third term from Equation (3.12) simplifies as shown in Equation (3.14).

$$\lambda_o \sum_{b=1}^{B}\sum_{v=1}^{V}\sum_{w=v}^{V} \mathbf{1}^2_{(v=w)} = \lambda_o \sum_{b=1}^{B}\sum_{v=1}^{V}\sum_{w=v}^{V} \mathbf{1}_{(v=w)} = \lambda_o \sum_{b=1}^{B}\sum_{v=1}^{V} 1 = \lambda_o BV \tag{3.14}$$

The combined changes give us the alternate energy function in Equation (3.15)

$$E(\boldsymbol{\mu}, \boldsymbol{p}, \boldsymbol{\beta}, \lambda_{r,m}, \lambda_{r,v}, \lambda_o) = \frac{1}{n}\sum_{i=1}^{n} \| \boldsymbol{W}_i \left(\boldsymbol{y}_i - \boldsymbol{\mu} - \boldsymbol{B}_i \boldsymbol{p} \right) \|^2 + \frac{\lambda_{r,m}}{B-1} \sum_{b=1}^{B-1} \| \boldsymbol{R}_{(M),b} \boldsymbol{\mu} \|^2 + \frac{\lambda_{r,v}}{B-1} \sum_{b=1}^{B-1} \sum_{v=1}^{V} \| \boldsymbol{R}_{(V),b} \boldsymbol{p} \|^2 + \lambda_o BV - 2\lambda_o \|\boldsymbol{p}\|^2 + \lambda_o \sum_{b=1}^{B} \sum_{v=1}^{V} \sum_{w=v}^{V} \langle \boldsymbol{p}_{b,v}, \boldsymbol{p}_{b,w} \rangle^2 \quad (3.15)$$

If one translates each squared norm and each vector products into the multiplication of a row vector by a column vector, then one get the differentiation-ready version in Equation (3.16).

$$E(\boldsymbol{\mu}, \boldsymbol{p}, \boldsymbol{\beta}, \lambda_{r,m}, \lambda_{r,v}, \lambda_o) = \frac{1}{n}\sum_{i=1}^{N} \left[\boldsymbol{W}_i \left(\boldsymbol{y}_i - \boldsymbol{\mu} - \boldsymbol{B}_i \boldsymbol{p} \right) \right]^T \left[\boldsymbol{W}_i \left(\boldsymbol{y}_i - \boldsymbol{\mu} - \boldsymbol{B}_i \boldsymbol{p} \right) \right] + \frac{\lambda_{r,m}}{B-1} \sum_{b=1}^{B-1} \left[\boldsymbol{R}_{(M),b} \boldsymbol{\mu} \right]^T \left[\boldsymbol{R}_{(M),b} \boldsymbol{\mu} \right] + \frac{\lambda_{r,v}}{B-1} \sum_{b=1}^{B-1} \left[\boldsymbol{R}_{(V),b} \boldsymbol{p} \right]^T \left[\boldsymbol{R}_{(V),b} \boldsymbol{p} \right] + \lambda_o BV - 2\lambda_o \boldsymbol{p}^T \boldsymbol{p} + \lambda_o \sum_{b=1}^{B} \sum_{v=1}^{V} \sum_{w=v}^{V} \left[\boldsymbol{p}^T \boldsymbol{T}_{b,v,w} \boldsymbol{p} \right]^2 \quad (3.16)$$

If one differentiates Equation (3.16) with respect to the stacked mean vector $\boldsymbol{\mu}$, then one gets the derivative in Equation (3.17). One can then expand the transposed terms to get the derivative in Equation (3.18).

$$\frac{\partial E}{\partial \boldsymbol{\mu}} = \frac{2}{n}\sum_{1=1}^{n} \left[\boldsymbol{W}_i \left(\boldsymbol{y}_i - \boldsymbol{\mu} - \boldsymbol{B}_i \boldsymbol{p} \right) \right]^T \left(-\boldsymbol{W}_i \right) + \frac{2\lambda_{r,m}}{B-1} \sum_{b=1}^{B-1} \left[\boldsymbol{R}_{(M),b} \boldsymbol{\mu} \right]^T \boldsymbol{R}_{(M),b} \quad (3.17)$$

$$\frac{\partial E}{\partial \boldsymbol{\mu}} = -\frac{2}{n}\sum_{1=1}^{n} \left(\boldsymbol{y}_i - \boldsymbol{\mu} - \boldsymbol{B}_i \boldsymbol{p} \right)^T \boldsymbol{W}_i^T \boldsymbol{W}_i + \frac{2\lambda_{r,m}}{B-1} \sum_{b=1}^{B-1} \boldsymbol{\mu}^T \boldsymbol{R}_{(M),b}^T \boldsymbol{R}_{(M),b} \quad (3.18)$$

The matrix $\boldsymbol{C}_{(M),i}$ is defined such that $\boldsymbol{C}_{(M),i} = \boldsymbol{W}_i^T \boldsymbol{W}_i$, and $\boldsymbol{\mu}^T$ can be factored out of a summation that pertains to b, giving us the version of the derivative in (3.19).

$$\frac{\partial E}{\partial \boldsymbol{\mu}} = -\frac{2}{n}\sum_{1=1}^{n} \left(\boldsymbol{y}_i - \boldsymbol{\mu} - \boldsymbol{B}_i \boldsymbol{p} \right)^T \boldsymbol{C}_{(M),i} + \frac{2\lambda_{r,m}}{B-1} \boldsymbol{\mu}^T \sum_{b=1}^{B-1} \boldsymbol{R}_{(M),b}^T \boldsymbol{R}_{(M),b} \quad (3.19)$$

Because $\boldsymbol{R}_{(M)} = \sum_{b=1}^{B-1} \boldsymbol{R}_{(M),b}^T \boldsymbol{R}_{(M),b}$, and because the matrices $\boldsymbol{C}_{(M),i}$ and $\boldsymbol{R}_{(M)}$ are symmetric, the transpose of the right-hand side of Equation (3.19) gives us the derivative from Equation (3.5).

$$\frac{\partial E}{\partial \boldsymbol{\mu}} = -\frac{2}{n}\sum_{i=1}^{n} \left[\boldsymbol{C}_{(M),i} \left(\boldsymbol{y}_i - \boldsymbol{\mu} - \boldsymbol{B}_i \boldsymbol{p} \right) \right] + \frac{2\lambda_{r,m}}{B-1} \boldsymbol{R}_{(M)} \boldsymbol{\mu}$$

3.1.2 Derivation of Estimated Stacked Mean Vector

One can derive the estimate $\boldsymbol{\mu}$ using the derivative in Equation (3.5). One must set $\frac{\partial E}{\partial \boldsymbol{\mu}}$ to the zero vector, and then one can rearrange the derivative by addition or subtraction to get Equation (3.20). If one divides both sides by two and rearranges them so terms with $\hat{\boldsymbol{\mu}}$ are separated from terms without $\hat{\boldsymbol{\mu}}$, one gets Equation (3.21).

$$\frac{2}{n}\sum_{i=1}^{n}\left[\boldsymbol{C}_{(M),i}\left(\boldsymbol{y}_i-\hat{\boldsymbol{\mu}}-\boldsymbol{B}_i\boldsymbol{p}\right)\right]=\frac{2\lambda_{r,m}}{B-1}\boldsymbol{R}_{(M)}\hat{\boldsymbol{\mu}} \tag{3.20}$$

$$\left(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{C}_{(M),i}+\frac{\lambda_{r,m}}{B-1}\boldsymbol{R}_{(M)}\right)\hat{\boldsymbol{\mu}}=\frac{1}{n}\sum_{i=1}^{n}\left[\boldsymbol{C}_{(M),i}\left(\boldsymbol{y}_i-\boldsymbol{B}_i\boldsymbol{p}\right)\right] \tag{3.21}$$

One can simplify more by multiplying both sides by a multiplication and a conversion of the right-hand side's $\boldsymbol{C}_{(M),i}$ back into $\boldsymbol{W}_i^T\boldsymbol{W}_i$, producing Equation (3.22).

$$\hat{\boldsymbol{\mu}}=\frac{1}{n}\left(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{C}_{(M),i}+\frac{\lambda_{r,m}}{B-1}\boldsymbol{R}_{(M)}\right)^{-1}\sum_{i=1}^{n}\left[\boldsymbol{W}_i^T\left(\boldsymbol{W}_i\boldsymbol{y}_i-\boldsymbol{W}_i\boldsymbol{B}_i\boldsymbol{p}\right)\right] \tag{3.22}$$

The two conversions $\boldsymbol{W}_i\boldsymbol{y}_i=\boldsymbol{x}_i$ and $\boldsymbol{W}_i\boldsymbol{B}_i\boldsymbol{p}=\boldsymbol{P}(\theta_i)\boldsymbol{\beta}_i$ produce the solution in Equation (3.1).

$$\hat{\boldsymbol{\mu}}=\frac{1}{n}\left(\frac{1}{n}\sum_{i=1}^{n}\left[\boldsymbol{C}_{(M),i}\right]+\frac{\lambda_{r,m}}{B-1}\boldsymbol{R}_{(M)}\right)^{-1}\sum_{i=1}^{n}\left(\boldsymbol{W}_i^T\left[\boldsymbol{x}_i-\boldsymbol{P}(\theta_i)\boldsymbol{\beta}_i\right]\right)$$

3.2 Learning Basis Vectors

We only use gradient descent to optimize $\boldsymbol{p}$, because the presence of a dot product within a quadratic term creates a quartic term that prevents a closed-form solution. The derivative is in Equation (3.23), and it relies on the BKV-length vectors $\boldsymbol{b}_i$, which stack products of the weights, coefficients, and residuals.

$$\frac{\partial E}{\partial \boldsymbol{p}}=-\frac{2}{n}\sum_{i=1}^{N}\boldsymbol{b}_i+\left(\frac{\lambda_{r,v}}{B-1}\boldsymbol{R}_{(V)}-4\lambda_{r,v}\right)\boldsymbol{p}+ 2\lambda_o\sum_{b=1}^{B}\sum_{v=1}^{V_b}\sum_{w=v}^{V_b}\left[\left(\boldsymbol{T}_{b,v,w}+\boldsymbol{T}_{b,w,v}\right)\boldsymbol{p}\boldsymbol{p}^T\boldsymbol{T}_{b,w,v}\boldsymbol{p}\right] \tag{3.23}$$

$$\boldsymbol{b}_i=\begin{bmatrix} w_{1,i}\beta_{i,1}\left[\boldsymbol{x}_i-\boldsymbol{\mu}(\theta_i)-\boldsymbol{P}(\theta_i)\boldsymbol{\beta}_i\right] \\ w_{1,i}\beta_{i,2}\left[\boldsymbol{x}_i-\boldsymbol{\mu}(\theta_i)-\boldsymbol{P}(\theta_i)\boldsymbol{\beta}_i\right] \\ \vdots \\ w_{B,i}\beta_{i,V-1}\left[\boldsymbol{x}_i-\boldsymbol{\mu}(\theta_i)-\boldsymbol{P}(\theta_i)\boldsymbol{\beta}_i\right] \\ w_{B,i}\beta_{i,V}\left[\boldsymbol{x}_i-\boldsymbol{\mu}(\theta_i)-\boldsymbol{P}(\theta_i)\boldsymbol{\beta}_i\right] \end{bmatrix} \tag{3.24}$$

Equation (3.23) also uses the transition-like matrix $\boldsymbol{T}_{b,v,w}$ from Equation (3.25) and the bin-comparison matrix $\boldsymbol{R}_{(V)}$ from Equation (3.26). $\boldsymbol{T}_{b,v,w}$, if multiplied by $\boldsymbol{p}$, will zero out all $\boldsymbol{p}_{b,w^*}$ except for $\boldsymbol{p}_{b,w}$, which gets moved to the appropriate spot for $\boldsymbol{p}_{b,v}$. In its definition, the functions $\mathbf{1}_{(\cdot)}$ are indicator functions for the events within the parentheses. $\boldsymbol{R}_{(V)}$ is a larger version of the matrix $\boldsymbol{R}_{(M)}$ used for the means.

$$\boldsymbol{T}_{b,v,w} = \begin{bmatrix} \mathbf{1}_{(b=1\cap v=1\cap w=1)}I_K & \mathbf{1}_{(b=1\cap v=1\cap w=2)}I_K & \cdots & \mathbf{0}_{K\times K} \\ \mathbf{1}_{(b=1\cap v=2\cap w=1)}I_K & \mathbf{1}_{(b=1\cap v=2\cap w=2)}I_K & \cdots & \mathbf{0}_{K\times K} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{K\times K} & \mathbf{0}_{K\times K} & \cdots & \mathbf{1}_{(b=B\cap v=V\cap w=V)}I_K \end{bmatrix} \tag{3.25}$$

$$\boldsymbol{R}_{(V)} = \begin{bmatrix} I_{KV} & -I_{KV} & \mathbf{0}_{KV\times KV} & \cdots & \mathbf{0}_{KV\times KV} & \mathbf{0}_{KV\times KV} & \mathbf{0}_{KV\times KV} \\ -I_{KV} & 2I_{KV} & -I_{KV} & \cdots & \mathbf{0}_{KV\times KV} & \mathbf{0}_{KV\times KV} & \mathbf{0}_{KV\times KV} \\ \mathbf{0}_{KV\times KV} & -I_{KV} & 2I_{KV} & \cdots & \mathbf{0}_{KV\times KV} & \mathbf{0}_{KV\times KV} & \mathbf{0}_{KV\times KV} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0}_{KV\times KV} & \mathbf{0}_{KV\times KV} & \mathbf{0}_{KV\times KV} & \cdots & 2I_{KV} & -I_{KV} & \mathbf{0}_{KV\times KV} \\ \mathbf{0}_{KV\times KV} & \mathbf{0}_{KV\times KV} & \mathbf{0}_{KV\times KV} & \cdots & -I_{KV} & 2I_{KV} & -I_{KV} \\ \mathbf{0}_{KV\times KV} & \mathbf{0}_{KV\times KV} & \mathbf{0}_{KV\times KV} & \cdots & \mathbf{0}_{KV\times KV} & -I_{KV} & I_{KV} \end{bmatrix} \tag{3.26}$$

The gradient descent algorithm has a soft constraint for orthonormal bases, but we implement a hard constraint for normality as well. After the gradient descent algorithm for $\boldsymbol{p}$ completes, we rescale each basis vector $\boldsymbol{p}_{b,v}$ to have a unit norm. We cannot similarly force orthogonality without undoing the gradient descent algorithm's attempts to enforce smoothness.

3.2.1 Derivation of Energy Derivative with Respect to Stacked Basis Vector

To get the derivative from Equation (3.23), we start with the rearranged energy function from Equation (3.16). If we differentiate with respect to $\boldsymbol{p}$, we get Equation (3.27).

$$\begin{aligned} \frac{\partial E}{\partial \boldsymbol{p}} = \frac{2}{n}\sum_{i=1}^{N}\left[\boldsymbol{W}_i\left(\boldsymbol{y}_i - \boldsymbol{\mu} - \boldsymbol{B}_i\boldsymbol{p}\right)\right]^T\left(-\boldsymbol{W}_i\boldsymbol{B}_i\right) + \frac{\lambda_{r,v}}{B-1}\sum_{b=1}^{B-1}\left[\boldsymbol{R}_{(V),b}\boldsymbol{p}\right]^T\boldsymbol{R}_{(V),b} \\ - 4\lambda_o\boldsymbol{p}^T + 2\lambda_o\sum_{b=1}^{B}\sum_{v=1}^{V}\sum_{w=v}^{V}\left[\boldsymbol{p}^T\boldsymbol{T}_{b,v,w}\boldsymbol{p}\boldsymbol{p}^T\left(\boldsymbol{T}_{b,v,w} + \boldsymbol{T}_{b,w,v}\right)\right] \end{aligned} \tag{3.27}$$

Expanding the transposes and taking the transpose of the right-hand side of Equation (3.27), we then get Equation (3.28).

$$\frac{\partial E}{\partial \boldsymbol{p}} = -\frac{2}{n}\sum_{i=1}^{N}\left[\boldsymbol{B}_i^T\boldsymbol{W}_i^T\boldsymbol{W}_i\left(\boldsymbol{y}_i - \boldsymbol{\mu} - \boldsymbol{B}_i\boldsymbol{p}\right)\right] + \frac{\lambda_{r,v}}{B-1}\left[\sum_{b=1}^{B-1}\boldsymbol{R}_{(V),b}^T\boldsymbol{R}_{(V),b}\right]\boldsymbol{p} - 4\lambda_o\boldsymbol{p} + 2\lambda_o\sum_{b=1}^{B}\sum_{v=1}^{V}\sum_{w=v}^{V}\left[\left(\boldsymbol{T}_{b,v,w} + \boldsymbol{T}_{b,w,v}\right)\boldsymbol{p}\boldsymbol{p}^T\boldsymbol{T}_{b,v,w}^T\boldsymbol{p}\right] \quad (3.28)$$

Because $\boldsymbol{R}_{(V)} = \sum_{b=1}^{B-1}\boldsymbol{R}_{(V),b}^T\boldsymbol{R}_{(V),b}$, this simplifies to Equation (3.29).

$$\frac{\partial E}{\partial \boldsymbol{p}} = -\frac{2}{n}\sum_{i=1}^{N}\left[\boldsymbol{B}_i^T\boldsymbol{W}_i^T\left(\boldsymbol{W}_i\boldsymbol{y}_i - \boldsymbol{W}_i\boldsymbol{\mu} - \boldsymbol{W}_i\boldsymbol{B}_i\boldsymbol{p}\right)\right] + \frac{\lambda_{r,v}}{B-1}\boldsymbol{R}_{(V)}\boldsymbol{p} - 4\lambda_o\boldsymbol{p} + 2\lambda_o\sum_{b=1}^{B}\sum_{v=1}^{V}\sum_{w=v}^{V}\left[\left(\boldsymbol{T}_{b,v,w} + \boldsymbol{T}_{b,w,v}\right)\boldsymbol{p}\boldsymbol{p}^T\boldsymbol{T}_{b,v,w}^T\boldsymbol{p}\right] \quad (3.29)$$

Using $\boldsymbol{x}_i = \boldsymbol{W}_i\boldsymbol{y}_i$, $\boldsymbol{\mu}(\theta_i) = \boldsymbol{W}_i\boldsymbol{\mu}$, and $\boldsymbol{P}(\theta_i)\boldsymbol{\beta}_i = \boldsymbol{W}_i\boldsymbol{B}_i\boldsymbol{p}$, we get Equation (3.30).

$$\frac{\partial E}{\partial \boldsymbol{p}} = -\frac{2}{n}\sum_{i=1}^{N}\left[\boldsymbol{B}_i^T\boldsymbol{W}_i^T\left(\boldsymbol{x}_i - \boldsymbol{\mu}(\theta_i) - \boldsymbol{P}(\theta_i)\boldsymbol{\beta}_i\right)\right] + \left(\frac{\lambda_{r,v}}{B-1}\boldsymbol{R}_{(V)} - 4\lambda_o\right)\boldsymbol{p} + 2\lambda_o\sum_{b=1}^{B}\sum_{v=1}^{V}\sum_{w=v}^{V}\left[\left(\boldsymbol{T}_{b,v,w} + \boldsymbol{T}_{b,w,v}\right)\boldsymbol{p}\boldsymbol{p}^T\boldsymbol{T}_{b,v,w}^T\boldsymbol{p}\right] \quad (3.30)$$

The vectors $\boldsymbol{b}_i$ are defined such that they can substitute for $\boldsymbol{B}_i^T\boldsymbol{W}_i^T\left(\boldsymbol{x}_i - \boldsymbol{\mu}(\theta_i) - \boldsymbol{P}(\theta_i)\boldsymbol{\beta}_i\right)$, producing the derivative from Equation (3.23).

$$\frac{\partial E}{\partial \boldsymbol{P}} = -\frac{2}{n}\sum_{i=1}^{N}\boldsymbol{b}_i + \left(\frac{\lambda_{r,v}}{B-1}\boldsymbol{R}_{(V)} - 4\lambda_{r,v}\right)\boldsymbol{P} + 2\lambda_o\sum_{b=1}^{B}\sum_{v=1}^{V_b}\sum_{w=v}^{V_b}\left[\left(\boldsymbol{T}_{b,v,w} + \boldsymbol{T}_{b,w,v}\right)\boldsymbol{P}\boldsymbol{P}^T\boldsymbol{T}_{b,w,v}\boldsymbol{P}\right]$$

3.3 Learning Coefficient Vectors

If one differentiates the energy function with respect to a single observation's coefficient vector $\boldsymbol{\beta}_i$ and sets this derivative equal to the zero vector, one can obtain the estimate $\hat{\boldsymbol{\beta}}_i$ below for a coefficient vector $\boldsymbol{\beta}_i$.

$$\hat{\boldsymbol{\beta}}_i = \left[\boldsymbol{P}(\theta_i)\right]^{-1}\left[\boldsymbol{x}_i - \boldsymbol{\mu}(\theta_i)\right] \quad (3.31)$$

This inverse is applied to a much smaller matrix than that inverted to find $\hat{\boldsymbol{\mu}}$, so we use a linear system solution to obtain $\hat{\boldsymbol{\beta}}_i$, even with high-dimensional data.

3.3.1 Derivation of Estimated Coefficient Vector

Because the solution is for a single observation's coefficient vector $\boldsymbol{\beta}_i$ instead of the stacked vector $\boldsymbol{\beta}$, it is more convenient to work from the original form of $E_{\text{data}}(\cdot)$ from Equation (2.6), except with squared norms converted to products of row vectors and column vectors. This version is shown in Equation (3.32).

$$E_{\text{data}}(\boldsymbol{\mu}, \boldsymbol{p}, \boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^{n} \left[\boldsymbol{x}_i - \boldsymbol{\mu}(\theta_i) - \boldsymbol{P}(\theta_i)\boldsymbol{\beta}_i\right]^T \left[\boldsymbol{x}_i - \boldsymbol{\mu}(\theta_i) - \boldsymbol{P}(\theta_i)\boldsymbol{\beta}_i\right] \tag{3.32}$$

The energy components $E_{\text{rough.}}$ and $E_{\text{ortho.}}$ do not depend on $\boldsymbol{\beta}_i$, so one can differentiate Equation (3.32) to get $\frac{\partial E}{\partial \boldsymbol{\beta}_i}$.

$$\frac{\partial E}{\partial \boldsymbol{\beta}_i} = \frac{2}{n} \left[\boldsymbol{x}_i - \boldsymbol{\mu}(\theta_i) - \boldsymbol{P}(\theta_i)\boldsymbol{\beta}_i\right]^T \left(-\boldsymbol{P}(\theta_i)\right) \tag{3.33}$$

Setting $\frac{\partial E}{\partial \boldsymbol{\beta}_i}$ to the zero vector gives us an equation for $\hat{\boldsymbol{\beta}}_i$.

$$\boldsymbol{0} = \left[\boldsymbol{x}_i - \boldsymbol{\mu}(\theta_i) - \boldsymbol{P}(\theta_i)\hat{\boldsymbol{\beta}}_i\right]^T \boldsymbol{P}(\theta_i) \tag{3.34}$$

If we transpose each side and isolate the term including $\hat{\boldsymbol{\beta}}_i$, we produce Equation (3.35).

$$\boldsymbol{P}(\theta_i)^T \boldsymbol{P}(\theta_i)\hat{\boldsymbol{\beta}}_i = \boldsymbol{P}(\theta_i)^T \left[\boldsymbol{x}_i - \boldsymbol{\mu}(\theta_i)\right] \tag{3.35}$$

Multiplying each side by $\left[\boldsymbol{P}(\theta_i)^T\right]^{-1}$ gives us Equation (3.36).

$$\boldsymbol{P}(\theta_i)\hat{\boldsymbol{\beta}}_i = \boldsymbol{x}_i - \boldsymbol{\mu}(\theta_i) \tag{3.36}$$

Multiplying again by $\boldsymbol{P}(\theta_i)^{-1}$ results in the solution in Equation (3.31).

$$\hat{\boldsymbol{\beta}}_i = \left[\boldsymbol{P}(\theta_i)\right]^{-1} \left[\boldsymbol{x}_i - \boldsymbol{\mu}(\theta_i)\right]$$

3.4 Initialization

PPCA finds an appropriate local minimum within the energy function, so an appropriate initialization is important for finding a local minimum that can perform similarly to the global minimum. We initialize PPCA using a procedure similar to IPCA, which runs PCA on groups made by binning the parameter θ. We calculate initial mean vectors $\hat{\boldsymbol{\mu}}_{(0),b}$ using Equation (3.37), which is like a weighted version of the mean calculation from IPCA.

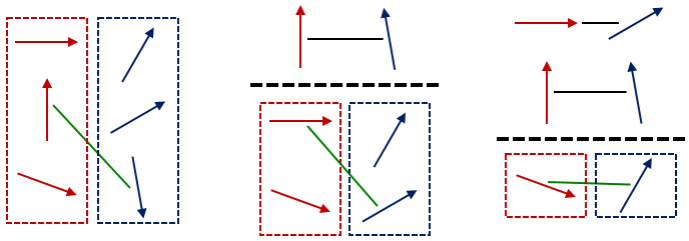

Figure 3.2: Example Reordering and Sign Change of Initial Basis Vectors

$$\hat{\boldsymbol{\mu}}_{(0),b} = \frac{\sum_{i=1}^{n} w_{b,i}\boldsymbol{x}_i}{\sum_{i=1}^{n} w_{b,i}} \tag{3.37}$$

To find the initial basis vectors, we choose overlapping subsets of the observations $\boldsymbol{x}_i$ and assign one subset to each bin endpoint b. The included observations are all with weight values $w_{b,i}$ above a given threshold such as 0.001. We run PCA on each of these subsets, except that we use the means $\hat{\boldsymbol{\mu}}_{(0),b}$ instead of recalculating the means based on the subsets of $\boldsymbol{x}_i$. We then reorder these PCA basis vectors to promote smoothness, using a greedy algorithm. One can start with the first bin endpoint's basis as the first reference basis, and reorder the bases from the second until the last bin endpoint. Alternatively, one can make the last bin endpoint's basis the first reference basis, and reorder the bases from the second-to-last until the first bin endpoint.

For each pair of bin endpoints, one first calculates the absolute values of the dot products between each pair of basis vectors using one from each endpoint. The two vectors with the highest absolute value of the dot product are paired, and the sign is inverted for the vector from the basis to reorder if the dot product is negative. This procedure continues, each time only using vectors that are not in any pairs, until all vectors in the reference basis have been paired. If any vectors remain in the basis to reorder, they are assigned to any unused locations. The basis just reordered then becomes the reference basis, the next basis in the order is assigned to be reordered, and the procedure continues until all bases except the original reference have been reordered. The coefficients can then be initialized from the initial mean and basis vectors using Equation (3.31).

3.5 Putting it all together

The complete PPCA training algorithm is summarized in Algorithm 1.

Algorithm 1 PPCA training algorithm

for $i = 1$ to n **do**
    set $w_l(\theta_i)$ and $w_u(\theta_i)$ using Equation (2.1)
end for
for $b = 1$ to B **do**
    initialize $\hat{\boldsymbol{\mu}}_b$ using Equation (3.37)
    initialize vectors $\hat{\boldsymbol{p}}_{b,v}$ using PCA on examples with $w_{b,i} > \epsilon$
    rearrange vectors $\hat{\boldsymbol{p}}_{b,v}$ for same b and switch signs if necessary
end for
for $i = 1$ to n **do**
    initialize $\hat{\boldsymbol{\beta}}_i$ using Equation (3.31)
end for
find E_0 using Equation (2.4)
for $c = 1$ to n_c **do**
    update $\hat{\boldsymbol{\mu}}$ using Equation (3.1) or gradient descent with Equation (3.5)
    update $\hat{\boldsymbol{p}}$ using gradient descent with Equation (3.23)
    for $i = 1$ to n **do**
        update $\hat{\boldsymbol{\beta}}_i$ using Equation (3.31)
    end for
    find E_c using Equation (2.4)
    if $E_c > E_{c-1}$ **then**
        break
    end if
end for

3.6 Tuning of Parameters

The energy must be tracked, so one can use its path to choose the number of overall cycles n_c, the learning rates (α_m for means, α_v for bases), the number of iterations with those learning rates (n_m for means, n_v for bases), and the non-orthonormality penalty coefficient λ_o. If one wants to choose appropriate roughness penalty coefficients ($\lambda_{r,m}$ for means, $\lambda_{r,v}$ for bases), then one should tune them using a validation set selected randomly from the training examples. Typically, λ_o should be much larger than $\lambda_{r,v}$. However, α_v must decrease as λ_o increases, so an excessively large λ_o leads to unnecessary increases in run-time.

CHAPTER 4

MODIFICATIONS AND GENERALIZATIONS OF PARAMETERIZED PRINCIPAL COMPONENT ANALYSIS FOR REAL APPLICATIONS

This chapter details two modifications for generalizations that allow dimensions to vary with the PPCA parameter. The first is for the dimension of the manifold, and the second is for the observations.

4.1 Generalization to Varied Manifold Dimension

For some applications, the manifold dimension can vary with the parameter θ. In Section 5.2, we present such a case, in which higher values of θ are thought to require a more complex representation (with a larger coefficient vector $\boldsymbol{\beta}_i$). In this case, each bin endpoint b would have V_b basis vectors, and V would be set to the largest V_b. In the stacked vector $\boldsymbol{p}$, one would still allocate V basis vectors for each bin endpoint, but one would set $\boldsymbol{p}_{b,v}$ to be a zero vector if $v > V_b$.

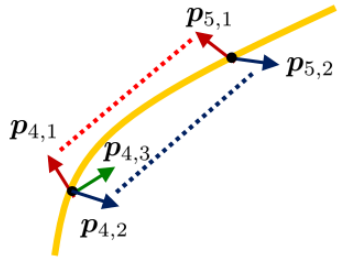

Figure 4.1: Example of Varied Number of Basis Vectors

If one has differently-sized bases, the energy component $E_{\text{rough.}}$ needs to follow Equation (4.1) instead of Equation (2.7). Also, the energy component $E_{\text{ortho.}}$ needs to follow Equation (4.2) instead of Equation (2.8).

$$E_{\text{rough.}}(\boldsymbol{\mu}, \boldsymbol{p}, \lambda_{r,m}, \lambda_{r,v}) = \frac{\lambda_{r,m}}{B-1} \sum_{b=1}^{B-1} \|\boldsymbol{\mu}_b - \boldsymbol{\mu}_{b+1}\|^2 + \frac{\lambda_{r,v}}{B-1} \sum_{b=1}^{B-1} \sum_{v=1}^{\min(V_b, V_{b+1})} \|\boldsymbol{p}_{b,v} - \boldsymbol{p}_{b+1,v}\|^2 \quad (4.1)$$

$$E_{\text{ortho.}}(\boldsymbol{p}, \lambda_o) = \lambda_o \sum_{b=1}^{B} \sum_{v=1}^{V_b} \sum_{w=v}^{V_b} \left(\langle \boldsymbol{p}_{b,v}, \boldsymbol{p}_{b,w} \rangle - \mathbf{1}_{(v=w)} \right)^2 \tag{4.2}$$

The only three changes to these two equations are to the upper boundaries of summations. In Equation (4.1), the third summation ends at $\min(V_b, V_{b+1})$ rather than at V. This is intended so PPCA only enforces similarity between the corresponding basis vectors for adjacent bin endpoints if the basis vectors exist for both. In Equation (4.2), the second and third summations end at V_b instead of at V. This is because there are no vectors beyond vector V_b upon which to enforce orthonormality.

In Section 3.4, we detailed a procedure of rearranging initial basis vectors produced by PCA. If the number of basis vectors is either non-decreasing or non-increasing with respect to the bin endpoint number, then this procedure still works, with one modification. If all bin endpoints use V basis vectors, the user has the choice of reordering from the second until the last bin endpoint, or from the second-to-last until the first bin endpoint. However, if the first bin endpoint's basis is smaller than the last bin endpoint's basis, the reordering procedure must go from the second basis to the last. If the last bin endpoint's basis is smaller than the first bin endpoint's basis, the reordering procedure must go from the second-to-last basis to the first. If the number of basis vectors both increases and decreases when going from the first to the last bin endpoint, then the reordering must be done more manually.

4.2 Generalization to Varied Manifold Ambient Space

Applications also exist in which the manifold ambient space varies with the parameter θ. Section 5.4 demonstrates an example of this sort, using face images. In these data, certain pixels may be considered outside the face shape for a given face. Like the observations, the mean and basis vectors for bin endpoint b may not use all K elements. As shown in Figure 4.2, we want the mean vectors $\boldsymbol{\mu}_b = (\mu_{b,1}, \ldots, \mu_{b,K})^T$ to have similarity enforced between elements $\mu_{b,k}$ and $\mu_{b+1,k}$ only if element k is relevant for both bin endpoint b and bin endpoint $b+1$. So, we create the indicator variables $m_{b,k}$ which equal one if element k is included for bin endpoint b, and zero otherwise. From these, we can construct matrices $\boldsymbol{M}_{(1),b}$ that can adjust mean vectors $\boldsymbol{\mu}_b$ or basis vectors $\boldsymbol{p}_{b,v}$, setting unused elements to zero. We also construct matrices $\boldsymbol{M}_{(R1),b}$ as shown in Equation (4.4), which

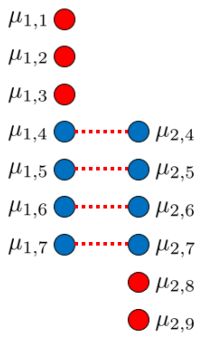

Figure 4.2: Example of Mean Vectors from Two Endpoints of Same Bin, in Scenario with Varied Ambient Space of Manifold

can similarly adjust larger vectors.

$$\boldsymbol{M}_{(1),b} = \begin{bmatrix} m_{b,1} & 0 & \cdots & 0 \\ 0 & m_{b,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & m_{b,K} \end{bmatrix}_{K \times K} \tag{4.3}$$

$$\boldsymbol{M}_{(R1),b} = \begin{bmatrix} \boldsymbol{M}_{(1),b} & \boldsymbol{0}_{K\times K} & \cdots & \boldsymbol{0}_{K\times K} \\ \boldsymbol{0}_{K\times K} & \boldsymbol{M}_{(1),b} & \cdots & \boldsymbol{0}_{K\times K} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}_{K\times K} & \boldsymbol{0}_{K\times K} & \cdots & \boldsymbol{M}_{(1),b} \end{bmatrix}_{KV \times KV} \tag{4.4}$$

For each bin endpoint b, one would then calculate $\boldsymbol{M}_{(2),b} = \boldsymbol{M}_{(1),b}\boldsymbol{M}_{(1),b+1}$ and $\boldsymbol{M}_{(R2),b} = \boldsymbol{M}_{(R1),b}\boldsymbol{M}_{(R1),b+1}$. $\boldsymbol{M}_{(2),b}$ can then be used to adjust the energy component $E_{\text{rough.}}$ as shown in Equation (4.5). The only adjustments made relative to Equation (2.7) are two additions of $\boldsymbol{M}_{(2),b}$.

$$E_{\text{rough.}}(\boldsymbol{\mu}, \boldsymbol{p}, \lambda_{r,m}, \lambda_{r,v}) = \frac{\lambda_{r,m}}{B-1} \sum_{b=1}^{B-1} \| \boldsymbol{M}_{(2),b} \left(\boldsymbol{\mu}_b - \boldsymbol{\mu}_{b+1} \right) \|^2 + \frac{\lambda_{r,v}}{B-1} \sum_{b=1}^{B-1} \sum_{v=1}^{V} \| \boldsymbol{M}_{(2),b} \left(\boldsymbol{p}_{b,v} - \boldsymbol{p}_{b+1,v} \right) \|^2 \tag{4.5}$$

The matrices $\boldsymbol{R}_{(M)}$ and $\boldsymbol{R}_{(V)}$ from Equations (3.1), (3.5), and (3.23) must be modified as well. These each still have three diagonals that can have non-zero elements, but these diagonals incorporate the indicator variables $m_{b,k}$ and thus can have zeros. The modified versions, shown in Equations (4.6) and (4.7), only differ from Equations (3.4) and (3.26) by including $\boldsymbol{M}_{(2),b}$ and

$\boldsymbol{M}_{(R2),b}$, respectively, instead of identity matrices of the same size.

$$\boldsymbol{R}_{(M)} = \begin{bmatrix} \boldsymbol{M}_{(2),1} & -\boldsymbol{M}_{(2),1} & \cdots & \boldsymbol{0}_{K\times K} & \boldsymbol{0}_{K\times K} \\ -\boldsymbol{M}_{(2),1} & \boldsymbol{M}_{(2),1}+\boldsymbol{M}_{(2),2} & \cdots & \boldsymbol{0}_{K\times K} & \boldsymbol{0}_{K\times K} \\ \boldsymbol{0}_{K\times K} & -\boldsymbol{M}_{(3),2} & \cdots & \boldsymbol{0}_{K\times K} & \boldsymbol{0}_{K\times K} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{0}_{K\times K} & \boldsymbol{0}_{K\times K} & \cdots & -\boldsymbol{M}_{(2),B-2} & \boldsymbol{0}_{K\times K} \\ \boldsymbol{0}_{K\times K} & \boldsymbol{0}_{K\times K} & \cdots & \boldsymbol{M}_{(2),B-2}+\boldsymbol{M}_{(2),B-1} & -\boldsymbol{M}_{(2),B-1} \\ \boldsymbol{0}_{K\times K} & \boldsymbol{0}_{K\times K} & \cdots & -\boldsymbol{M}_{(2),B-1} & \boldsymbol{M}_{(2),B-1} \end{bmatrix} \tag{4.6}$$

$$\boldsymbol{R}_{(V)} = \begin{bmatrix} \boldsymbol{M}_{(R2),1} & -\boldsymbol{M}_{(R2),1} & \cdots & \boldsymbol{0}_{KV\times KV} & \boldsymbol{0}_{KV\times KV} \\ -\boldsymbol{M}_{(R2),1} & \boldsymbol{M}_{(R2),1}+\boldsymbol{M}_{(R2),2} & \cdots & \boldsymbol{0}_{KV\times KV} & \boldsymbol{0}_{KV\times KV} \\ \boldsymbol{0}_{KV\times KV} & -\boldsymbol{M}_{(R2),2} & \cdots & \boldsymbol{0}_{KV\times KV} & \boldsymbol{0}_{KV\times KV} \\ \vdots & \vdots & \ddots & \vdots & \\ \boldsymbol{0}_{KV\times KV} & \boldsymbol{0}_{KV\times KV} & \cdots & -\boldsymbol{M}_{(R2),B-2} & \boldsymbol{0}_{KV\times KV} \\ \boldsymbol{0}_{KV\times KV} & \boldsymbol{0}_{KV\times KV} & \cdots & \boldsymbol{M}_{(R2),B-2}+\boldsymbol{M}_{(R2),B-1} & -\boldsymbol{M}_{(R2),B-1} \\ \boldsymbol{0}_{KV\times KV} & \boldsymbol{0}_{KV\times KV} & \cdots & -\boldsymbol{M}_{(R2),B-1} & \boldsymbol{M}_{(R2),B-1} \end{bmatrix} \tag{4.7}$$

CHAPTER 5

EXPERIMENTS WITH PARAMETERIZED PRINCIPAL COMPONENT ANALYSIS

We evaluated PPCA on four datasets. One had data created from known parameters, so we could ensure that PPCA could recover these true parameters well. The other three were for applications of PPCA to real data: shapes for lymph nodes of varied sizes, appearances for faces of varied blurriness, and appearances for faces of varied yaw rotation.

5.1 Simulation Experiments

First, we tested PPCA's ability to recover a true model, using three-dimensional data created from known mean and basis vectors. These were based on smooth functions of a known parameter θ, defined on the range from 0 to 360. We used 45 observations with $\theta = 4, 12, 20, \ldots, 356$. The data were based on two basis vectors and on coefficients drawn independently from a $U(-1, 1)$ distribution. We also added random noise to each element, using a $U(-1.5, 1.5)$ distribution. The formulas for the true mean vectors $\boldsymbol{\mu}(\theta)$ and true basis vectors $\boldsymbol{p}_1(\theta)$ and $\boldsymbol{p}_2(\theta)$ were as follows.

$$\boldsymbol{\mu}(\theta) = \left\{ \sin\left(\frac{7\pi\theta}{720}\right), -\frac{91\theta}{1800} + 8, \sin\left(\frac{7\pi\theta}{576} + 0.6\right) \right\}^T \tag{5.1}$$

$$\boldsymbol{p}_1(\theta) = \left\{ \sin\left(\frac{7\pi\theta}{1080} + 0.4\right), \tan\left(\frac{7\pi\theta}{4860} - 0.8\right), \frac{49\theta}{1800} - 1.1 \right\}^T \tag{5.2}$$

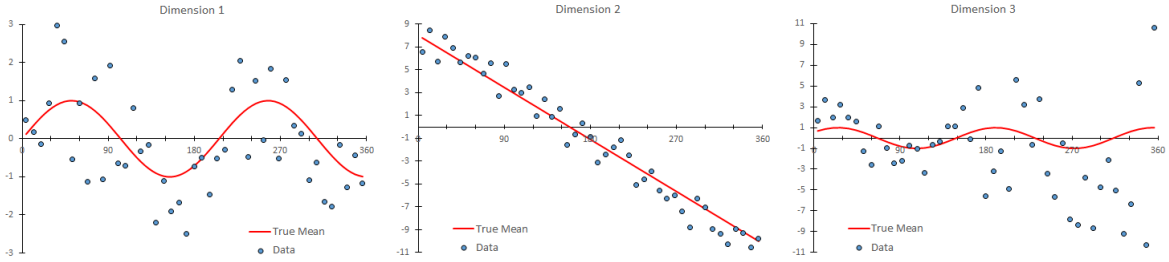

Figure 5.1: Artificial Data by Element, Compared with True Mean Function

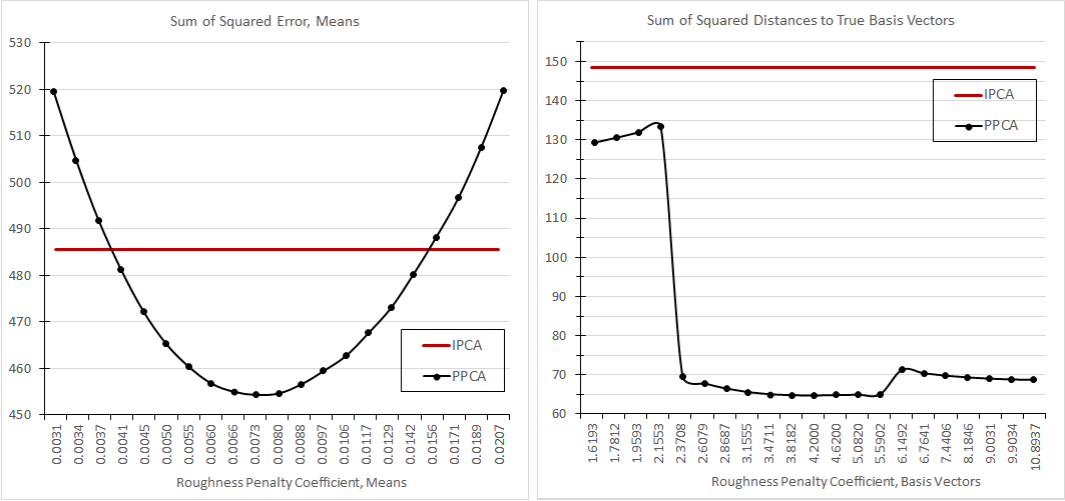

Figure 5.2: Sum of Squared ℓ_2 Errors for Elements of Artificial Data's Mean Vectors (Left) and Sum of Squared ℓ_2 Distances to Artificial Data's True Basis Vectors (Right)

$$\boldsymbol{p}_2(\theta) = \left\{\cos\left(\frac{7\pi\theta}{972}\right), \cos\left(\frac{7\pi\theta}{576} - 0.4\right), \frac{7\theta}{600} + 1.4\right\}^T \tag{5.3}$$

We divided the acceptable parameter range into 14 equally-sized bins. Because the data and model were small (three dimensions and two basis vectors), we could use the analytical solution to calculate the mean vectors and only needed gradient descent for the bases. We used various roughness penalty coefficients, but always used $\lambda_o = 20$, $n_c = 1000$, $n_v = 500$, and $\alpha_v = 0.0001$.

Figure 5.2 shows the sum of squared ℓ_2 norms for the error in PPCA's and IPCA's estimates of the mean vector, compared to the true mean vectors $\boldsymbol{\mu}(\theta)$. This uses various $\lambda_{r,m}$ but fixes $\lambda_{r,v} = 4.2$. For the bases, we could not use a simple ℓ_2 error because a proper recovery could have the same span, but different vectors and coefficients. We instead measured the ℓ_2 norms of the normal vectors from the planes created by the recovered basis vectors to each of the true basis vectors. Figure 5.2 shows the sums (across the observations and two true vectors) of the squares of these ℓ_2 distances. This uses various $\lambda_{r,v}$ but fixes $\lambda_{r,m} = 0.008$. In these simulations, PPCA outperformed IPCA if using a roughness penalty coefficient for means close to 0.008, and using any of the tested roughness penalty coefficients for basis vectors.

5.2 Lymph Node Segmentation

Lymph nodes are organs that are part of the circulatory system and the immune system, which are important for the diagnosis and treatment of cancer and other medical conditions. For cancer patients, one may want a segmentation for targeting radiation or for volume estimates. Lymph node sizes generally increase with the onset of cancer, and decrease as treatment succeeds, so volume estimates are used to assess the efficacy of treatment. Generally, radiologists use 3D computed tomography (CT) to assess lymph nodes, and lymph nodes tend to have spherical, elliptical, and bean-like shapes. However, their shape can become more complicated as their size increases. Barbu et al. (2012) demonstrated a model for representing lymph nodes by the lengths of radii, which extend in 162 pre-determined directions from the lymph node's center [1]. We reduce this 162-dimensional representation of a lymph node's shape even further, using PPCA and IPCA with 10 or fewer dimensions.

We had a dataset available, in which an experienced radiologist had manually segmented 592 lymph nodes from various patients treated at the National Institutes of Health Clinical Center. We used only the 397 lymph nodes for which the 162-dimensional model was most appropriate. We eliminated 182 lymph nodes which were part of conglomerates of lymph nodes, and 16 for which the radial model's Sørensen-Dice coefficient was less than 0.8. We then randomly selected 79 lymph nodes to be in the test set, and assigned the remaining 318 lymph nodes to the training set.

The parameter of interest for this application was the lymph node's diameter, because lymph nodes' sizes are related to the types of shapes they can take. We used an estimated diameter based on the 162 modeled radii. We divided the lymph nodes into bins of 6-12, 12-18, 18-24, and 24-43 millimeters. The dataset had 55, 156, 77, and 30 training examples per bin and 16, 39, 17, and 7 test examples per bin, in increasing order of bin. We used 3, 5, 7, and 9 basis vectors, respectively, for the IPCA bins, and 2, 4, 6, 8, and 10 basis vectors for the PPCA bin endpoints. These were chosen to allow more shape complexity for the larger lymph nodes, and to ensure that PPCA and IPCA could be compared fairly, with the same average dimensions used per bin.

We evaluated PPCA and IPCA after fitting the models using different numbers of training examples per bin, from 2 to 30. These smaller training sets were chosen randomly from the full training set. All smaller training sets were subsets of the larger training sets, to ensure a more valid comparison of the effect of the training set size. Each time, we calculated the root mean squared

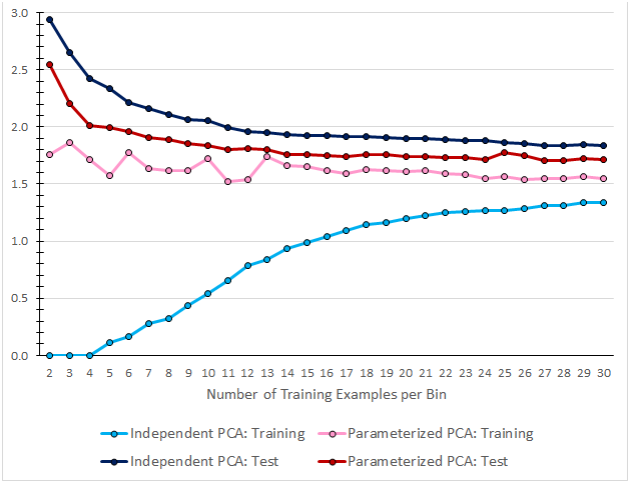

Figure 5.3: Mean RMSE for Projection of Radial Representation of Lymph Nodes, Evaluated on Training and Test Sets Using Varied Numbers of Training Examples

error (RMSE) of the projection of each lymph node's 162-dimensional vector of radii, and then found the mean across the lymph nodes of the training or test set. For PPCA, we used gradient descent algorithms for both the mean and basis vectors. We used $\lambda_{r,m} = 0.007$, $\lambda_{r,v} = 30{,}000$, $\lambda_o = 10^7$, $n_c = 200$, $n_m = 100$, $n_v = 100$, $\alpha_m = 0.01$, and $\alpha_v = 10^{-8}$.

Figure 5.3 shows that IPCA overfit the data compared to PPCA, particularly for smaller training sets. For all tested sizes, PPCA had noticeably higher error than IPCA when projecting the training set, but noticeably lower error than IPCA did when projecting the test set.

5.3 Facial Images with Blur

Modeling images of human faces in photos or video frames is useful for creating novel images that satisfy the constraints of realistic faces, such as for animation. It can also modify a known face to show poses or expressions not in available images. Face models can be used as generative face detectors, too, with applications such as auto-focusing a camera or detecting intruders on a security camera. Face modeling can also aid face recognition, by aligning the images to be recognized or by providing the lower-dimensional representation that can be matched against a dictionary.

Variations in the conditions (such as illumination) of images present challenges for face models. One such variation is the blurriness of photographs. Digital cameras, particularly those in many mobile phones, are used frequently to produce photos that may not be appropriately sharp. Even

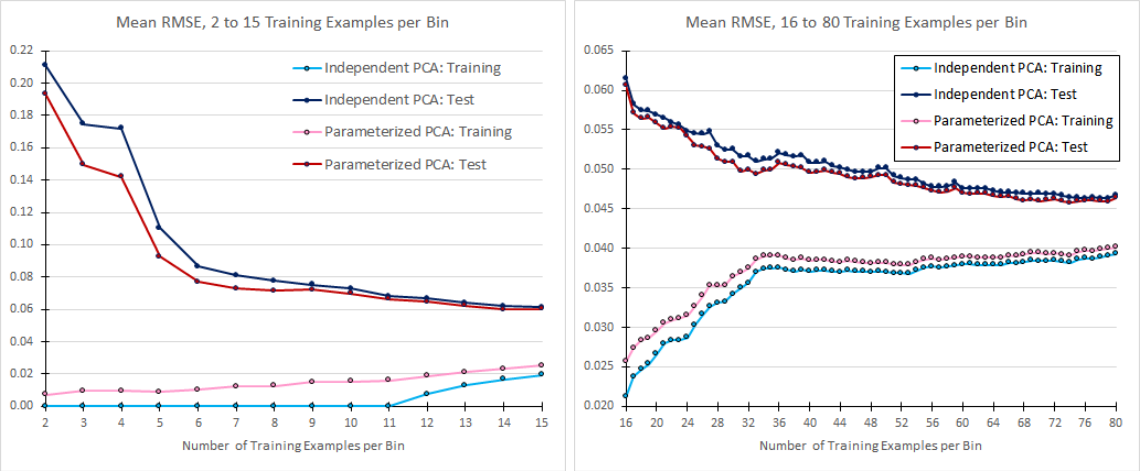

Figure 5.4: Mean RMSE for Projection of Test Set Blurred Facial Images, Using Varied Numbers of Training Examples (Left: 2-15, Right: 16-80)

professional photographers using high-grade cameras can produce images with blurred faces in the background. The blurriness of a facial image changes one's expectations for the face's appearance, as well as the types of variation in the appearance, so we modeled facial images using a parameter based on blurriness.

To quantify blurriness, we assumed that a Gaussian blur filter could approximate the transformation from an unobserved, unblurred image to the observed, blurred image. This Gaussian blur is a convolution using the kernel $K(x, y|\sigma)$ from Equation (5.4), where x is the horizontal distance and y is the vertical distance between the two pixels involved. We used σ from Equation (5.4) (with $\sigma = 0$ for an unblurred image) as the PPCA parameter, because higher values of σ create blurrier images.

$$K(x, y|\sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{5.4}$$

We treated the facial images from the CBCL Face Database #1 [12] as unblurred, and added Gaussian blur with a 7×7 kernel and varied σ. The database had 472 faces chosen as the test set, and the remaining 2,429 as the training set. We used three bins for σ: 0-1, 1-2, and 2-3. The training and test images were created from the original faces such that each original face produced one blurred image for each of the three bins. The parameter σ for each observation was selected randomly from a $U(0, 1)$, $U(1, 2)$, or $U(2, 3)$ distribution, depending on which bin's image was being produced.

We used 10 basis vectors for each IPCA bin or PPCA bin endpoint. For PPCA, we used gradient descent for both the mean and basis vectors. We used $\lambda_{r,m} = 0.6$, $\lambda_{r,v} = 2$, $\lambda_o = 1000$, $n_c = 300$, $n_m = 100$, $n_v = 100$, $\alpha_m = 0.01$, and $\alpha_v = 0.0001$. The training set varied from 2 to 80 examples per bin. For all sizes of training set, all bins had images made from the same faces, but had different added blur according to the values σ. The smaller training sets were always subsets of the larger training sets, to allow for better examination of the effect of the training set size. Figure 5.4 shows the mean across training or test set images for the RMSE of the blurred images' projections. PPCA had lower projection error on the test set than IPCA for each training size from 2 to 80 examples per bin, but had a more noticeable advantage when both methods used eight or fewer training examples per bin.

5.4 Facial Images with Rotation

Section 5.3 addressed the challenges of modeling facial images with different levels of blurriness. A separate challenge in face modeling is out-of-plane rotation, which changes the expected appearance of facial features and produces predictable changes in the occlusion of important facial features. Yaw rotation is highly prevalent in photos, particularly for "in the wild" photos, which are those taken in uncontrolled settings, often by ordinary users. One could model pitch or roll rotation instead using PPCA, but we focus on yaw rotation because it has the largest variation in available face images.

5.4.1 Background

Linear models for facial appearances exist, such as active appearance models (AAMs) [4] and 3D morphable models (3DMMs) [2]. AAMs typically incorporate in-plane rotation and suffer from an inability to model out-of-plane rotation, but 3DMMs exist in 3D and can use 3D rotation. 3DMMs can be fit to 2D test images, but they are trained using 3D facial scans performed in a laboratory setting. Potential users typically do not have the necessary equipment for these scans, and even with the equipment, one has very limited training data relative to a dataset of 2D images. Furthermore, applications for in-the-wild images grow as these images become more important for social media and other Internet uses, and the laboratory setting on the scan data makes them dissimilar to in-the-wild images. Zhu and Ramanan (2012) showed that training on in-the-wild

images greatly increases face detection performance on in-the-wild test data [19], and it seems logical that similarity between training and test data would be desirable for face modeling as well.

Gross, Matthews, and Baker (2004) modify AAMs to address occlusion [7]. This does not distinguish occlusion by an object (such as a hand in front of a face) from self-occlusion caused by out-of-plane rotation, so it does not take advantage of the more predictable nature of rotation-based self-occlusion. Xiao et al. (2004) also modify AAMs, creating a hybrid of a 2D and a 3D model by adding extra parameters and constraints to a 2D model [17]. It allows the training advantages of a 2D model with some of the advantages of a 3D model, but compared to PPCA, it does not address out-of-plane rotation as directly and relies more on 3D elements not directly observable in the 2D data.

AAMs and 3DMMs each incorporate two linear models: one for the shape mesh, and one for the appearance after removing the influence of shape variation. AAMs typically use frontal images only and translate the appearance from the original image's shape mesh to the mean shape mesh by triangular warping. Yaw rotation creates predictable changes to both the shape and the appearance. PPCA could model both, but we chose to focus on the appearance component, and modeled the shape using a rigid, 3D shape model built on other data.

5.4.2 Data

We used 272 human facial images from the Annotated Facial Landmarks in the Wild (AFLW) database [9], which includes annotations of the locations and occlusion status of 21 key points. We chose the subset such that the faces were all in color and appeared to be of 272 different people. We used yaw rotation in radians as the PPCA parameter θ. It was limited to the range from -0.5π to 0.5π, and we divided the range into 16 equally-sized bins. Our subset of AFLW had 17 images in each bin, and three images per bin were selected randomly to be in the test set. The remaining 14 images per bin were eligible for training, but we varied the training set size from 2 to 14 images per bin. The smaller training sets were always subsets of the larger training sets. Values of θ came from finding the roll, pitch, and yaw angles that best rotated the rigid shape model to fit the unoccluded key points' horizontal and vertical coordinates. Several yaw angles and key point locations were corrected manually.

AAMs commonly use a triangulation of the face to translate a shape mesh of key points into a shape that can cover pixels. We also used a triangulation, which we constructed manually to have

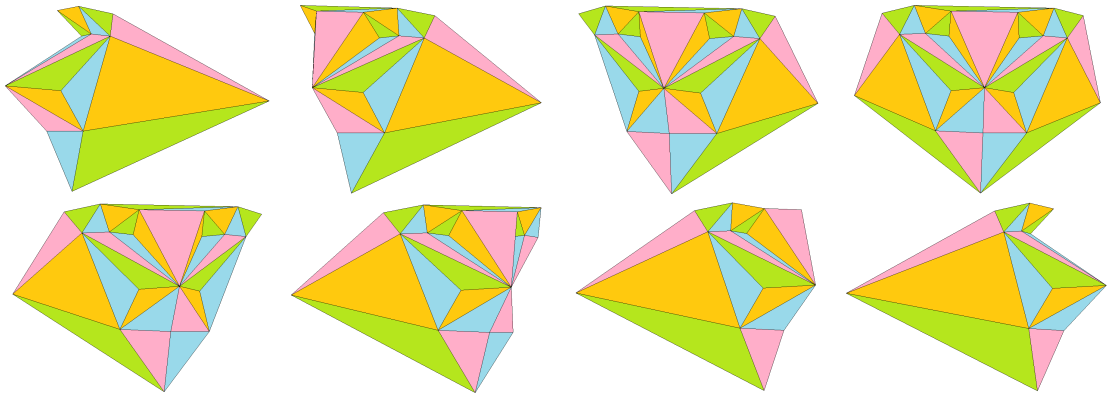

Figure 5.5: Triangulation at Bin Endpoints 2, 5, 8, 9, 10, 12, 14, and 16

triangles that are less likely to have one of three vertices occluded at yaw angles from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$. This generally implied triangles that ran more vertically than in automatic triangulation methods. PPCA promotes the smoothness of adjacent bin endpoints, so the triangles needed to use pixels that corresponded to equivalent areas in other bin endpoints' shapes. We calculated the triangle's area for each bin endpoint shape in our 3D model, and used the largest-area version of the triangle for PPCA. We warped each triangle from the original images to these model triangles, which were considered occluded or not based on the direction of the normal vector to that triangle in the rigid shape model rotated to the appropriate yaw angle. AFLW's image-specific occlusion annotations were not used after estimating θ.

We tested two numbers of basis vectors per bin endpoint, 4 and 10, to determine whether this modeling choice affects the appropriateness of PPCA. IPCA bins used the same number of basis vectors as PPCA bin endpoints did. We also allowed the model to vary based on the presence of whitening. The intensities before whitening were represented as double floating-point numbers from zero (black) to one (white). If whitening were used, each image would get six additional parameters in its representation, which were not a part of PPCA (or IPCA) itself. After warping an image, we stored the original mean intensity and standard deviation for red, green, and blue. We translated and rescaled the intensities such that each color had a mean of 0.5 and a standard deviation of 0.031. The latter was chosen to be just large enough to keep all whitened intensities within the [0, 1] interval. PPCA and IPCA modeled the whitened versions, and after projecting

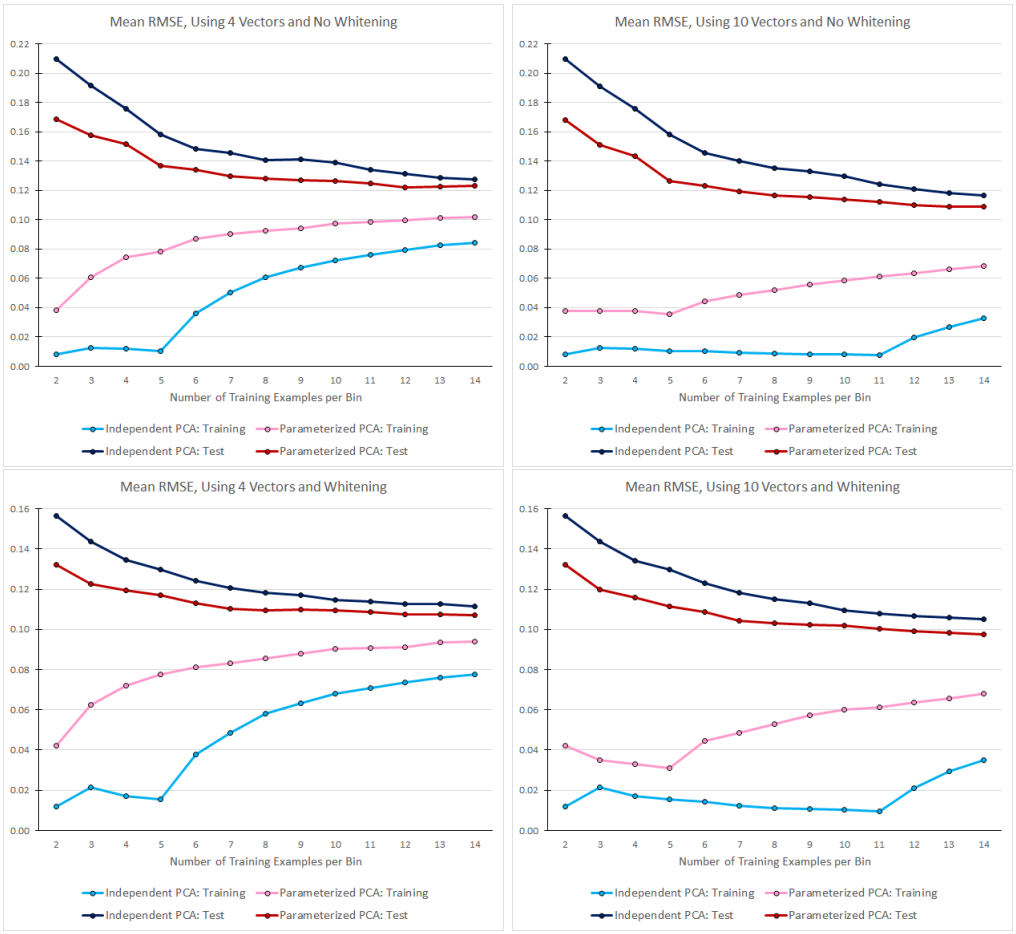

Figure 5.6: Mean RMSE for Projection of Facial Images with Yaw Rotation Parameter, Evaluated on Training and Test Sets Using Varied Numbers of Training Examples

the whitened image, we reversed the whitening transformation using the image-specific means and standard deviations by color.

5.4.3 Model Fitting and Results

We trained models with training set sizes from 2 to 14 examples per bin. PPCA needed to use gradient descent to optimize both the mean and basis vectors. We used $\lambda_{r,m} = 0.001$, $\lambda_{r,v} = 0.01$, $\lambda_o = 1000$, $n_c = 200$, $n_m = 100$, $n_v = 250$, $\alpha_m = 0.0001$, and typically $\alpha_v = 10^{-6}$. The models with two to four training examples per bin and 10 basis vectors required smaller α_v to avoid divergence. The occlusion of each triangle was considered known, because it was treated as a function of a known yaw angle. So, we set the image-specific mean vector and basis vectors in IPCA and PPCA

Figure 5.7: Mean Facial Images by Rotation-Based Bin (or Bin Endpoint) for IPCA (Left) and PPCA (Right), Using No Whitening

Figure 5.8: Mean Facial Images by Rotation-Based Bin Endpoint for PPCA, Using Higher Roughness Penalty and No Whitening

projections to have zeros for any out-of-shape pixels before we found images' coefficients. After projecting the image and reversing whitening if it was used, we calculated the RMSE for each image in the training and test sets.

Figure 5.6 shows the means of these RMSEs, which are averaged across the images of the training or test set. For all four styles of fitting the models and for all training set sizes tested, IPCA overfits the data relative to PPCA. PPCA has higher error on the training set but lower error on the test set than IPCA does. The difference is even more noticeable when we use eight or fewer training examples per bin.

Figure 5.7 shows the IPCA and PPCA mean vectors, warped to the bin midpoint (IPCA) or bin endpoint (PPCA) shapes. These models used four vectors per bin (or bin endpoint), no whitening, and 12 training examples per bin. The IPCA means appear to treat characteristics of the training images as characteristics of the bin to a higher degree than the PPCA means do. One can see

more noticeable changes from bin to bin for IPCA with respect to eye color and shape, lip color, illumination, and skin complexion. The smoothness of the mean shape can be improved further for PPCA by increasing the penalty $\lambda_{r,m}$ to 0.1, as shown in Figure 5.8. We did not test additional training set sizes with $\lambda_{r,m} = 0.1$, but for this example, the mean RMSE for the test set (0.1220) was effectively the same as for $\lambda_{r,m} = 0.001$ (mean RMSE = 0.1220). Both had lower mean RMSEs for projection error than IPCA (0.1313) did.

CHAPTER 6

CLUSTERING FOR PARAMETER-SENSITIVE FACE MODELING

In Section 5.4, we presented the challenge of modeling the appearances of faces with varied yaw rotation angles. If one knew the yaw angles or could detect them suitably, then one could use parameterized principal component analysis (PPCA) with the yaw angle as the parameter. We had a dataset from a 272-image subset of the Annotated Facial Landmarks in the Wild (AFLW) database [9], which had a much more uniform distribution of yaw angles from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$ than the full AFLW database did. The dataset also had detected yaw angles, as well as manual corrections both to yaw angles and to annotated locations of key points.

In this section, we investigated whether clustering could produce an independent principal component analysis (IPCA) model that could also model such data effectively like PPCA does. If so, the cluster-based IPCA model could be applied to a situation involving a population thought to be sensitive to one or more parameters that are difficult to detect. Clusters could also potentially capture multiple types of information that might affect the observations, including categorical information. For the example of rotated faces, clusters might capture quantitative information about pitch, roll, and yaw rotation, as well as categories such as hair color and the presence of glasses.

Lauer and Schnörr (2009) applied spectral clustering to motion segmentation [10], which under the affine camera model is a subspace separation problem with four-dimensional subspaces. They had very high segmentation accuracy and very low run-time because of differences from other segmentation methods based on spectral clustering. We attempted to use their method to separate subspaces to create a piecewise linear model, although our subspaces were not necessarily four-dimensional.

In spectral clustering of N observations, one first produces an $N \times N$ affinity matrix $\boldsymbol{A}$ with elements A_{ij}. One makes $\boldsymbol{L} = \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{A} \boldsymbol{D}^{-\frac{1}{2}}$, where the diagonal matrix $\boldsymbol{D}$ has non-zero elements $D_{ii} = \sum_{j=1}^{N} A_{ij}$. One makes a matrix with the eigenvectors of $\boldsymbol{L}$ as column vectors, and normalizes

the rows to get the matrix $\boldsymbol{U}$. One clusters the rows of $\boldsymbol{U}$. Multiple algorithms are possible, but Lauer and Schnörr used the typical choice of k-means clustering. The cluster assignments for the rows of $\boldsymbol{U}$ are used as the cluster assignments for the original observations.

Lauer and Schnörr's method differed from previous methods in that they used an affinity matrix with correlations raised to even-integer powers as its elements A_{ij}. These are shown in Equation (6.1), and require the user to choose a parameter $\alpha \in \mathbb{N}$ based on tuning.

$$A_{ij} = \left(\frac{\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle}{\|\boldsymbol{x}_i\| \|\boldsymbol{x}_j\|} \right)^{2\alpha} \tag{6.1}$$

This affinity matrix, unlike the radial affinity matrix often used before, could separate intersecting subspaces such as lines. It could segment higher dimension subspaces than lines if one had enough training examples per subspace to ensure adequate angular density. Like several previous methods, Lauer and Schnörr's method used singular value decomposition (SVD) to reduce the dimension of the ambient space. Unlike previous authors, Lauer and Schnörr suggested keeping a high ambient dimension and introduced an eigenvalue-based method to detect this dimension.

6.1 Data

To test the effectiveness of a clustering-based model, we used the same 272-image set of human face images that we used in Section 5.4. However, we added two key points in the bottom-left and bottom-right areas of the face, using formulas based on the chin and ears. We used the same triangulation of the face as in Section 5.4, except that one triangle in the bottom-left and one

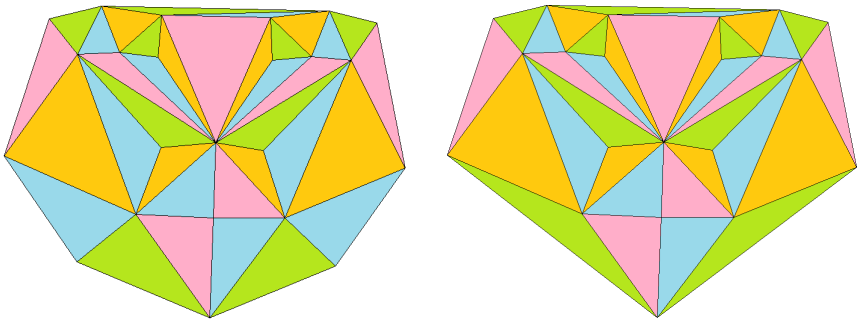

Figure 6.1: Triangulation of 23-Point Shape Mesh Used (Left) and 21-Point Shape Mesh from Section 5.4 (Right)

Figure 6.2: Example Warped Face Appearances without Adjustments for Occlusion

triangle in the bottom-right each got split into two triangles to accommodate the two added key points. The triangulation used is shown in the left half of Figure 6.1.

We modeled the shape mesh and the appearances, as separate tasks. We first aligned all the shape meshes using a similarity transformation. This rotates, scales, and translates points $(x_{\text{old}}, y_{\text{old}})$ to produce $(x_{\text{new}}, y_{\text{new}})$ as in Equation (6.2), such that the points $(x_{\text{new}}, y_{\text{new}})$ are similar to corresponding points from a reference shape mesh.

$$\begin{bmatrix} x_{\text{new}} \\ y_{\text{new}} \end{bmatrix} = s \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x_{\text{old}} \\ y_{\text{old}} \end{bmatrix} \tag{6.2}$$

Above, t_x and t_y are translations, s is a scaling constant, and ϕ is the angle of in-plane rotation. The reference shape mesh used was the unrotated, 23-point version of the rigid shape model used in Section 5.4.

For the appearances, we warped the images from the annotated, 23-point shape meshes to the unrotated rigid shape model. We whitened the appearances such that each color's intensities had a mean of 0.5 and a standard deviation of 0.027, which kept all whitened intensities in the range [0, 1] like the original images. When calculating projection error, we reversed the whitening transformation on the projection, and compared this to the original image, like we had done for the whitened appearances in Section 5.4. The appearances needed to use the same shape, because these faces were being clustered as though the yaw angles were unknown. However, we did use the known angles, rather than the annotated statuses of points as occluded or not, to modify triangles for occlusion. The results appeared to be similar to the use of annotated occlusion information.

For occluded triangles, we set all pixels' appearance intensity to 0.5, so PCA would not treat these as important sources of variation. If, instead, we had included these pixels as though they were unoccluded, there would be large variations in appearances of occluded regions from duplicating

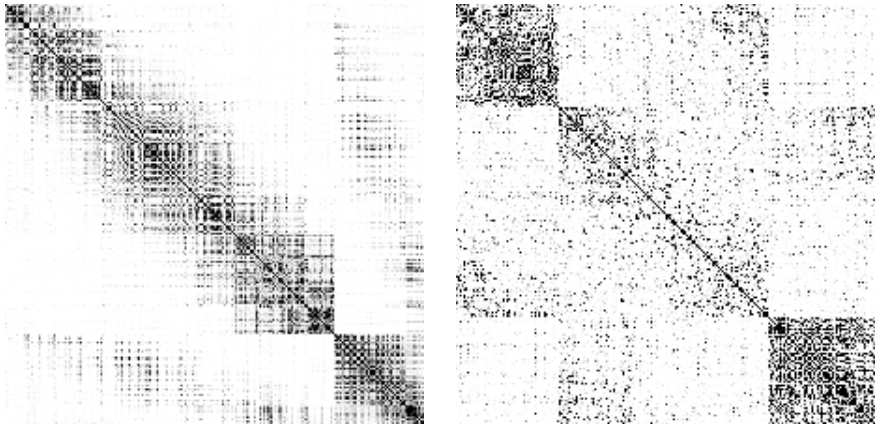

Figure 6.3: Affinity Matrices for Shapes at 5 Dimensions with $\alpha = 3$ (Left) and Appearances at 20 Dimensions with $\alpha = 3$ (Right)

pixels in unoccluded regions. Profile faces tended to produce two copes of the unoccluded half of the face, and mid-profile faces tended to stretch pixels from unoccluded regions. Some examples are shown in Figure 6.2. When evaluating the projections of these faces, we again excluded the occluded triangles.

6.2 Methods

We evaluated three styles of PCA-based projections on the shape and appearance information. One was PCA, without any groupings. The other two were both IPCA methods. One used bins for the yaw angle, just like the IPCA reference method in Section 5.4, but with fewer bins. The other used clusters from Lauer and Schnörr's spectral clustering method to produce groups. We used the correlation-based affinity matrices, k-means clustering, and SVD-based dimension reduction from [10], but evaluated several manually chosen dimensions of the manual ambient space instead of using the automatic, eigenvalue-based method. Preliminary experiments suggested that the automatic method to select an ambient space dimension was much less effective for face modeling than for motion segmentation, which had known four-dimensional subspaces. The method requires estimated lower and upper bounds for the subspace dimension, which were too subjective for the face appearance data. The eigenvalues involved also tended to be overly sensitive to noise for these data.

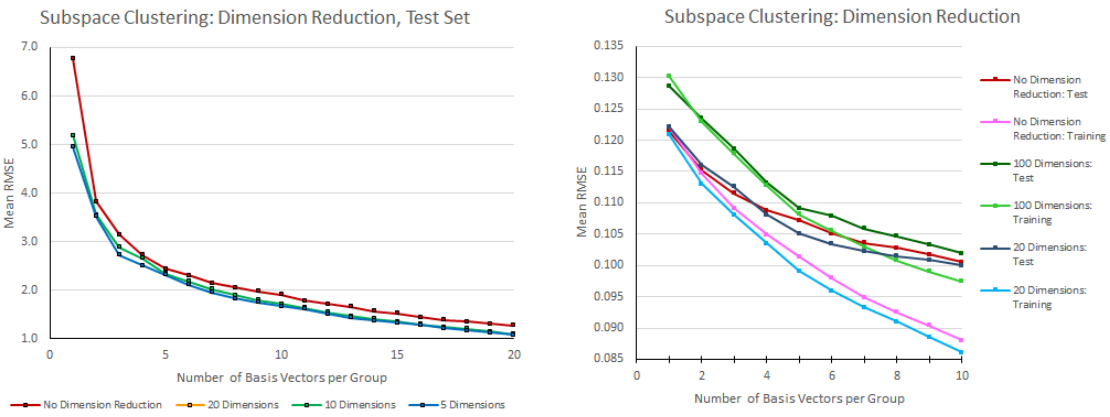

Figure 6.4: Mean RMSE for Spectral Clustering Using Different Ambient Space Dimensions for 4 Shape Clusters (Left) and 3 Appearance Clusters (Right)

For the shape data, we tested the use of no dimension reduction for the ambient space, as well as reduction to 5, 10, and 20 dimensions. For the appearance data, we tested no dimension reduction, 100 dimensions, and 20 dimensions. Tuning of the affinity matrices revealed that $\alpha = 1$ looked best for the appearances with no dimension reduction. We chose $\alpha = 3$ for the shapes at 5 and 10 dimensions and for the appearances at 20 and 100 dimensions. For the shapes at 20 dimensions, $\alpha = 4$ looked the most appropriate. If using no dimension reduction for the shapes, $\alpha = 6$ looked the best. Figure 6.3 shows some sample affinity matrices, after scaling the intensities and inverting the style of display (using one minus the scaled affinity matrix) so the strongest affinities are black. The scaling was chosen to make the means of all such matrices approximately 0.9. These rows have also been rearranged so observations in the same cluster are adjacent to each other. This rearrangement used the assignments to four shape clusters and three appearance clusters. One wants to raise α high enough to reduce noise, without making α so high that some observations are not connected to clusters. Our choices for α were made because they appeared to strike that balance.

When using k-means clustering on these affinity matrices, we ran the k-means algorithm 1000 times. Each time, the initial cluster centroids were randomly-chosen rows from the affinity matrix. We used the k-means result that, of the 1000 choices, had the lowest sum of observations' squared ℓ_2 distances to their assigned cluster centroids. The distances used the affinity matrix values rather than the original observations.

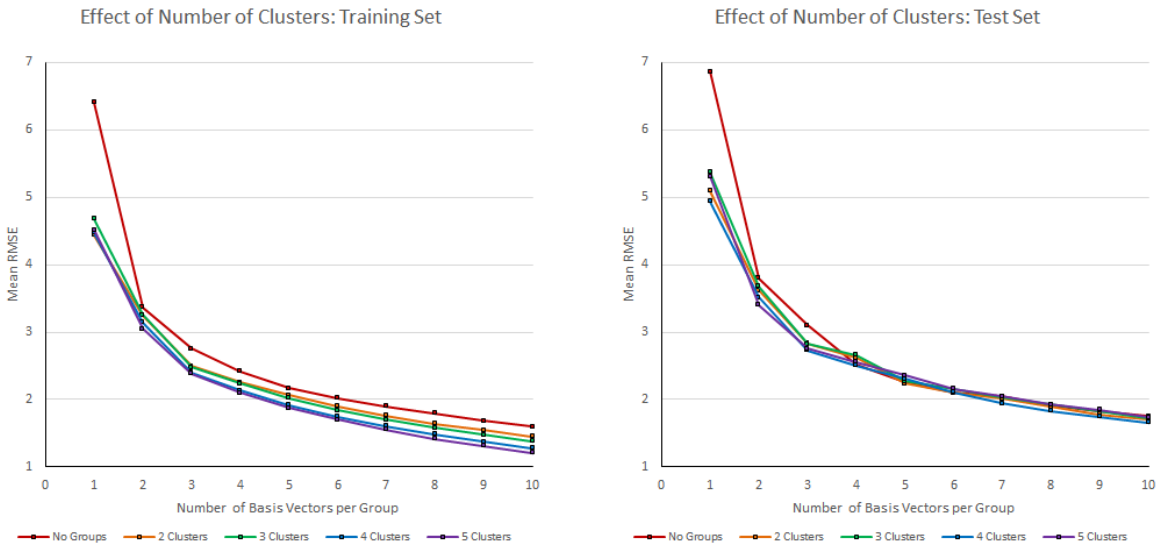

Figure 6.5: Mean RMSE of Projected Shape Mesh Coordinates for Training (Left) and Test (Right) Sets

6.3 Results

First, we compared potential dimensions for the ambient space. Figure 6.4 shows the mean of the test set observations' RMSEs for clustering of shape meshes, as well as the mean of training or test set observations' RMSEs for clustering of appearances. The shape's training set RMSEs are not shown because they were too similar to the test set RMSEs, making it difficult to compare dimensions on the test set. They were similar to the test set RMSEs, though, in terms of which ambient space dimensions performed better and by how much. Unlike in [10], low dimensions performed better if using SVD dimension reduction. We also tested a mean shift without any dimension reduction, and for appearances, this performed noticeably better than reduction to 100 dimensions. For the shape data, the version with no dimension reduction fared the worst of all tested methods. We chose dimension reduction of the ambient space to five dimensions for the shape meshes, and to 20 dimensions for the appearances. In both cases, these were the lowest-dimension versions tested. The discrepancy from the findings in [10] may be attributable to how we examined the fit of projections using the segmentation, rather than the segmentation accuracy itself.

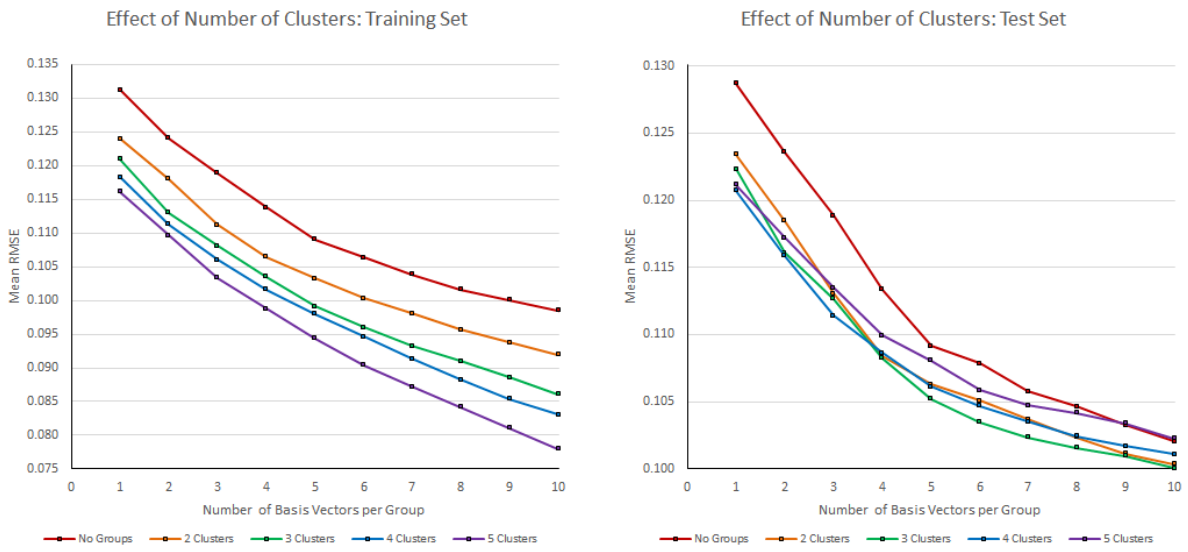

Figure 6.6: Mean RMSE of Projected Appearances for Training (Left) and Test (Right) Sets

For the selected ambient space dimensions, we also evaluated potential numbers of clusters. As expected, more clusters led to better performance on the training set, as shown in Figures 6.5 and 6.6. Models with five or more clusters performed poorly on the test set, though. The five-cluster model only outperformed the two-cluster model if both model types used two to four basis vectors per cluster on the shape data, or if both used one to two basis vectors per cluster on the appearance data. Models with more than five clusters showed even worse performance on the test set than the five-cluster model. High numbers of groups also seemed inappropriate for these data, because Lauer and Schnörr's correlation-based affinities require sampling each group at high angular density to ensure appropriate connectedness of training examples for high-dimensional manifolds.

We selected four clusters as best for the shapes, and three clusters as best for the appearances. The four-cluster model also looked very good for the appearances, particularly if using very few basis vectors per cluster. For the chosen clustering methods, the appearances had at least 55 training examples in each cluster, and the shapes had at least 48 training examples in each cluster. The parameter-based bins had at least 73 training examples in each bin if using three bins, and 56 if using four bins.

For both shape and appearance data, the two IPCA methods (spectral clustering and binning of the yaw parameter) were very similar in performance, and the PCA model was surprisingly

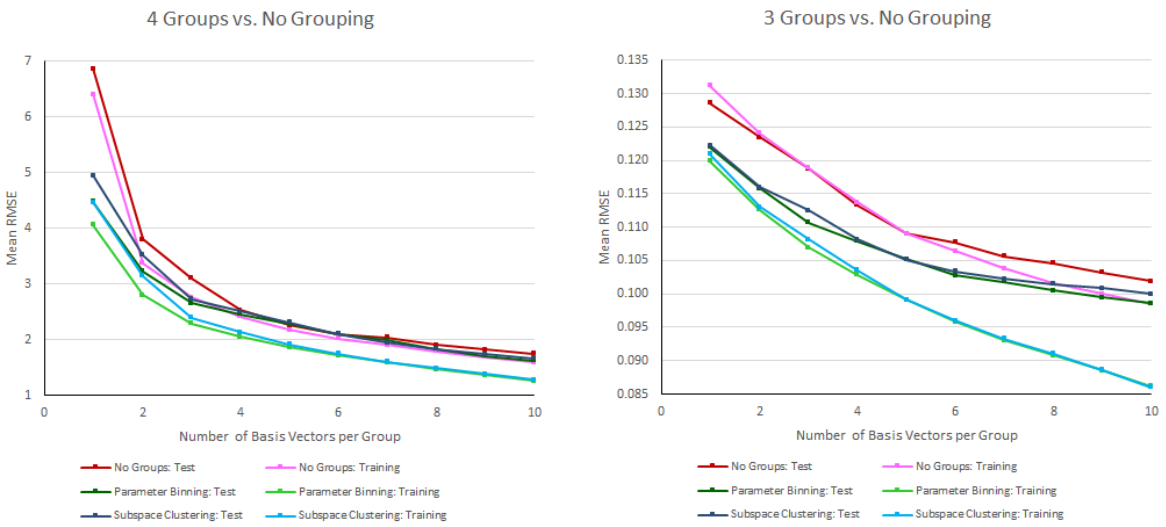

Figure 6.7: Mean RMSE of Projected Shape Meshes (Left) and Appearances (Right) Using Grouped and Ungrouped PCA Methods

competitive on the test set. The ungrouped PCA model was particularly effective on the shape data, possibly because the shape meshes had already been aligned by a similarity transformation. Both IPCA models showed lower mean RMSEs than the PCA model for the training sets on both the shape and appearance data, and for the test set of the appearance data, if the models used the same number of basis vectors per group. For the test set shape data, the IPCA models clearly had lower mean RMSEs if using three or fewer basis vectors per group, but the models were generally comparable if using four or more basis vectors per group. The PCA model even outperformed both IPCA models if using four or six basis vectors per group, and outperformed the parameter-binning IPCA model if using five basis vectors per group.

In general, subspace clustering appears to be an adequate replacement for the yaw parameter information on these shape and appearance data, but does not outperform the binning-based IPCA model consistently. The ungrouped PCA model also generally appears to be competitive, and the use of additional basis vectors appears to be more valuable for the PCA model than for the IPCA models. The PCA model continues to show large gains in performance with each extra basis vector after 10 basis vectors, whereas the IPCA models continue to improve but the rate decreases dramatically after a few basis vectors. For the shape data, the IPCA models' mean RMSEs level off

Table 6.1: Required Model Size for PCA to Match or Outperform IPCA

IPCA Model	IPCA Model Size	Required PCA Model Size			
		Clustering, Training Set	Clustering, Test Set	Binning, Training Set	Binning, Test Set
Shape, 4 Groups	8	3	3	3	3
	12	4	4	4	4
	16	5	6	5	6
	20	6	7	6	7
	24	6	9	6	8
	28	7	10	7	10
	32	9	12	9	11
	36	10	13	10	13
	40	12	14	12	14
	44	13	16	13	15
Appearance, 3 Groups	6	4	4	4	4
	9	6	5	6	5
	12	7	6	7	6
	15	9	7	9	7
	18	11	9	11	9
	21	13	10	13	11
	24	15	11	15	12
	27	16	12	16	13
	30	18	12	18	14
	33	20	14	20	15

considerably after three basis vectors. For the appearance data, the IPCA models' rate of decrease in the mean RMSE slows noticeably after five basis vectors.

If one were evaluating the models based on both projection error and complexity, then the PCA model might even be considered the best model, depending on the measure of complexity. Table 6.1 shows the necessary model size for a PCA model to achieve a mean RMSE less than or equal to what an IPCA model of a given size produces. The model size refers to the total number of mean and basis vectors in the model. For example, an IPCA model with two groups and one basis vector per group would have a model size of four, because it has two mean vectors (one per group) and two basis vectors (one per group). The PCA model outperforms the IPCA model of the same model size in all cases on both the training and test set data, regardless of whether the IPCA model uses spectral clustering or binning of the yaw parameter to form groups.

Table 6.2: Required Number of Coefficients for PCA to Match or Outperform IPCA

IPCA Model	IPCA Complexity (One Plus Number of Coefficients)	Required PCA Number of Coefficients			
		Clustering, Training Set	Clustering, Test Set	Binning, Training Set	Binning, Test Set
Shape, 4 Groups	2	2	2	2	2
	3	3	3	3	3
	4	4	5	4	5
	5	5	6	5	6
	6	5	8	5	7
	7	6	9	6	9
	8	8	12	8	10
	9	9	12	9	12
	10	11	13	11	13
	11	12	15	12	14
Appearance, 3 Groups	2	5	5	5	5
	3	7	6	7	6
	4	8	7	8	7
	5	10	8	10	8
	6	12	10	12	10
	7	14	11	14	12
	8	16	12	16	13
	9	17	13	17	14
	10	19	13	19	15
	11	21	15	21	16

If, however, the measure of complexity were about the dimension to represent an observation, then the IPCA models might be considered superior. Consider a dimension measure which would be the size of an observation's coefficient vector for an ungrouped model, and would be the size of an observation's group-specific coefficient vector plus one (for an element to store the observation's group number) for a grouped model. Table 6.2 shows that in this case, the IPCA models are both superior to the PCA model in all tested cases on the appearance data. For the shape data, the IPCA models are considered better than the PCA model in some cases only. On the training set, the PCA model is generally better, and the IPCA models are only better if using nine or more basis vectors per group. On the test set, the IPCA models are generally better, and outperform the PCA model if using three or more basis vectors per group.

CHAPTER 7

CONCLUSION

We have presented a novel method, parameterized principal component analysis (PPCA), for modeling multidimensional data on linear manifolds that vary smoothly according to a contextually important parameter θ. We compared PPCA to independent principal component analysis (IPCA), the use of separate principal component analysis models for groups formed by values of the parameter θ. We showed that PPCA outperformed IPCA at recovering known true mean vectors and true basis vectors based on smooth functions of the parameter θ, and at producing lower projection error on three datasets. These datasets contained lymph node shapes that varied by the diameter, blurred human facial images that varied by the standard deviation σ of the Gaussian blur applied, and human facial images that varied by the angle of yaw rotation. In each of the three datasets, PPCA's performance on the test set was the strongest relative to IPCA when the two methods used smaller training sets.

We also tested the use of spectral clustering with a correlation-based affinity matrix for forming an IPCA model's groups, with applications to the shape mesh and appearance (on a frontal shape) of facial images that varied by the yaw rotation angle. Spectral clustering appeared to be able to capture the information from the variation in yaw rotation without direct access to the observations' yaw angles. IPCA models using these clusters produced very similar projection error to IPCA models using bins of the observations' yaw angles, if both model types used the same number of groups and manifold dimension. Both styles of IPCA models were generally able to outperform a single PCA model if all three styles of model used the same number of basis vectors per group, but the ungrouped PCA model was surprisingly effective. For both the shape and appearance data, an ungrouped PCA model could outperform either type of IPCA model if the PCA and IPCA models used the same total number of mean and basis vectors.

BIBLIOGRAPHY

[1] Adrian Barbu, Michael Suehling, Xun Xu, David Liu, S Kevin Zhou, and Dorin Comaniciu. Automatic detection and segmentation of lymph nodes from ct data. *Medical Imaging, IEEE Transactions on*, 31(2):240–250, 2012.

[2] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194. ACM Press/Addison-Wesley Publishing Co., 1999.

[3] Songcan Chen and Tingkai Sun. Class-information-incorporated principal component analysis. *Neurocomputing*, 69(1):216–223, 2005.

[4] Timothy F Cootes, Gareth J Edwards, and Christopher J Taylor. Active appearance models. *IEEE Transactions on pattern analysis and machine intelligence*, 23(6):681–685, 2001.

[5] Fernando De la Torre and Takeo Kanade. Multimodal oriented discriminant analysis. In *Proceedings of the 22nd international conference on Machine learning*, pages 177–184. ACM, 2005.

[6] Andrew B Goldberg, Xiaojin Zhu, Aarti Singh, Zhiting Xu, and Robert Nowak. Multi-manifold semi-supervised learning. In *International Conference on Artificial Intelligence and Statistics*, pages 169–176, 2009.

[7] Ralph Gross, Iain Matthews, and Simon Baker. Constructing and fitting active appearance models with occlusion. In *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on*, pages 72–72. IEEE, 2004.

[8] Xiaofei He and Partha Niyogi. Locality preserving projections. In *Neural information processing systems*, volume 16, page 153. MIT, 2004.

[9] Martin Koestinger, Paul Wohlhart, Peter M. Roth, and Horst Bischof. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization, 2011.

[10] Fabien Lauer and Christoph Schnörr. Spectral clustering of linear subspaces for motion segmentation. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 678–685. IEEE, 2009.

[11] Aleix M Martínez and Avinash C Kak. Pca versus lda. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(2):228–233, 2001.

[12] MIT Center For Biological and Computation Learning. Cbcl face database #1, 2000. Accessed: 2016-04-07.

[13] Nikolaos Pitelis, Chris Russell, and Lourdes Agapito. Learning a manifold as an atlas. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1642–1649. IEEE, 2013.

[14] René Vidal, Yi Ma, and Shankar Sastry. Generalized principal component analysis (gpca). In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–621. IEEE, 2003.

[15] Rene Vidal, Yi Ma, and Shankar Sastry. Generalized principal component analysis (gpca). *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(12):1945–1959, 2005.

[16] Elif Vural and Pascal Frossard. Learning smooth pattern transformation manifolds. *Image Processing, IEEE Transactions on*, 22(4):1311–1325, 2013.

[17] Jing Xiao, Simon Baker, Iain Matthews, and Takeo Kanade. Real-time combined 2d+ 3d active appearance models. In *CVPR (2)*, pages 535–542, 2004.

[18] Daoqiang Zhang, Zhi-Hua Zhou, and Songcan Chen. Semi-supervised dimensionality reduction. In *SDM*, pages 629–634. SIAM, 2007.

[19] Xiangxin Zhu and Deva Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2879–2886. IEEE, 2012.

BIOGRAPHICAL SKETCH

Ajay Gupta has been involved heavily in statistics since earning a B.S. degree in statistics, with an option in actuarial science, from Case Western Reserve University. Ajay worked extensively as an actuary after receiving his B.S., and left to join the Ph.D. program in statistics at The Florida State University. Ajay is still a Fellow of the Society of Actuaries and a Member of the American Academy of Actuaries, but no longer works in actuarial science.

He has been a devoted teacher while at The Florida State University, teaching the applied statistics courses labeled Statistics through Example, Fundamentals of Business Statistics, and Statistics for Biology. He is very interested in applied statistics, particularly with financial data, image data, and sports data. While at The Florida State University, his published and unpublished work have generally revolved around image data, particularly of human faces. Once attaining his degree, Ajay hopes to resume work in business and to help his three-year-old daughter learn important skills, which eventually will include statistics.